Higher-Order Finite Element Methods

PAVEL ŠOLÍN, *Rice University, Houston, Texas*
KAREL SEGETH, *Academy of Sciences of the Czech Republic, Prague*
IVO DOLEŽEL, *Czech Technical University, Prague*

CHAPMAN & HALL/CRC

A CRC Press Company
Boca Raton London New York Washington, D.C.

Library of Congress Cataloging-in-Publication Data

Šolín, Pavel.
Higher-order finite element methods / Pavel Šolín, Karel Segeth, Ivo Doležel.
p. cm.
Includes bibliographical references and index.
ISBN 1-58488-438-X
1. Finite element method. I. Segeth, Karel. II. Doležel, Ivo. III. Title. IV. Series.

TA347.F5+.S68 2003
620′.001′51535—dc21 2003051470

International Standard Book Number 1-58488-438-X
Library of Congress Card Number 2003051470
Printed in the United States of America 1 2 3 4 5 6 7 8 9 0
Printed on acid-free paper

We dedicate this book to the memory of Prof. Jindřich Nečas (December 14, 1929 – December 5, 2002), an outstanding Czech mathematician and a world-renowned authority in the field of partial differential equations and modern functional analysis.

Prof. Jindřich Nečas contributed substantially to the development of modern functional analytic methods of solution to elliptic partial differential equations in his famous monograph *Les méthodes directes en théorie des équations elliptiques* (1967). He followed the modern Italian and French school and enhanced it with important results, for example, by a new "algebraic" proof of general inequalities of Korn's type and generalized regularity results. A few years later, in 1973, he published with collaborators the monograph *Spectral Analysis of Nonlinear Operators*, which aroused great interest. Prof. Nečas was always intrigued by the problem of regularity of solutions. Outstanding results in this field appeared in his book *Introduction to the Theory of Nonlinear Elliptic Equations* (1983, 1986).

From the very beginning Prof. Nečas devoted great effort to applications in mathematical physics and engineering. In 1967 he established a seminar on problems of continuum mechanics that continues to the present day. From this seminar came the monographs *Mathematical Theory of Elastic and Elastoplastic Bodies: An Introduction* (1981, 1983) and *Solution of Variational Inequalities in Mechanics* (1982). The latter book was translated into Russian (1986) and English (1988). Both these monographs also were directed toward numerical methods of solution based on the finite element method. This prompted P. G. Ciarlet and J. L. Lions to invite Prof. Nečas to write an article, "Numerical Methods for Unilateral Problems in Solid Mechanics," for their *Handbook of Numerical Analysis* (1996).

During the last two decades of his life Prof. Nečas' field of interest changed from solid to fluid mechanics, in particular to problems of transsonic flow. Using the method of entropic compactification and the method of viscosity, he achieved remarkable results that he published in his monograph *Écoulements de fluide: Compacité par entropie* (1989). Recent results of Prof. Nečas and his collaborators have been collected in the book *Weak and Measure Valued Solutions to Evolutionary PDE's* (1996).

Besides the above-mentioned monographs, Prof. Nečas initiated and published more than 180 papers in outstanding mathematical journals and conference proceedings.

An excellent teacher, Prof. Nečas influenced many students and colleagues with his never-ending enthusiasm. He organized lectures, seminars and two series of summer schools, and guided many students on the way to their diplomas and Ph.D. theses. They all will remember him with gratitude.

Both P. Šolín and K. Segeth were, at different times, students of J. Nečas.

Preface

The finite element method is one of the most popular tools for the numerical solution of engineering problems formulated in terms of partial differential equations. The latest developments in this field indicate that its future lies in adaptive higher-order methods, which successfully respond to the increasing complexity of engineering simulations and satisfy the overall trend of simultaneous resolution of phenomena with multiple scales.

Among various adaptive strategies for finite elements, the best results can be achieved using goal-oriented *hp*-adaptivity. Goal-oriented adaptivity is based on adaptation of the finite element mesh with the aim of improving the resolution of a specific quantity of interest (instead of minimizing the error of the approximation in some global norm), and *hp*-adaptivity is based on the combination of spatial refinements (h-adaptivity) with simultaneous variation of the polynomial order of approximation (p-adaptivity). There are nonacademic examples where the goal-oriented *hp*-adaptivity turned out to be the only way to resolve the problem on a required level of accuracy (see, e.g., [185]). Automatic *hp*-adaptivity belongs to the most advanced topics in the higher-order finite element technology and it is subject to active ongoing research. We refer the reader to works by Demkowicz et al. (see [162, 64, 62, 8, 122, 149, 172, 191] and references therein). The goal of this book is more modest – we present the basic principles of higher-order finite element methods and the technology of conforming discretizations based on hierarchic elements in spaces H^1, $\boldsymbol{H}(\mathrm{curl})$ and $\boldsymbol{H}(\mathrm{div})$. An example of an efficient and robust strategy for automatic goal-oriented *hp*-adaptivity is given in Chapter 6.

In the introductory Chapter 1 we review the aforementioned function spaces and their basic properties, define unisolvency of finite elements, formulate conformity requirements for finite elements in these spaces, introduce the basic steps in the finite element procedure, and present several families of orthogonal polynomials. Section 1.3 is devoted to the solution of a one-dimensional model problem on a mesh consisting of elements of arbitrary polynomial order. The technical simplicity of the one-dimensional case gives the reader the opportunity to encounter all the important features of higher-order finite element discretization at the same time.

A database of scalar and vector-valued hierarchic master elements of arbitrary order on the most commonly used reference domains in 2D and 3D is provided in Chapter 2. This chapter contains many formulae of higher-order shape functions and is intended for reference rather than for systematic reading. Chapter 3 discusses the basic principles of higher-order finite element methods in two and three spatial dimensions that the reader was first exposed to in Section 1.3. We begin with generalizing the standard nodal interpolation to higher-order hierarchic elements, and describe the design of reference maps based on the transfinite interpolation technique as well as their polynomial isoparametric approximation. We discuss an approach to the treatment of constrained approximations (approximations comprising "hanging nodes") and mention selected software-technical aspects at the end of this chapter.

Chapter 4 is devoted to higher-order numerical quadrature in two and three spatial dimensions. Numerical quadrature lies at the heart of higher-order finite element codes and its proper implementation is crucial for their optimal performance. In particular the construction of integration points and weights for higher-order Gaussian numerical quadrature is not at all trivial, since they are not unique and the question of their optimal selection is extremely difficult. For illustration, each newly explored order of accuracy usually means a new paper in a journal of the numerical quadrature community. Tables of integration points and weights for all reference domains up to the order of accuracy $p = 20$ are available on the CD-ROM that accompanies this book.

Chapter 5 addresses the numerical solution of algebraic and ordinary differential equations resulting from the finite element discretization. We present an overview of contemporary direct and iterative methods for the solution of large systems of linear algebraic equations (such as matrix factorization, preconditioning by classical and block-iterative methods, multigrid techniques), and higher-order one-step and multistep schemes for evolutionary problems.

Chapter 6 presents several approaches to automatic mesh optimization and automatic h-, p- and hp-adaptivity based on the concept of *reference solutions*. Reference solutions are approximations of the exact solution that are substantially more accurate than the finite element approximation itself. We use reference solutions as robust error indicators to guide the adaptive strategies. We also find it useful to recall the basic principles of goal-oriented adaptivity and show the way goal-oriented adaptivity can be incorporated into standard adaptive schemes. The mathematical aspects are combined with intuitive explanation and illustrated with many examples and figures.

We assume that the reader has some experience with the finite element method – say that he/she can solve the Poisson equation ($-\triangle u = f$) in two spatial dimensions using piecewise-linear elements on a triangular mesh. Since it is our goal to make the book readable for both engineers and applied researchers, we attempt to avoid unnecessarily specific mathematical language whenever possible. Usually we prefer giving references to more difficult proofs rather than including them in the text. A somewhat deeper knowledge of mathematics (such as Sobolev spaces, embedding theorems, basic inequalities, etc.) is necessary to understand the theoretical results that accompany some of the finite element algorithms, but some of these can be skipped if the reader is interested only in implementation issues.

The first author is indebted to Prof. Leszek Demkowicz (ICES, The University of Texas at Austin) for many motivating discussions on theoretical issues related to the De Rham diagram, theory of higher-order finite elements and automatic *hp*-adaptivity. He further gratefully acknowledges the numerous suggestions of Prof. Jan Hesthaven (Division of Applied Mathematics, Brown University, Providence, RI), who despite his many other duties found time to read the whole manuscript. Especially noteworthy have been the ideas of Dr. Fabio Nobile (ICES, The University of Texas at Austin), who significantly influenced the structure of the first chapter. Deep appreciation goes to graduate student Denis Ridzal (Department of Computational and Applied Mathematics, Rice University, Houston, TX), who gave freely of his time in investigating the conditioning properties of higher-order shape functions for various types of finite elements in one and two spatial dimensions.

The authors would like to thank Prof. Ronald Cools (Departement Computerwetenschappen, Katholieke Universiteit Leuven, Belgium) for providing them with valuable information related to higher-order numerical quadrature and for his help with the review of Chapter 4. Many thanks are owed to Jan Haškovec (Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic), Dr. Petr Klouček (Department of Computational and Applied Mathematics, Rice University, Houston, TX), Dr. Dalibor Lukáš (Technical University of Ostrava, Czech Republic), Dr. Andreas Obereder (Institute of Industrial Mathematics, Johannes Kepler University, Linz, Austria), Dr. Tomáš Vejchodský (Mathematical Institute of the Academy of Sciences of the Czech Republic, Prague), and Martin Zítka (Faculty of Mathematics and Physics, Charles University, Prague) for their invaluable help with the review of the manuscript.

The authors would also like to thank Dr. Sunil Nair, Helena Redshaw, Jasmin Naim and Christine Andreasen (Chapman & Hall/CRC Press) for their friendly and efficient assistance during the final stage of the publishing process.

The work of the first author was sponsored partially by the Grant Agency of the Czech Republic under grants GP102/01/D114 and 102/01/0184, and partially by the TICAM Postdoctoral Fellowship Award. Several results from TICAM Reports No. 02-32 and No. 02-36 are included. The second and third authors acknowledge partial financial support of the Grant Agency of the Czech Republic under grants 201/01/1200 and 102/01/0184.

Our efforts could never have been successful without the understanding, patience and support of our families, for which we are deeply grateful.

Houston and Praha, March 2003 **P. Šolín, K. Segeth, I. Doležel**

Contents

List of Tables

List of Figures

Chapter 1

Introduction

As stated in the preface, we assume that the reader knows the basic concepts of the finite element method (see, e.g., [11, 39, 46, 47, 119, 178, 180, 191]). Nevertheless, let us review at least the most commonly used concepts – scalar and vector-valued Hilbert spaces, trivia of finite elements in these spaces, basic principles of the discretization of time-independent and evolutionary problems and a few additional topics that will be important for higher-order finite element technology.

1.1 Finite elements

DEFINITION 1.1 (Finite element) Finite element *in the sense of Ciarlet [47] is a triad $\mathcal{K} = (K, P, \Sigma)$, where*

- *K is a domain in $\mathbb{R}^d$ – we will confine ourselves to intervals ($d = 1$), triangles and quadrilaterals ($d = 2$), and tetrahedra, bricks and prisms ($d = 3$).*
- *P is a space of polynomials on K of dimension $dim(P) = N_P$.*
- *$\Sigma = \{L_1, L_2, \dots, L_{N_P}\}$ is a set of linear forms*

$$L_i : P \to \mathbb{R}, \quad i = 1, 2, \dots, N_P. \tag{1.1}$$

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

1.1.1 Function spaces H^1, $\boldsymbol{H}(\mathrm{curl})$ and $\boldsymbol{H}(\mathrm{div})$

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

$$H^1 = \{u \in L^2(\Omega); \partial u/\partial x_i \in L^2(\Omega),\ 1 \le i \le d\} \tag{1.2}$$

is the basic and most commonly used Sobolev space. Recall that the partial derivatives in (1.2) are understood in the sense of distributions. The Hilbert spaces

$$\boldsymbol{H}(\mathrm{curl}) = \{\boldsymbol{u} \in [L^2(\Omega)]^d; \mathbf{curl}\,\boldsymbol{u} \in [L^2(\Omega)]^d\} \tag{1.3}$$

and

$$\boldsymbol{H}(\mathrm{div}) = \{\boldsymbol{u} \in [L^2(\Omega)]^d; \mathrm{div}\,\boldsymbol{u} \in L^2(\Omega)\} \tag{1.4}$$

of vector-valued functions (defined for $d = 2, 3$) appear in variational formulations of problems rooted, e.g., in Maxwell's equations, mixed formulations in elasticity and acoustics.

Notice that the spaces $\boldsymbol{H}(\mathrm{curl})$ and $\boldsymbol{H}(\mathrm{div})$ fall between the spaces L^2 and H^1 in the sense that only some combinations of the partial derivatives need to be square-integrable.

1.1.2 Unisolvency of finite elements

Definition 1.2 introduces *unisolvency* as another expression for *compatibility* of the set of degrees of freedom Σ with the polynomial space P.

DEFINITION 1.2 (Unisolvency of finite elements) *The finite element $\mathcal{K} = (K, P, \Sigma)$ is said to be* unisolvent *if for every function $g \in P$ it holds*

$$L_1(g) = L_2(g) = \ldots = L_{N_P}(g) = 0 \ \Rightarrow \ g = 0. \tag{1.5}$$

In other words, every vector of numbers

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

uniquely identifies a polynomial g in the space P.

Definition 1.3 together with Theorem 1.1 offer a useful characterization of unisolvency of finite elements.

DEFINITION 1.3 (δ-property) *Let $\mathcal{K} = (K, P, \Sigma)$, dim$(P) = N_P$, be a finite element. We say that a set of functions $\mathcal{B} = \{\theta_1, \theta_2, \ldots, \theta_{N_P}\} \subset P$ has the δ-*property *if*

$$L_i(\theta_j) = \delta_{ij} \quad \textit{for all } 1 \le i, j \le N_P. \tag{1.6}$$

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

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

PROOF First assume that the finite element $\mathcal{K}$ is unisolvent. We take an arbitrary basis $\{g_1, g_2, \ldots, g_{N_P}\} \subset P$ and express each sought function θ_j, $j = 1, \ldots, N_P$, as

$$\theta_j = \sum_{k=1}^{N_P} a_{kj} g_k.$$

To satisfy the δ-property, we require that

$$L_i(\theta_j) = L_i\left(\sum_{k=1}^{N_P} a_{kj} g_k\right) = \sum_{k=1}^{N_P} a_{kj} L_i(g_k) = \delta_{ij}, \quad 1 \le i, j \le N_P, \qquad (1.7)$$

which yields a system of N_P linear equations for each j. Summarized, these linear systems give a matrix equation

$$\boldsymbol{L}\boldsymbol{A} = \boldsymbol{I},$$

where $\boldsymbol{L} = \{L_i(g_k)\}_{i,k=1}^{N_P}$, $\boldsymbol{I} = \{\delta_{ij}\}_{i,j=1}^{N_P}$ is the canonical matrix and the unknown matrix $\boldsymbol{A} = \{a_{kj}\}_{k,j=1}^{N_P}$ contains in its rows coefficients corresponding to the functions $\theta_1, \theta_2, \ldots, \theta_{N_P}$, respectively. Let us assume that the columns of $\boldsymbol{L}$ are linearly dependent, i.e., that there exists a nontrivial set of coefficients $\alpha_1, \alpha_2, \ldots, \alpha_{N_P}$ such that

$$\sum_{k=1}^{N_P} \alpha_k L_i(g_k) = L_i\left(\sum_{k=1}^{N_P} \alpha_k g_k\right) = 0 \quad \text{for all } i = 1, 2, \ldots, N_P. \qquad (1.8)$$

Since $\sum_{k=1}^{N_P} \alpha_k g_k$ is a nontrivial function, (1.8) is in contradiction with unisolvency of the element $\mathcal{K}$. Hence, the matrix $\boldsymbol{L}$ is invertible and the functions $\theta_1, \theta_2, \ldots, \theta_{N_P}$ are uniquely identified by the coefficients

$$\boldsymbol{A} = \boldsymbol{L}^{-1}\boldsymbol{I}.$$

It remains to be shown that the functions $\theta_1, \theta_2, \ldots, \theta_{N_P}$ are linearly independent, i.e., that

$$\sum_{j=1}^{N_P} \beta_j \theta_j = 0 \;\Rightarrow\; \beta_i = 0 \text{ for all } i = 1, 2, \ldots, N_P.$$

Obviously this is true since

$$0 = L_i\left(\sum_{j=1}^{N_P} \beta_j \theta_j\right) = \sum_{j=1}^{N_P} \beta_j L_i(\theta_j) = \beta_i$$

for all $i = 1, 2, \ldots, N$. Therefore the functions $\theta_1, \theta_2, \ldots, \theta_{N_P}$ constitute a basis in the space P, which by (1.7) has the δ-property.

The other implication is also easy to verify: Let $\mathcal{B} = \{\theta_1, \theta_2, \ldots, \theta_{N_P}\}$ be a basis of the space P satisfying the δ-property. Every function $g \in P$ can be expressed as

$$g = \sum_{j=1}^{N_P} \gamma_j \theta_j.$$

Assuming that

$$L_1(g) = L_2(g) = \ldots = L_{N_P}(g) = 0,$$

we immediately conclude that

$$0 = L_i(g) = L_i\left(\sum_{j=1}^{N_P} \gamma_j \theta_j\right) = \gamma_i \quad \text{for all } i = 1, 2, \ldots, N_P.$$

Hence necessarily $g = 0$ and the finite element is unisolvent. □

REMARK 1.1 (Checking unisolvency) The proof to Theorem 1.1 offers a simple procedure to check the unisolvency of a finite element: one considers an arbitrary basis of the polynomial space P and constructs the matrix $\boldsymbol{L}$ by applying the linear forms $L_1, L_2, \ldots, L_{N_P} \in \Sigma$ to the basis functions. If the matrix $\boldsymbol{L}$ is invertible, one knows that the element is unisolvent, and moreover $\boldsymbol{L}^{-1}$ yields the basis functions satisfying the δ-property. If the matrix $\boldsymbol{L}$ is not invertible, the element is not unisolvent. □

REMARK 1.2 (Nodal, hierarchic and dual basis) The expression *node* has in the finite element analysis several different meanings. To begin with, by *nodal* some authors denote the unique basis $\mathcal{B} \subset P$ that satisfies the δ-property (1.6). We will work with *nodal* and *hierarchic* bases of the space P which both satisfy the δ-property. Definitions will be given when appropriate. Existence and uniqueness of a basis $\mathcal{B} \subset P$ satisfying the δ-property is equivalent to the condition that the linear forms $L_1, \ldots, L_{N_P}$ form a *dual basis* to $\mathcal{B}$ in the space P' of linear forms over P. □

Example 1.1 (A nonunisolvent element)
Consider a square domain $K = (-1, 1)^2$, polynomial space

$$P = \text{span}\{1, x_1, x_2, x_1 x_2\}$$

and a set of degrees of freedom Σ consisting of linear forms $L_i : P \to \mathbb{R}$ associated with function values at points $[-1, 0]$, $[1, 0]$, $[0, -1]$ and $[0, 1]$, as shown in Figure 1.1.

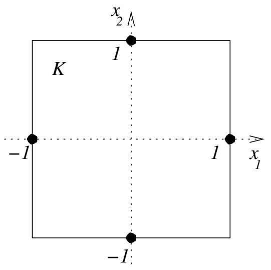

FIGURE 1.1: An example of a nonunisolvent finite element.

Hence, the degrees of freedom L_i are defined by

$$\begin{aligned} L_1(g) &= g(-1, 0), \\ L_2(g) &= g(1, 0), \\ L_3(g) &= g(0, -1), \\ L_4(g) &= g(0, 1), \end{aligned}$$

and the matrix $\boldsymbol{L}$ corresponding to the functions $1, x_1, x_2, x_1x_2$ has the form

$$\boldsymbol{L} = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}.$$

Obviously the matrix $\boldsymbol{L}$ is singular and therefore the finite element (K, P, Σ) is *not unisolvent*. It can be easily checked that the finite element becomes unisolvent after using the vertices instead of edge midpoints. $\square$

Example 1.2 (Unisolvent nodal elements)
Consider, for example, the interval $K_a = (-1, 1)$ and a space $P_a = P^p(K_a)$ of polynomials of the order at most p over K_a. Let us cover K_a with $N_P = p + 1$ points (geometrical nodes) $-1 = X_1 < X_2 < \ldots < X_{N_P} = 1$. These points have to be chosen carefully since their distribution determines the basis functions and consequently the conditioning of the discrete problem. Define the set of degrees of freedom $\Sigma_a = \{L_1, L_2, \ldots, L_{N_P}\}$, $L_i : P_a \to \mathbb{R}$ by

$$L_i(g) = g(X_i) \quad \text{for all } g \in P_a \quad \text{and} \quad i = 1, 2, \ldots, N_P. \tag{1.9}$$

Obviously the forms L_i are linear. Since every polynomial $g \in P_a$ satisfies

$$g(X_1) = g(X_2) = \ldots = g(X_{N_P}) = 0 \ \Rightarrow \ g = 0,$$

the finite element $\mathcal{K}_a = (K_a, P_a, \Sigma_a)$ is unisolvent. It is easy to construct a basis of the space P_a that satisfies the δ-property – in 1D one does not even have to solve systems of linear equations since the Lagrange interpolation polynomial can be exploited (see Section 1.3, formula 1.76).

Nodal elements in higher spatial dimensions are constructed in the same way. Consider, for instance, an equilateral triangle T, $\mathrm{diam}(T) = 2$ and a space $P_T = P^p(T)$ of polynomials of the order at most p over T. One may cover T (for example) with $N_P = (p+1)(p+2)/2$ Gauss-Lobatto points $X_1, X_2, \ldots, X_{N_P}$ as shown in Figure 1.2. Basis functions satisfying the δ-property are constructed by inverting the $N_P \times N_P$ matrix $\boldsymbol{L}$ as described in Remark 1.1.

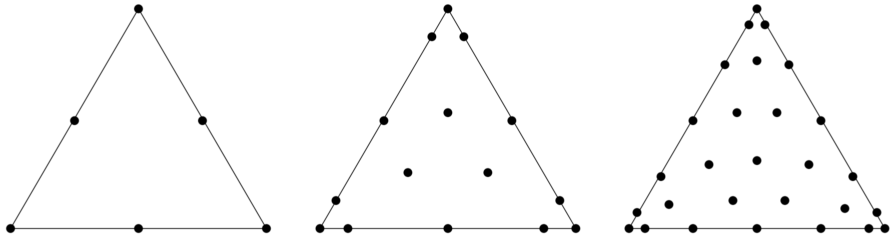

FIGURE 1.2: Nodal points in the equilateral triangle based on Gauss-Lobatto points for $p = 2$ (6 points), $p = 4$ (15 points) and $p = 6$ (28 points) with optimized interpolation properties. For the construction and properties of these geometrical nodes see, e.g., [109].

Finite elements based on nodal values are popular in connection with h-adaptive methods (see, e.g., [109, 111, 110, 112] and others). $\square$

Example 1.3 (Unisolvent hierarchic elements)
Application of hierarchic shape functions represents another major approach to the design of finite elements. Consider a domain K and a space P of polynomials of the order at most p of dimension N_P. Consider a *hierarchic basis* $\mathcal{B}^p = \{\theta_1, \theta_2, \ldots, \theta_{N_p}\}$ in the space P. By hierarchic we mean that

$$\mathcal{B}^p \subset \mathcal{B}^{p+1}$$

for every p. Every polynomial $g \in P$ can be uniquely expressed as a linear combination

$$g = \sum_{i=1}^{N_p} \beta_i \theta_i = \sum_{i=1}^{N_p} L_i(g)\theta_i, \tag{1.10}$$

where β_i are real coefficients and $L_i(g) = \beta_i$ linear forms

$$L_i : P \to \mathbb{R}, \quad i = 1, 2, \dots, N_p. \tag{1.11}$$

Obviously the choice $\Sigma = \{L_1, L_2, \dots, L_{N_P}\}$ yields a unisolvent finite element $(\mathcal{K}, P, \Sigma)$, and by definition the hierarchic basis $\mathcal{B}$ has the δ-property (1.6).

Hierarchic elements allow for locally nonuniform distribution of the order of polynomial approximation more easily than nodal elements, which makes them suitable for p- and hp-adaptivity. We will pursue the hierarchic approach in this book. $\square$

REMARK 1.3 (Selection of finite elements) The choice of a finite element (or their combination) depends upon the problem solved and upon the expectations that we put into the finite element scheme. Obviously it has a crucial effect on the behavior of the finite element scheme (see, for example, comparison of *conditioning properties* of various types of elements in Section 1.3). Several examples of standard as well as exotic finite elements were collected in [38]. $\square$

1.1.3 Finite element mesh

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

DEFINITION 1.4 (Finite element mesh) Finite element mesh $\mathcal{T}_{h,p} = \{K_1, K_2, \dots, K_M\}$ *over a domain* $\Omega_h \subset \mathbb{R}^d$ *with a piecewise-polynomial boundary is a geometrical division of* Ω_h *into a finite number of nonoverlapping (curved) open polygonal cells* K_i *such that*

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

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

DEFINITION 1.5 (Hybrid mesh) *If various types of cells are combined, the mesh is called* hybrid.

DEFINITION 1.6 (Regular mesh) *The mesh is called* regular *if for any two elements K_i and K_j, $i \neq j$ only one of the following alternatives holds:*

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

By assuming that the mesh is regular we avoid *hanging nodes*, the existence of which substantially complicates the discretization procedure. In this book we will use the word *node* as an abstraction for vertices, edges, faces and element interiors (the reasons for this soon become clear). Basically, hanging nodes can be grid vertices which lie in the interior of an edge or face of another cell, grid edges which lie in the interior of an edge or of a face of another cell, and grid faces which lie in the interior of a face of another cell. The constrained approximation technique in 2D and 3D will be discussed in more detail in Section 3.6. Various constellations involving hanging nodes are illustrated in Figure 1.3.

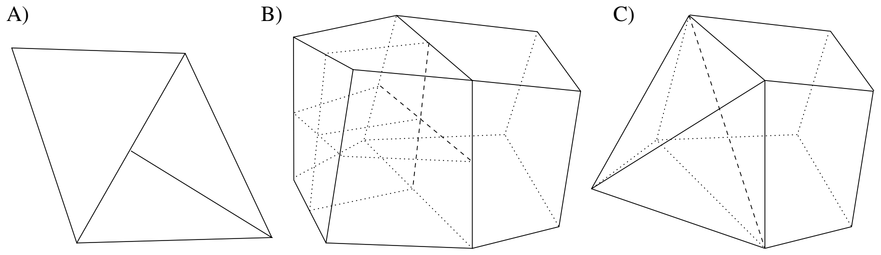

FIGURE 1.3: Examples of hanging nodes. A) Single hanging vertex and two edges in a 2D mesh. B) Five hanging vertices, twelve edges and four faces in a 3D mesh. C) Single hanging edge and two faces in a 3D hybrid mesh.

We will consider all of the most commonly used types of cells: one-dimensional elements, triangles and quadrilaterals in two spatial dimensions and tetrahedra, prisms and hexahedra in 3D.

1.1.4 Finite element interpolants and conformity

In Paragraph 1.1.2 we introduced *unisolvency* of the finite element (K, P, Σ) as another expression for the compatibility of the set of degrees of freedom Σ with the polynomial space P. Now we will address the compatibility of finite elements with spaces of functions where they will be used for approximation purposes – their *conformity* to these spaces.

The notion of conformity of finite elements to spaces of functions is tightly connected with their interpolation properties in these spaces.

Finite element interpolant

In the beginning let us assume that the degrees of freedom $L_1, L_2, \ldots, L_{N_P}$ of the finite element (K, P, Σ) are defined in a larger Hilbert space $V(K)$, $P \subset V(K)$.

DEFINITION 1.7 (Finite element interpolant) *Given a unisolvent finite element (K, P, Σ), let $\mathcal{B} = \{\theta_1, \theta_2, \ldots, \theta_{N_P}\}$ be the unique basis of the space P satisfying the δ-property (1.6). Let $v \in V$, where $P \subset V$, be a function for which all the linear forms $L_1, L_2, \ldots, L_{N_P}$ are defined. We define the (local)* interpolant *as*

$$\mathcal{I}_K(v) = \sum_{i=1}^{N_P} L_i(v)\theta_i. \tag{1.12}$$

It follows immediately from the linearity of the forms L_i that the interpolation operator $\mathcal{I}_K : V \to P$ is linear.

PROPOSITION 1.1
Let (K, P, Σ) be a unisolvent finite element and let $v \in V$, $P \subset V$ be a function for which all the linear forms $L_1, L_2, \ldots, L_{N_P}$ are defined. Then

$$L_i(\mathcal{I}_K(v)) = L_i(v), \quad 1 \le i \le N_P. \tag{1.13}$$

PROOF It follows immediately from Definition 1.7 and the δ-property (1.6) that

$$L_i\left(\sum_{j=1}^{N_P} L_j(v)\theta_j\right) = \sum_{j=1}^{N_P} L_j(v)L_i(\theta_j) = L_i(v).$$

$\square$

PROPOSITION 1.2
Let (K, P, Σ) be a unisolvent finite element. The finite element interpolation operator $\mathcal{I}_K$ is idempotent*,*

$$\mathcal{I}_K^2 = \mathcal{I}_K. \tag{1.14}$$

PROOF It follows immediately from Proposition 1.1 that

$$\mathcal{I}_K(v) = v, \quad v \in P. \tag{1.15}$$

Hence, for all $v \in V$, where $P \subset V$, it is

$$\mathcal{I}_K(\underbrace{\mathcal{I}_K(v)}_{\in P}) = \mathcal{I}_K(v)$$

which was to be shown. $\square$

Conformity of finite elements to spaces of functions

Let $V(\Omega_h)$ be a general Hilbert space of functions and let the computational domain Ω_h be covered with a finite element mesh $\mathcal{T}_{h,p}$. Global finite element interpolant $\mathcal{I}$ is defined elementwise by means of local element interpolants $\mathcal{I}_{K_i}$,

$$\mathcal{I}(v)|_{K_i} \equiv \mathcal{I}_{K_i} \quad \text{for all } i = 1, 2, \ldots, M. \tag{1.16}$$

It is natural to request that for all functions $v \in V$ also $\mathcal{I}(v) \in V$ whenever $\mathcal{I}(v)$ is defined.

Most commonly used definitions of conformity cover neither finite element meshes that comprise various types of finite elements nor hierarchic elements which are the main issue in this book. Therefore we will speak about the conformity of finite elements in the sense of conformity of *whole finite element meshes.* This notion naturally reduces to the standard *conformity of finite elements* if all the finite elements in the mesh are of the same type.

DEFINITION 1.8 (Conformity of finite elements) *Let $\mathcal{T}_{h,p}$ be a finite element mesh consisting of M unisolvent finite elements (K_i, P_i, Σ_i), $i = 1, 2, \ldots, M$. Let $V(\Omega_h)$ be a Hilbert space of functions and $\mathcal{I}_{K_i} : V(K_i) \to P_i$ the (local) finite element interpolation operators. We say that the finite element mesh $\mathcal{T}_{h,p}$ is conforming to the space V if and only if there exists a subspace $V^*(\Omega_h) \subset V(\Omega_h)$, which is dense in V (i.e., $\overline{V^*} = V$), such that for each function $v \in V^*(\Omega_h)$ the corresponding global interpolant $\mathcal{I}(v)$ is defined and lies in the space $V(\Omega_h)$.*

Global conformity requirements for the most commonly used Hilbert spaces H^1, $\boldsymbol{H}(\mathrm{curl})$ and $\boldsymbol{H}(\mathrm{div})$ are formulated in the following Lemmas 1.1 – 1.3 (see also, e.g., [143, 166, 159]).

LEMMA 1.1 (Conformity requirements of the space H^1)
Consider a domain $\Omega_h \subset \mathbb{R}^d$ covered with a finite element mesh $\mathcal{T}_{h,p}$. A function $v : \Omega_h \to \mathbb{R}$ belongs to $H^1(\Omega_h)$ if and only if

1. *$v|_K \in H^1(K)$ for each element $K \in \mathcal{T}_{h,p}$,*

2. *for each common face $f = \overline{K}_1 \cap \overline{K}_2$, $K_1, K_2 \in \mathcal{T}_{h,p}$ the trace of $v|_{K_1}$ and $v|_{K_2}$ on f is the same.*

PROOF Using *1.*, define the functions $w_j \in L^2(\Omega_h)$, $j = 1, 2, \ldots, d$ as

$$w_j|_K = D_j(v|_K)$$

for all $K \in \mathcal{T}_{h,p}$. We will show that $v \in H^1(\Omega_h)$ by simply verifying that $w_j = D_j v$.

Using the Green theorem we have for every $\varphi \in \mathcal{D}(\Omega_h)$

$$\int_{\Omega_h} w_j \varphi = \sum_{K \in \mathcal{T}_{h,p}} \int_K w_j \varphi = -\sum_K \int_K (v|_K) D_j \varphi + \sum_K \int_{\partial K} v|_K \varphi \nu_{K,j},$$

where ν_K is the outward normal vector to K on ∂K. The symbol $\mathcal{D}(\Omega_h)$ stands for the space of *distributions* over Ω_h, where distributions are infinitely smooth functions $\varphi : \mathbb{R}^d \to \mathbb{R}$ whose support lies within the domain Ω_h. Since φ is vanishing on $\partial\Omega_h$ and $\nu_{K_1} = -\nu_{K_2} = \nu$ on the common face f, we have by *2.*

$$\begin{aligned}\int_{\Omega_h} w_j \varphi &= -\int_{\Omega_h} v D_j \varphi + \sum_{f, f = \overline{K}_1 \cap \overline{K}_2, K_1, K_2 \in \mathcal{T}_{h,p}} \int_f (v|_{K_1} - v|_{K_2}) \varphi \nu_j \\ &= -\int_{\Omega_h} v D_j \varphi,\end{aligned}$$

and thus $w_j = D_j v$.

Conversely, if we assume that $v \in H^1(\Omega_h)$, it follows at once that *1.* holds. Using further $w_j = D_j v$, in the same way as before we obtain that

$$\sum_{f, f = \overline{K}_1 \cap \overline{K}_2, K_1, K_2 \in \mathcal{T}_{h,p}} \int_f (v|_{K_1} - v|_{K_2}) \varphi \nu_j = 0$$

for all $\varphi \in \mathcal{D}(\Omega_h)$, $j = 1, 2, \ldots, d$. Hence, *2.* is satisfied. $\square$

LEMMA 1.2 (Conformity requirements of the space $\boldsymbol{H}(\mathrm{div})$)
Consider a domain $\Omega_h \subset \mathbb{R}^d$ covered with a finite element mesh $\mathcal{T}_{h,p}$. Consider a function $\boldsymbol{v} : \Omega_h \to \mathbb{R}^d$ such that

1. *$\boldsymbol{v}|_K \in [H^1(K)]^d$ for each element $K \in \mathcal{T}_{h,p}$,*
2. *for each common face $f = \overline{K}_1 \cap \overline{K}_2$, $K_1, K_2 \in \mathcal{T}_{h,p}$ the trace of the normal component $\boldsymbol{n} \cdot \boldsymbol{v}|_{K_1}$ and $\boldsymbol{n} \cdot \boldsymbol{v}|_{K_2}$ on f is the same (here $\boldsymbol{n}$ is a unique normal vector to the face f).*

Then $\boldsymbol{v} \in \boldsymbol{H}(\mathrm{div})$. On the other hand, if $\boldsymbol{v} \in \boldsymbol{H}(\mathrm{div})$ and 1. holds, then 2. is satisfied.

PROOF Define $w \in L^2(\Omega_h)$ by

$$w|_K = \mathrm{div}(\boldsymbol{v}|_K)$$

for all $K \in \mathcal{T}_{h,p}$. The Green formula implies for every $\varphi \in \mathcal{D}$

$$\begin{aligned}(\mathrm{div}\boldsymbol{v}, \varphi) &= -\int_{\Omega_h} \boldsymbol{v} \cdot \nabla\varphi = -\sum_{K \in \mathcal{T}_{h,p}} \int_K (\boldsymbol{v}|_K) \cdot \nabla\varphi \\ &= \sum_{K \in \mathcal{T}_{h,p}} \int_K \mathrm{div}(\boldsymbol{v}|_K)\varphi \\ &\quad - \sum_{f, f=\overline{K}_1 \cap \overline{K}_2,\ \ K_1, K_2 \in \mathcal{T}_{h,p}} \int_f (\boldsymbol{n} \cdot \boldsymbol{v}|_{K_1} - \boldsymbol{n} \cdot \boldsymbol{v}|_{K_2})\varphi = \int_{\Omega_h} w\varphi\end{aligned}$$

and therefore $\mathrm{div}\boldsymbol{v} = w$.

Conversely, if $\boldsymbol{v} \in \boldsymbol{H}(\mathrm{div})$, we have $w = \mathrm{div}\boldsymbol{v}$. Since $\boldsymbol{v}|_K \in [H^1(\Omega_h)]^d$, the trace on f is well defined and we obtain

$$\sum_{f, f=\overline{K}_1 \cap \overline{K}_2, K_1, K_2 \in \mathcal{T}_{h,p}} \int_f (\boldsymbol{n} \cdot \boldsymbol{v}|_{K_1} - \boldsymbol{n} \cdot \boldsymbol{v}|_{K_2})\varphi = 0$$

for all $\varphi \in \mathcal{D}$. Hence *1.* holds. $\square$

LEMMA 1.3 (Conformity requirements of the space $\boldsymbol{H}(\mathrm{curl})$)
Consider a domain $\Omega_h \subset \mathbb{R}^d$ covered with a finite element mesh $\mathcal{T}_{h,p}$. Consider a function $\boldsymbol{v} : \Omega_h \to \mathbb{R}^d$ such that

1. *$\boldsymbol{v}|_K \in [H^1(K)]^d$ for each element $K \in \mathcal{T}_{h,p}$,*
2. *for each common face $f = \overline{K}_1 \cap \overline{K}_2$, $K_1, K_2 \in \mathcal{T}_{h,p}$ the trace of the tangential component $\boldsymbol{n} \times \boldsymbol{v}|_{K_1}$ and $\boldsymbol{n} \times \boldsymbol{v}|_{K_2}$ on f is the same (again, $\boldsymbol{n}$ is a unique normal vector to the face f).*

Then $\boldsymbol{v} \in \boldsymbol{H}(\mathrm{curl})$. On the other hand, if $\boldsymbol{v} \in \boldsymbol{H}(\mathrm{curl})$ and 1. holds, then 2. is satisfied.

PROOF Similar to the previous case. $\square$

REMARK 1.4 Although the conditions declared in Lemmas 1.2 and 1.3 are weaker than those associated with the space H^1, their algorithmic realization is more demanding. More about $\boldsymbol{H}(\mathrm{curl})$- and $\boldsymbol{H}(\mathrm{div})$-conforming approximations will follow. $\square$

REMARK 1.5 (Conformity requirements of the space L^2) There are *no continuity requirements* on interelement boundaries in L^2-conforming approximations. See, e.g., [143, 166, 159] for additional conformity properties of finite elements. $\square$

REMARK 1.6 (Interpolation on hierarchic elements) Notice that Definition 1.7 says nothing about the interpolation on hierarchic elements since the degrees of freedom L_i (coefficients β_i in (1.10)) are undefined when $v \notin P$. Definition of local interpolation operators on hierarchic elements requires deeper mathematical analysis. Since the standard Lagrange interpolation has to be combined with *projection* onto hierarchically constructed subspaces of the space P, sometimes the technique is called *projection-based interpolation.* We find it appropriate to postpone the discussion of this issue to Chapter 3 where relevant machinery will be in place. The reader does not need to worry in the meantime since, of course, the projection-based interpolation operators will be compatible with the global conformity requirements presented in Lemmas 1.1 – 1.3. $\square$

Example 1.4 (A conforming and a nonconforming element)
We find it useful to present a very simple example where all ideas presented in this paragraph can be fixed. Let us consider linear triangular finite elements of the (i) Lagrange and (ii) Crouzeix-Raviart type (for the latter see [56]). Both of them are defined on a triangular domain K, using the space of linear polynomials $P(K)$. The degrees of freedom $L_1, L_2, \ldots, L_{N_P}$ are associated with (i) element vertices $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ and (ii) midpoints $\boldsymbol{d}, \boldsymbol{e}, \boldsymbol{f}$ of the edges (as shown in Figure 1.4).

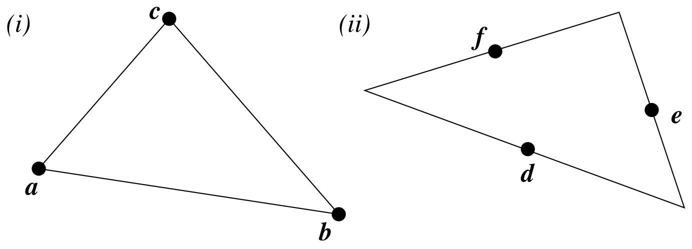

FIGURE 1.4: Nodal points for (i) linear Lagrange and (ii) Crouzeix-Raviart elements.

In both cases $\dim(P) = 3$. For the Lagrange element the degrees of freedom $L_i : P \to \mathbb{R}$ are defined as

$$L_1(g) = g(\boldsymbol{a}),\ L_2(g) = g(\boldsymbol{b}),\ L_3(g) = g(\boldsymbol{c}),$$

and unisolvency check (see Remark 1.1) yields a matrix $\boldsymbol{L}$ in the form

$$\boldsymbol{L}^{Lagr} = \begin{pmatrix} 1 & a_1 & a_2 \\ 1 & b_1 & b_2 \\ 1 & c_1 & c_2 \end{pmatrix}. \tag{1.17}$$

Analogously, in the Crouzeix-Raviart case we have the linear forms

$$L_1(g) = g(\boldsymbol{d}),\ L_2(g) = g(\boldsymbol{e}),\ L_3(g) = g(\boldsymbol{f}) \tag{1.18}$$

and

$$\boldsymbol{L}^{C-R} = \begin{pmatrix} 1 & d_1 & d_2 \\ 1 & e_1 & e_2 \\ 1 & f_1 & f_2 \end{pmatrix}. \tag{1.19}$$

Obviously both the matrices $\boldsymbol{L}^{Lagr}, \boldsymbol{L}^{C-R}$ are regular unless the triangle K is degenerated (and this is forbidden since K is a domain in $\mathbb{R}^2$).

Consider now a finite element mesh $\mathcal{T}_{h,p} = \{K_1, K_2\}$ consisting of two elements as illustrated in Figure 1.5.

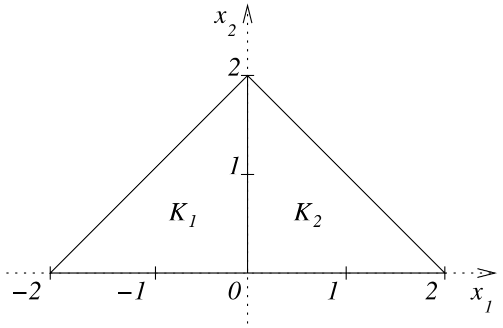

FIGURE 1.5: Sample mesh consisting of two triangular elements.

In the Lagrange case, the corresponding two sets of basis functions that satisfy the δ-property (1.6) are obtained after inverting the matrices $\boldsymbol{L}^{Lagr}_{K_1}$ and $\boldsymbol{L}^{Lagr}_{K_2}$ (see Remark 1.1). We obtain

$$\theta^{(1)}_1 = -x_1/2,\ \ \theta^{(1)}_2 = (x_1 - x_2 + 2)/2,\ \ \theta^{(1)}_3 = x_2/2,$$
$$\theta^{(2)}_1 = (2 - x_1 - x_2)/2,\ \ \theta^{(2)}_2 = x_1/2,\ \ \theta^{(2)}_3 = x_2/2.$$

Hence the interpolation operators $\mathcal{I}^{Lagr}_{K_1} : V(K_1) \to P(K_1)$ and $\mathcal{I}^{Lagr}_{K_2} : V(K_2) \to P(K_2)$ (Definition 1.7 with $V = H^1$ and $V^* = C$) have the form

$$\mathcal{I}^{Lagr}_{K_1}(v) = v(-2,0)\theta^{(1)}_1 + v(0,0)\theta^{(1)}_2 + v(0,2)\theta^{(1)}_3,$$
$$\mathcal{I}^{Lagr}_{K_2}(v) = v(0,0)\theta^{(2)}_1 + v(2,0)\theta^{(2)}_2 + v(0,2)\theta^{(2)}_3.$$

Analogously in the Crouzeix-Raviart case the inversion of the matrices $\boldsymbol{L}_{K_1}^{C-R}$ and $\boldsymbol{L}_{K_2}^{C-R}$ yields the nodal bases

$$\theta_1^{(1)} = 1 - x_2, \quad \theta_2^{(1)} = x_1 + 1, \quad \theta_3^{(1)} = -(x_1 - x_2 + 1),$$
$$\theta_1^{(2)} = 1 - x_2, \quad \theta_2^{(2)} = x_1 + x_2 - 1, \quad \theta_3^{(2)} = 1 - x_1$$

and the local interpolation operators

$$\mathcal{I}_{K_1}^{C-R}(v) = v(-1,0)\theta_1^{(1)} + v(0,1)\theta_2^{(1)} + v(-1,1)\theta_3^{(1)},$$
$$\mathcal{I}_{K_2}^{C-R}(v) = v(1,0)\theta_1^{(2)} + v(1,1)\theta_2^{(2)} + v(0,1)\theta_3^{(2)}.$$

Consider now a function

$$v(x_1, x_2) = (x_1 - x_2)^2 \in C(\overline{K}_1 \cup \overline{K}_2).$$

The pair of the corresponding piecewise-linear global interpolants are depicted in Figures 1.6 and 1.7.

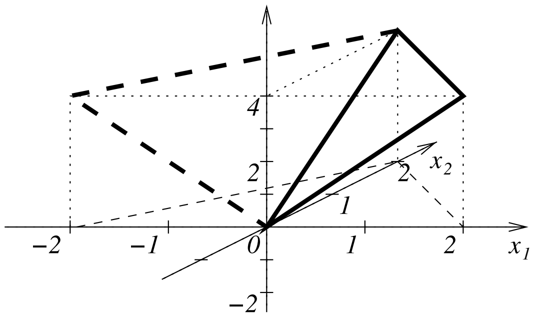

FIGURE 1.6: Continuous piecewise-linear Lagr. interpolant on $\overline{K}_1 \cup \overline{K}_2$.

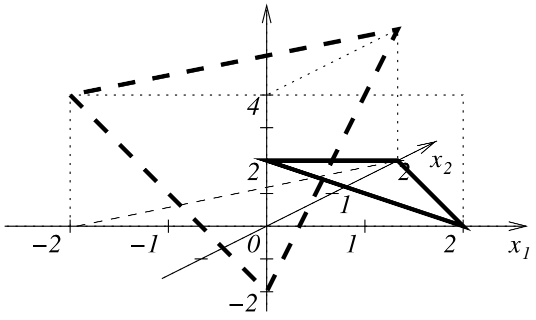

FIGURE 1.7: Discontinuous piecewise-linear Crouzeix-Raviart interpolant on $\overline{K}_1 \cup \overline{K}_2$. This figure illustrates that the finite element of Crouzeix-Raviart does not conform to the space H^1.

Obviously one example is not enough to prove *conformity* – one has to consider each pair of adjacent elements $K_i, K_j \in \mathcal{T}_{h,p}$ and show that the interpolant of all functions $v \in V(\overline{K}_i \cup \overline{K}_j)$ still lies in the space $V(\overline{K}_i \cup \overline{K}_j)$, using global conformity requirements of the space V (see Lemmas 1.1 – 1.3). In the linear Lagrange case, the coincidence of values of the function v at the pair of shared vertices together with the linearity of the local interpolant on element edges imply the continuity along the whole shared edge of K_i, K_j. Hence,

$$v \in V(\overline{K}_1 \cup \overline{K}_2) \ \Rightarrow\ \mathcal{I}^{Lagr}(v) \in V(\overline{K}_1 \cup \overline{K}_2)$$

holds for all $v \in H^1(\overline{K}_1 \cup \overline{K}_2)$, therefore the linear Lagrange finite element is conforming to the space $H^1(\overline{K}_1 \cup \overline{K}_2)$. $\square$

REMARK 1.7 (Nonconforming elements) For selected types of problems, nonconforming finite elements are used with excellent results. We refer to [45, 56, 90, 106, 121, 124, 125, 165] to mention at least a couple of examples. In this book we confine ourselves to conforming finite element approximations only. $\square$

REMARK 1.8 (Interpolation error estimates) At this point the next logical step would be to introduce local *element interpolation error estimates*. However, for all standard types of nodal elements this can be found in standard finite element textbooks. Relevant for our purposes are *projection-based interpolation* error estimates for *hierarchic elements* in spaces H^1, $\boldsymbol{H}(\mathrm{curl})$ and $\boldsymbol{H}(\mathrm{div})$. We will address them in Chapter 3. $\square$

1.1.5 Reference domains and reference maps

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

The situation changes dramatically with higher-order finite elements since they use a large amount of overlapping information, whose efficient management requires more structure to be imposed. For example, while first-order numerical quadrature can be implemented using coordinates of grid vertices only, higher-order quadrature schemes require many integration points per element (the actual amount depends on the order of accuracy, and in 2D and 3D it easily achieves several hundred). Both storing these values (d spatial coordinates and a weight per point) in all elements as well as reconstructing them periodically from a reference configuration would be extremely inefficient. The situation is analogous for higher-order basis functions.

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

$$\boldsymbol{x}_{K_i} : \hat{K} \to K_i. \tag{1.20}$$

The maps $\boldsymbol{x}_K$, together with polynomial spaces on the reference domain $\hat{K}$, will be used for the definition of the space of functions $V_{h,p}(\Omega_h)$ where the finite element solution will be sought. An example of a reference map in the quadrilateral case in 2D is illustrated in Figure 1.8.

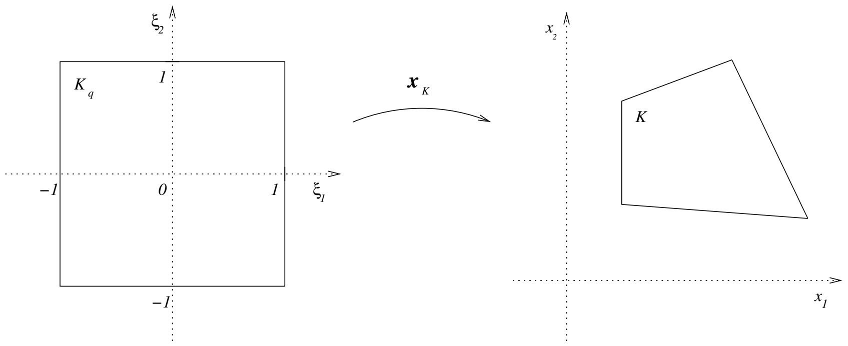

FIGURE 1.8: Reference map for a quadrilateral element.

The design of reference maps for all standard types of reference domains will be discussed in detail in Chapter 3. There we also show how one uses them to transfer the integrals over elements K_i from the variational formulation to the reference domains. The higher-order finite element discretization is performed almost exclusively on the reference domains.

1.1.6 Finite element discretization

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

$$L(u^* + \bar{u}, v) = f(v) \quad \text{for all } v \in V. \tag{1.21}$$

Both the forms L and f are assumed linear in v. If nonhomogeneous Dirichlet conditions are present, the solution $u = u^* + \bar{u}$ is sought in an affine function space which is different from V (functions satisfying nonhomogeneous

Dirichlet boundary conditions obviously cannot constitute any linear function space). The *lift function* u^* is chosen to satisfy nonhomogeneous Dirichlet conditions. Only the unknown component $\bar{u}$ satisfying *homogeneous* Dirichlet boundary conditions is sought. All functions $v \in V$ vanish on any Dirichlet part of the boundary $\partial\Omega$.

Neumann and Newton (Robin) boundary conditions are enforced by substituting them directly into boundary integrals in (1.21) over the corresponding part of the boundary $\partial\Omega$. See any basic finite element textbook for more details. An example of higher-order finite element discretization in 1D involving nonhomogeneous Dirichlet boundary conditions will be given in Section 1.3.

Approximation of weak forms and discretization

The finite element discretization of (1.21) is done in the following standard steps:

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

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

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

REMARK 1.9 (Variational crimes) Let us remark that in reality usually $V_{h,p} \not\subset V$ since the domains Ω and Ω_h differ, and moreover often even $\Omega_h \not\subset \Omega$. Hence this step is sometimes classified as a *variational crime* in the FE community. $\square$

Step 4: Approximate the form L by another form $L_{h,p}$, replacing the exact integration over Ω and $\partial\Omega$ by numerical integration over Ω_h and $\partial\Omega_h$. Notice that boundary conditions are shifted from $\partial\Omega$ to $\partial\Omega_h$ in this step. The question of an optimal choice of quadrature schemes will be addressed in Chapter 4.

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

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

$$L_{h,p}(u^*_{h,p} + \bar{u}_{h,p}, v_{h,p}) = f_{h,p}(v_{h,p}), \tag{1.22}$$

for all $v_{h,p} \in V_{h,p}$. The function $u^*_{h,p}$ is a suitable piecewise polynomial (usually a simple piecewise-linear) approximation of the lift function u^* from (1.21).

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

$$u_{h,p}(\boldsymbol{x}) = u^*_{h,p}(\boldsymbol{x}) + \bar{u}_{h,p}(\boldsymbol{x}) = u^*_{h,p}(\boldsymbol{x}) + \sum_{j=1}^{N} y_j v_j(\boldsymbol{x}). \tag{1.23}$$

Step 8: Insert the construction (1.23) into the approximate weak form (1.22) and select $v_{h,p} := v_i$, $i = 1, 2, \ldots, N$. This turns (1.22) into a system of algebraic equations

$$L_{h,p}\left(u^*_{h,p} + \sum_{j=1}^{N} y_j v_j, v_i\right) = f_{h,p}(v_i), \quad i = 1, 2, ..., N. \tag{1.24}$$

If the form $L_{h,p}$ is bilinear, (1.24) represents a system of linear algebraic equations which can be written in a matrix form $\boldsymbol{SY} = \boldsymbol{F}$,

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

Otherwise the algebraic system (1.24) is nonlinear.

Step 9: Solve the system (1.24) for the unknown coefficients $\boldsymbol{Y}$ of $\bar{u}_{h,p}$ with a suitable numerical scheme. Retrieve the approximate solution $u_{h,p}$ using (1.23). Numerical methods for the treatment of discrete problems will be discussed in Chapter 5.

1.1.7 Method of lines for evolutionary problems

The method of lines (MOL) is one of the most popular tools for the solution of evolutionary PDEs. The basic idea is to perform the discretization in space only while keeping the time-variable continuous. This is achieved by expressing the approximate solution $u_{h,p}(\boldsymbol{x}, t)$ in a form analogous to (1.23), *with time-dependent coefficients* $y_i = y_i(t)$:

$$u_{h,p}(\boldsymbol{x}, t) = u^*_{h,p}(\boldsymbol{x}) + \bar{u}_{h,p}(x, t) = u^*_{h,p}(\boldsymbol{x}) + \sum_{j=1}^{N} y_j(t) v_j(\boldsymbol{x}). \tag{1.25}$$

Thus, instead of a system of algebraic equations (1.24) we end up with a system of *ordinary differential equations.*

For details on the MOL itself as well as for error estimates for evolutionary problems solved by this method see, e.g., [13, 16, 78, 79, 80, 144, 155, 174, 175, 182, 194, 201, 198] and others.

1.2 Orthogonal polynomials

Orthogonal polynomials find applications in diverse fields of mathematics, both for theoretical and numerical issues. In our case they will play an essential role in the design of optimal higher-order shape functions. For additional information on orthogonal polynomials see, e.g., [192], which is usually referred to as a basic textbook on this subject.

1.2.1 The family of Jacobi polynomials

The class of Jacobi polynomials,

$$P_{n,\alpha,\beta}(x) = \frac{(-1)^n}{2^n n!}(1-x)^{-\alpha}(1+x)^{-\beta}\frac{\mathrm{d}^n}{\mathrm{d}x^n}\left[(1-x)^{\alpha+n}(1+x)^{\beta+n}\right], \quad (1.26)$$

holds the prominent position among orthogonal polynomials. It satisfies the Jacobi differential equation

$$(1-x^2)\frac{\mathrm{d}^2}{\mathrm{d}x^2}P_{n,\alpha,\beta}+(\beta-\alpha-(\beta+\alpha+2)x)\frac{\mathrm{d}}{\mathrm{d}x}P_{n,\alpha,\beta}+n(n+\alpha+\beta+1)P_{n,\alpha,\beta}=0 \quad (1.27)$$

($\alpha, \beta > -1$ are real parameters). Let $L^2_{\alpha,\beta}(I)$, where $I = (-1,1)$, denote the space of all functions which are square integrable in I with the weight

$$w_{\alpha,\beta}(x) = (1-x)^\alpha(1+x)^\beta \quad (1.28)$$

and with the corresponding norm

$$\|u\|^2_{L^2_{\alpha,\beta}} = \int_{-1}^{1} |u|^2 w_{\alpha,\beta}\mathrm{d}x. \quad (1.29)$$

Then every $u \in L^2_{\alpha,\beta}(I)$ can be expanded into the series

$$u(x) = \sum_{n=0}^{\infty} c_n P_{n,\alpha,\beta}(x) \quad \text{satisfying} \quad \lim_{k\to\infty} \|u - \sum_{n=0}^{k} c_n P_{n,\alpha,\beta}\|_{L^2_{\alpha,\beta}(I)} = 0. \quad (1.30)$$

Orthogonality of the Jacobi polynomials is exactly specified by

$$\int_{-1}^{1} P_{n,\alpha,\beta} P_{m,\alpha,\beta} (1-x)^\alpha (1+x)^\beta \mathrm{d}x = \begin{cases} e_{n,\alpha,\beta} \text{ for } n = m, \\ \\ \quad 0 \ \ \text{otherwise} \end{cases} \tag{1.31}$$

where

$$e_{n,\alpha,\beta} = \frac{2^{\alpha+\beta+1}}{2n+\alpha+\beta+1} \frac{\Gamma(\alpha+n+1)\Gamma(\beta+n+1)}{\Gamma(n+1)\Gamma(\alpha+\beta+n+1)}. \tag{1.32}$$

Here Γ is the standard Γ-function. The coefficients c_n are computed using the relation

$$c_n = \frac{1}{e_{n,\alpha,\beta}} \int_{-1}^{1} (1-x)^\alpha (1+x)^\beta P_{n,\alpha,\beta}(x) u(x) \, \mathrm{d}x. \tag{1.33}$$

We have the relation

$$\frac{\mathrm{d}^k}{\mathrm{d}x^k} P_{n,\alpha,\beta}(x) = 2^{-k} \frac{\Gamma(n+k+\alpha+\beta+1)}{\Gamma(n+\alpha+\beta+1)} P_{n-k,\alpha+k,\beta+k}. \tag{1.34}$$

Ultraspherical polynomials

The ultraspherical polynomials are a special case of the Jacobi polynomials, defined by

$$U_{n,\alpha} = P_{n,\alpha,\alpha}, \quad n = 0, 1, 2. \dots \tag{1.35}$$

They inherit all basic properties from the Jacobi polynomials (1.26).

Gegenbauer polynomials

Putting $\alpha = \beta = \nu - 1/2$ in (1.26), the Jacobi polynomials come over to the Gegenbauer polynomials

$$G_n^\nu(x) = \frac{\Gamma(n+2\nu)\Gamma(\nu+1/2)}{\Gamma(2\nu)\Gamma(n+\nu+1/2)} P_{n,\nu-1/2,\nu-1/2}(x), \tag{1.36}$$

which again inherit all basic properties of the Jacobi polynomials.

Chebyshev polynomials

The Chebyshev polynomials are another special case of the Jacobi polynomials (1.26) and inherit all of their basic properties. Putting $\alpha = \beta = -1/2$ we obtain

$$C_n(x) = \frac{2^{2n}(n!)^2}{(2n!)} P_{n,-1/2,-1/2}(x). \tag{1.37}$$

1.2.2 Legendre polynomials

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

$$L_n(x) = P_{n,0,0}(x). \tag{1.38}$$

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

$$(1-x^2)\frac{\mathrm{d}^2y}{\mathrm{d}x^2} - 2x\frac{\mathrm{d}y}{\mathrm{d}x} + k(k+1)y = 0. \tag{1.39}$$

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

$$\begin{aligned} L_0(x) &= 1, \\ L_1(x) &= x, \\ L_k(x) &= \frac{2k-1}{k}xL_{k-1}(x) - \frac{k-1}{k}L_{k-2}(x), \quad k = 2,3,\ldots, \end{aligned} \tag{1.40}$$

but they can be defined also by the differential relation

$$L_k(x) = \frac{1}{2^k k!}\frac{\mathrm{d}^k}{\mathrm{d}x^k}\left(x^2-1\right)^k, \quad \text{for } k = 0,1,2,\ldots. \tag{1.41}$$

Their orthogonality is exactly specified by

$$\int_{-1}^{1} L_k(x)L_m(x)\mathrm{d}x = \begin{cases} \dfrac{2}{2k+1} & \text{for } k = m, \\ 0 & \text{otherwise.} \end{cases} \tag{1.42}$$

Each consequent triad of Legendre polynomials obeys the relation

$$L_n(x) = \left(\frac{\mathrm{d}}{\mathrm{d}x}L_{n+1}(x) - \frac{\mathrm{d}}{\mathrm{d}x}L_{n-1}(x)\right), \quad n \geq 1, \tag{1.43}$$

and all of them satisfy

$$L_n(1) = 1, \ L_n(-1) = (-1)^n, \quad n \geq 0. \tag{1.44}$$

The Legendre expansion of a function $u \in L^2(I)$ has the form

$$u(x) = \sum_{n=0}^{\infty} c_n L_n(x), \tag{1.45}$$

which is understood as

$$\lim_{k\to\infty} \|u - \sum_{n=0}^{k} c_n L_n(x)\|_{L^2(I)} = 0. \tag{1.46}$$

The coefficients c_n are computed using the relation

$$c_n = \frac{2n+1}{2} \int_{-1}^{1} u(x) L_n(x). \tag{1.47}$$

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

$$\begin{aligned}
L_0(x) &= 1, \\
L_1(x) &= x, \\
L_2(x) &= \frac{3}{2}x^2 - \frac{1}{2}, \\
L_3(x) &= \frac{1}{2}x(5x^2 - 3), \\
L_4(x) &= \frac{1}{8}(35x^4 - 30x^2 + 3), \\
L_5(x) &= \frac{1}{8}x(63x^4 - 70x^2 + 15), \\
L_6(x) &= \frac{1}{16}(231x^6 - 315x^4 + 105x^2 - 5), \\
L_7(x) &= \frac{1}{16}x(429x^6 - 693x^4 + 315x^2 - 35), \\
L_8(x) &= \frac{1}{128}(6435x^8 - 12012x^6 + 6930x^4 - 1260x^2 + 35), \\
L_9(x) &= \frac{1}{128}x(12155x^8 - 25740x^6 + 18018x^4 - 4620x^2 + 315), \\
L_{10}(x) &= \frac{1}{256}(46189x^{10} - 109395x^8 + 90090x^6 - 30030x^4 + 3465x^2 - 63).
\end{aligned} \tag{1.48}$$

The functions $L_0, L_1, \ldots, L_9$ are illustrated in Figures 1.9 – 1.13.

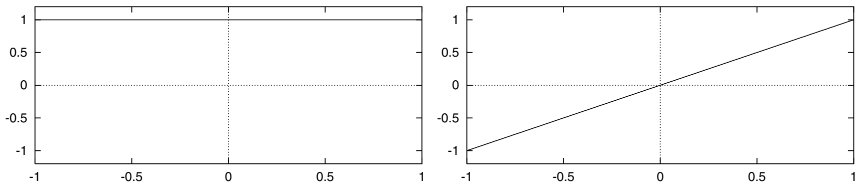

FIGURE 1.9: Legendre polynomials L_0, L_1.

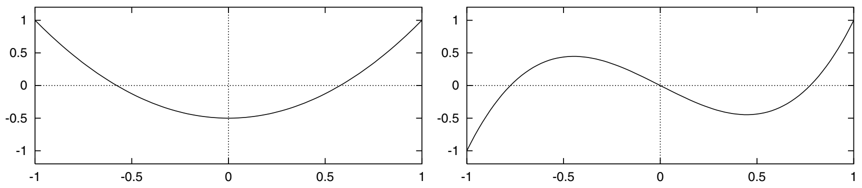

FIGURE 1.10: Legendre polynomials L_2, L_3.

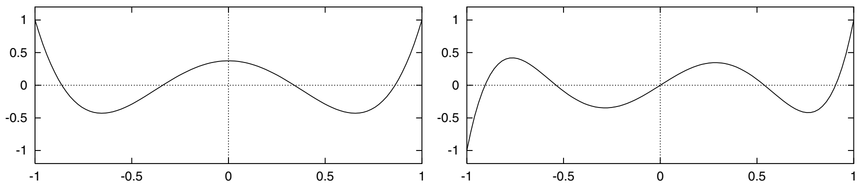

FIGURE 1.11: Legendre polynomials L_4, L_5.

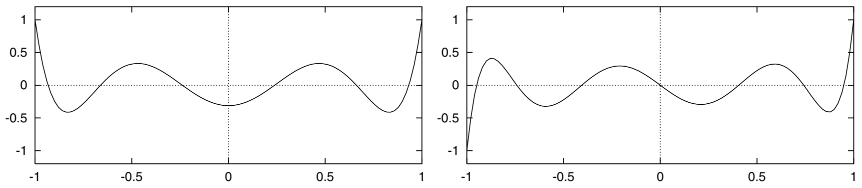

FIGURE 1.12: Legendre polynomials L_6, L_7.

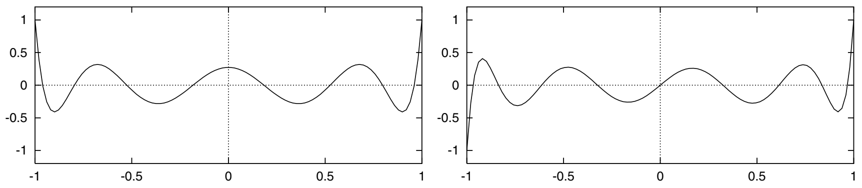

FIGURE 1.13: Legendre polynomials L_8, L_9.

1.2.3 Lobatto shape functions

Let us define functions

$$\begin{aligned} l_0(x) &= \frac{1-x}{2}, \quad l_1(x) = \frac{x+1}{2}, \\ l_k(x) &= \frac{1}{\|L_{k-1}\|_2} \int_{-1}^{x} L_{k-1}(\xi)\,\mathrm{d}\xi, \quad 2 \le k, \end{aligned} \tag{1.49}$$

where $\|L_{k-1}\|_2 = \sqrt{2/(2k-1)}$ from (1.42). Obviously $l_k(-1) = 0$, $k = 2, 3, \ldots$. It follows from the orthogonality of higher-order Legendre polynomials L_k to $L_0 \equiv 1$,

$$\int_{-1}^{1} L_k(x)\,\mathrm{d}x = 0, \quad k \ge 1, \tag{1.50}$$

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

$$\begin{aligned} l_2(x) &= \frac{1}{2}\sqrt{\frac{3}{2}}(x^2-1), \\ l_3(x) &= \frac{1}{2}\sqrt{\frac{5}{2}}(x^2-1)x, \\ l_4(x) &= \frac{1}{8}\sqrt{\frac{7}{2}}(x^2-1)(5x^2-1), \\ l_5(x) &= \frac{1}{8}\sqrt{\frac{9}{2}}(x^2-1)(7x^2-3)x, \\ l_6(x) &= \frac{1}{16}\sqrt{\frac{11}{2}}(x^2-1)(21x^4-14x^2+1), \\ l_7(x) &= \frac{1}{16}\sqrt{\frac{13}{2}}(x^2-1)(33x^4-30x^2+5)x, \\ l_8(x) &= \frac{1}{128}\sqrt{\frac{15}{2}}(x^2-1)(429x^6-495x^4+135x^2-5), \\ l_9(x) &= \frac{1}{128}\sqrt{\frac{17}{2}}(x^2-1)(715x^6-1001x^4+385x^2-35)x, \\ l_{10}(x) &= \frac{1}{256}\sqrt{\frac{19}{2}}(x^2-1)(2431x^8-4004x^6+2002x^4-308x^2+7). \end{aligned} \tag{1.51}$$

The Lobatto shape functions will play an essential role in the design of hierarchic shape functions in Chapter 2. Some of them are illustrated in Figures 1.14 – 1.18 (notice the different scales).

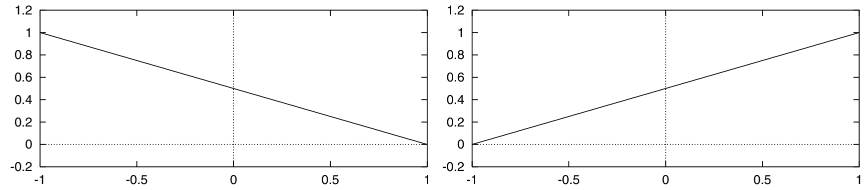

FIGURE 1.14: Lobatto shape functions l_0, l_1.

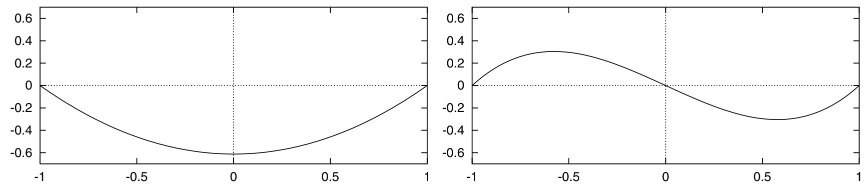

FIGURE 1.15: Lobatto shape functions l_2, l_3.

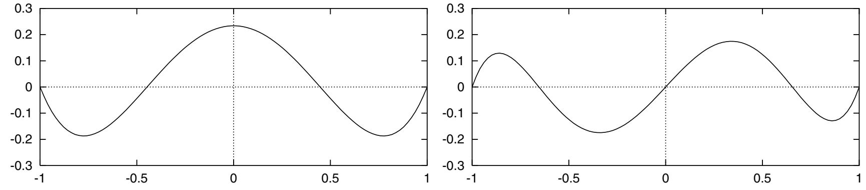

FIGURE 1.16: Lobatto shape functions l_4, l_5.

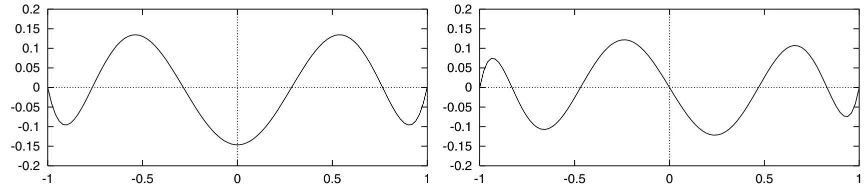

FIGURE 1.17: Lobatto shape functions l_6, l_7.

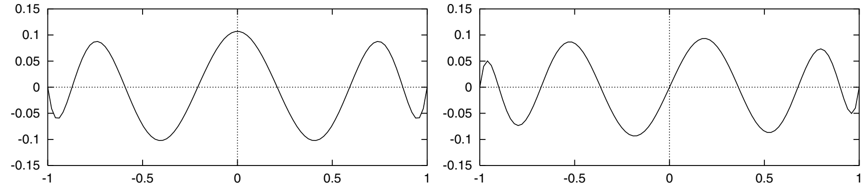

FIGURE 1.18: Lobatto shape functions l_8, l_9.

1.2.4 Kernel functions

For future use it is convenient to decompose the higher-order Lobatto shape functions $l_2, l_3, \dots$ into products of the form

$$l_k(x) = l_0(x)l_1(x)\phi_{k-2}(x), \quad 2 \le k. \tag{1.52}$$

Since all functions l_k, $2 \le k$ vanish at ± 1, the *kernel functions* ϕ_{k-2}, $k = 2, 3, \dots$ are *polynomials* of the order $k-2$. Let us list the following,

$$\begin{aligned}
\phi_0(x) &= -2\sqrt{\frac{3}{2}}, \\
\phi_1(x) &= -2\sqrt{\frac{5}{2}}x, \\
\phi_2(x) &= -\frac{1}{2}\sqrt{\frac{7}{2}}(5x^2-1), \\
\phi_3(x) &= -\frac{1}{2}\sqrt{\frac{9}{2}}(7x^2-3)x, \\
\phi_4(x) &= -\frac{1}{4}\sqrt{\frac{11}{2}}(21x^4-14x^2+1), \\
\phi_5(x) &= -\frac{1}{4}\sqrt{\frac{13}{2}}(33x^4-30x^2+5)x, \\
\phi_6(x) &= -\frac{1}{32}\sqrt{\frac{15}{2}}(429x^6-495x^4+135x^2-5), \\
\phi_7(x) &= -\frac{1}{32}\sqrt{\frac{17}{2}}(715x^6-1001x^4+385x^2-35)x, \\
\phi_8(x) &= -\frac{1}{64}\sqrt{\frac{19}{2}}(2431x^8-4004x^6+2002x^4-308x^2+7), \\
&\ \vdots
\end{aligned} \tag{1.53}$$

The kernel functions $\phi_0, \phi_1, \dots$ will be used for the definition of higher-order hierarchic shape functions on triangular, tetrahedral and prismatic elements in Chapter 2.

1.2.5 Horner's algorithm for higher-order polynomials

An attempt to implement the formulae (1.48), (1.51) and (1.53) in the same form as they are written on paper could cause significant roundoff errors for higher polynomial orders. Here is a simple explanation of what would happen.

Recall that a floating point operand is stored in the form of a *mantisa* and *exponent*. As long as the number of decimal digits does not exceed the length of mantisa, no information is lost. Therefore we can have both extremely large and extremely small numbers stored exactly. However, problems arise when summation is involved, as operands are set to have the same exponent, and digits stored in the mantisa are shifted (to the right if exponent is increased, or to the left if it is decreased). In either case digits get lost when the difference between the exponents is large.

For the evaluation of higher-order polynomials this means that we should avoid direct summation of contribution of the form ax^n, bx^m where the powers m and n significantly differ. One of the standard ways to avoid these problems (sometimes called *Horner's algorithm*) is to transform the polynomials into the form

$$\sum_{n=0}^{M} a_n x^n = a_0 + x(a_1 + x(a_2 + x(a_3 + \ldots))). \tag{1.54}$$

A simple concrete example would be

$$a_0 + a_1x^2 + a_2x^4 + a_3x^6 = a_0 + x^2(a_1 + x^2(a_2 + a_3x^2)). \tag{1.55}$$

Moreover, computer summation takes less time than computer multiplication and is therefore a more efficient means to evaluate higher-order polynomials.

1.3 A one-dimensional example

The technical simplicity of the one-dimensional situation allows us to present the higher-order discretization procedure in detail. Let us begin with the formulation of a simple model problem.

1.3.1 Continuous and discrete problem

Consider an interval $I = (a, b) \subset \mathbb{R}$ and a load function $f \in L^2(I)$. We will solve the Poisson equation

$$-u''(x) = f(x) \tag{1.56}$$

in I, equipped with nonhomogeneous Dirichlet boundary conditions

$$\begin{aligned} u(a) &= g_a, \\ u(b) &= g_b. \end{aligned} \tag{1.57}$$

We proceed according to Paragraph 1.1.6. First notice that functions satisfying conditions (1.57) cannot constitute a vector space. This would be the case with zero Dirichlet boundary conditions; however, now the sum of two functions satisfying (1.57) does not satisfy it anymore. Therefore we have to decompose the sought function u into

$$u(x) = u^*(x) + \bar{u}(x), \tag{1.58}$$

where the *Dirichlet lift* $u^* \in H^1(a,b)$ satisfies the boundary conditions (1.57),

$$\begin{aligned} u^*(a) &= g_a, \\ u^*(b) &= g_b, \end{aligned} \tag{1.59}$$

and the function $\bar{u}$, satisfying homogeneous Dirichlet boundary conditions

$$\bar{u}(a) = \bar{u}(b) = 0, \tag{1.60}$$

is the unknown part of the solution u. The function $\bar{u}$ already can be sought in a linear function space, namely

$$V = H_0^1(a,b). \tag{1.61}$$

Hence, the task is to find a function $\bar{u} \in V$ satisfying the variational formulation

$$\int_a^b [u^*(x) + \bar{u}(x)]' v'(x) \mathrm{d}\boldsymbol{x} = \int_a^b f(x)v(x)\mathrm{d}\boldsymbol{x} \quad \text{for all } v \in V. \tag{1.62}$$

This is the same as

$$\int_a^b (u^*)'(x)v'(x)\mathrm{d}\boldsymbol{x} + \int_a^b (\bar{u})'(x)v'(x)\mathrm{d}\boldsymbol{x} = \int_a^b f(x)v(x)\mathrm{d}\boldsymbol{x} \quad \text{for all } v \in V \tag{1.63}$$

and as

$$\int_a^b (\bar{u})'(x)v'(x)\mathrm{d}\boldsymbol{x} = \int_a^b f(x)v(x) - (u^*)'(x)v'(x)\mathrm{d}\boldsymbol{x} \quad \text{for all } v \in V. \tag{1.64}$$

Discretization of (1.64)

In the next step we specify a finite element mesh $\mathcal{T}_{h,p} = \{K_1, K_2, \ldots, K_M\}$ of elements with arbitrary polynomial orders $1 \le p_1, p_2, \ldots, p_M$. We choose a reference domain $K_a = (-1,1)$ and for each element $K_i = (x_i, x_{i+1})$, $i = 1, 2, \ldots, M$ we define an affine reference map $x_{K_i} : K_a \to K_i$,

$$\begin{aligned} x_{K_i}(\xi) &= c_1^{(i)} + c_2^{(i)}\xi, \\ x_{K_i}(-1) &= x_i, \\ x_{K_i}(1) &= x_{i+1}. \end{aligned} \tag{1.65}$$

Hence it is

$$c_1^{(i)} = \frac{x_i + x_{i+1}}{2}, \quad c_2^{(i)} = J_{K_i} = \frac{x_{i+1} - x_i}{2}. \tag{1.66}$$

The space V is approximated by a subspace

$$V_{h,p} = \{v \in V;\ v|_{K_i} \circ x_{K_i} \in P^{p_i}(K_a) \text{ for all } i = 1, 2, \ldots, M\}, \tag{1.67}$$

where $(f \circ g)(x) \equiv f(g(x))$, of the dimension

$$N = \dim(V_{h,p}) = \underbrace{M - 1}_{\text{first-order part}} + \underbrace{\sum_{i=1}^{M}(p_i - 1)}_{\text{higher-order part}} = -1 + \sum_{i=1}^{M} p_i. \tag{1.68}$$

Hence, the approximate variational formulation (discrete problem) is

$$\int_a^b (\bar{u}_{h,p})'(x) v'_{h,p}(x) \mathrm{d}\boldsymbol{x} = \int_a^b f(x) v_{h,p}(x) - (u^*_{h,p})'(x) v'_{h,p}(x) \mathrm{d}\boldsymbol{x} \tag{1.69}$$

for all $v_{h,p} \in V_{h,p}$.

Choice of the lift function u^*

It is natural to require that the solution $u_{h,p} = u^*_{h,p} + \bar{u}_{h,p}$ is a polynomial of the order p_i on each element K_i, $i = 1, 2, \ldots, M$. Since $\bar{u}_{h,p} \in V_{h,p}$, this only can be achieved if the lift function $u^*_{h,p}$ is a polynomial of the order p_i on each element K_i as well. In practice one usually selects $u^*_{h,p}$ to be as simple as possible, i.e., as a continuous piecewise-linear function that vanishes in all interior elements (see Figure 1.19).

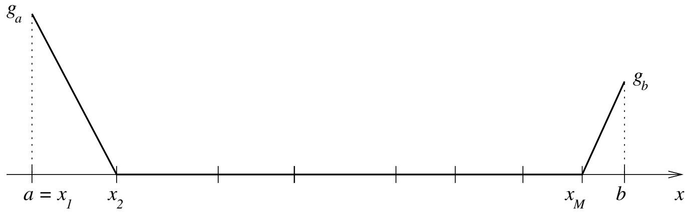

FIGURE 1.19: Example of a Dirichlet lift function for 1D problems.

In the following we will describe an algorithm that turns (1.69) into a system of linear algebraic equations.

1.3.2 Transformation to reference domain

The first step toward an efficient element-by-element assembly of the discrete problem is to transform the approximate variational formulation (1.69) elementwise from the mesh $\mathcal{T}_{h,p}$ to the reference domain K_a.

Transformation of function values

By $\bar{u}_{h,p}^{(i)}(\xi)$ we denote the approximate solution $\bar{u}_{h,p}$, transformed from the element K_i to the reference domain K_a, i.e.,

$$\bar{u}_{h,p}^{(i)}(\xi) \equiv (\bar{u}_{h,p} \circ x_{K_i})(\xi) = \bar{u}_{h,p}(x)|_{x=x_{K_i}(\xi)}. \tag{1.70}$$

In other words, the function value of $\bar{u}_{h,p}^{(i)}$ at a reference point $\xi \in K_a$ has to be the same as the function value of $\bar{u}_{h,p}$ at its image $x_{K_i}(\xi) \in K_i$.

Transformation of derivatives

One has to be a little more careful when transforming derivatives. The chain rule yields

$$[\bar{u}_{h,p}^{(i)}(\xi)]' = (u_{h,p} \circ x_{K_i})'(\xi) = u'_{h,p}(x)|_{x=x_{K_i}(\xi)} J_{K_i}. \tag{1.71}$$

This means that

$$[\bar{u}_{h,p}]'(x) = \frac{1}{J_{K_i}}[\bar{u}_{h,p}^{(i)}]'(\xi), \tag{1.72}$$

i.e., the derivative of $\bar{u}_{h,p}$ at the physical point $x = x_{K_i}(\xi) \in K_i$ is expressed as the derivative of the new function $\bar{u}_{h,p}^{(i)}$ at the corresponding reference point $\xi \in K_a$ divided by the Jacobian J_{K_i} (which obviously for the affine map x_{K_i} is constant).

Transformation of integrals from the variational formulation

The test functions $v_{h,p}$ and their derivatives are transformed analogously. Using the Substitution Theorem (that produces a factor J_{K_i} behind the integral sign), it is easy to conclude that

$$\int_{K_i} [\bar{u}_{h,p}]'(x)v'_{h,p}(x)\mathrm{d}\boldsymbol{x} = \int_{K_a} \frac{1}{J_{K_i}}[\bar{u}_{h,p}^{(i)}]'(\xi)[\bar{v}_{h,p}^{(i)}]'(\xi)\,\mathrm{d}\xi \quad \text{for all } i = 1, 2, \ldots, M. \tag{1.73}$$

The right-hand side is transformed analogously,

$$\begin{aligned}&\int_{K_i} f(x)v_{h,p}(x) - (u^*_{h,p})'(x)v'_{h,p}(x)\mathrm{d}\boldsymbol{x} \qquad (1.74)\\ &= \int_{K_a} J_{K_i} f^{(i)}(\xi)v_{h,p}^{(i)}(\xi) - \frac{1}{J_{K_i}}[u_{h,p}^{*(i)}]'(\xi)[v_{h,p}^{(i)}]'(\xi)\,\mathrm{d}\xi,\end{aligned}$$

where $f^{(i)}(\xi) = (f \circ x_{K_i})(\xi)$ and so on.

1.3.3 Higher-order shape functions

Basic ideas of the design of *nodal* and *hierarchic* elements were revisited in Paragraph 1.1.2. The one-dimensional model problem offers a good chance to look in more detail at both the nodal and hierarchic shape functions and their conditioning properties.

Nodal higher-order shape functions

Consider an element K_i of the polynomial order p_i. For simplicity we distribute the nodal points equidistantly, i.e., we define

$$X_{j+1} = -1 + \frac{2j}{p_i} \in K_a, \quad j = 0, 1, \dots, p_i. \tag{1.75}$$

Exploiting the Lagrange interpolation polynomial and the δ-property (1.6), we obtain the $p_i + 1$ nodal functions of the order p_i of the form

$$\theta_i(\xi) = \frac{\prod\limits_{1\le j\le p_i+1, j\ne i} (\xi - X_j)}{\prod\limits_{1\le j\le p_i+1, j\ne i} (X_i - X_j)}. \tag{1.76}$$

For piecewise linear approximations ($p_i = 1$) there are two points $X_1 = -1$, $X_2 = 1$, and the two linear (affine) shape functions have the form

$$\theta_1(\xi) = \frac{1-\xi}{2}, \ \theta_2(\xi) = \frac{\xi+1}{2}. \tag{1.77}$$

Complete sets of nodal shape functions for $p_i = 2$ and $p_i = 3$ are illustrated in Figures 1.20 and 1.21.

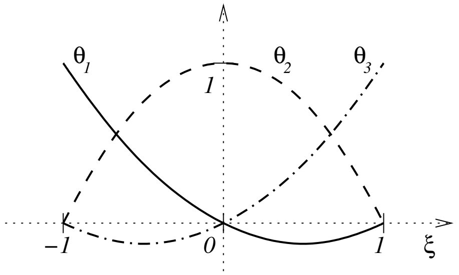

FIGURE 1.20: Quadratic nodal shape functions.

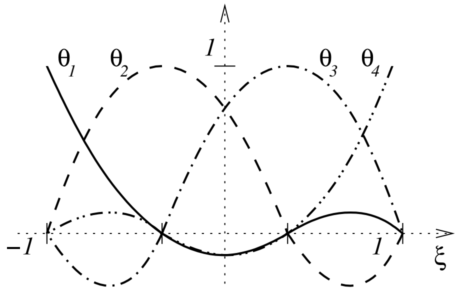

FIGURE 1.21: Cubic nodal shape functions.

These shape functions are advantageous from the point of view that the corresponding unknown coefficients, obtained from the solution of the discrete problem, directly represent the value of the approximate solution $u_{h,p}$ at the geometrical nodes X_i. On the other hand they are not hierarchic and thus one has to replace the whole set of shape functions when increasing the polynomial order of elements. Further, in higher spatial dimensions it is difficult to combine nodal elements with various polynomial orders in the mesh and therefore they are not suitable for *p*- and *hp*-adaptivity. With simple choices of nodal points these shape functions yield ill-conditioned stiffness matrices.

Hierarchic higher-order shape functions

As mentioned in Paragraph 1.1.2, by hierarchic we mean that the basis $\mathcal{B}^{p+1}$ of the polynomial space $P_{p+1}(K_a)$ is obtained from the basis $\mathcal{B}^p$ of the polynomial space $P_p(K_a)$ by adding new shape functions only. Particularly in 1D we add always a single $(p+1)$th-order shape function to the previous basis only. This is essential for *p*- and *hp*-adaptive finite element codes since one does not have to change his shape functions completely when increasing the order of polynomial approximation. Among hierarchic shape functions, one of the most popular choices is *Lobatto shape functions* (integrated Legendre polynomials) $l_0, l_1, \dots$ that we introduced in Section 1.2. Their excellent conditioning properties are rooted in the fact that their derivatives are (normalized) Legendre polynomials, and thus that their H_0^1-product satisfies

$$\int_{-1}^{1} l'_{i-1}(\xi) l'_{j-1}(\xi)\,\mathrm{d}\xi = 0 \quad \text{whenever } i > 2 \text{ or } j > 2, \quad i \neq j. \tag{1.78}$$

DEFINITION 1.9 (Master element stiffness matrix) *Let θ_1, θ_2, ..., θ_{p+1} be a basis in the polynomial space $P^p(K_a)$. Then the master element stiffness matrix of order p corresponding to the problem (1.56), i.e., to the Laplace operator in 1D, is a $(p+1)\times(p+1)$ matrix of the form*

$$\hat{\boldsymbol{S}} = \{\hat{s}_{ij}\}_{i,j=1}^{p+1}, \ \hat{s}_{ij} = \int_{K_a} \theta_i'(\xi)\theta_j'(\xi)\, d\xi. \tag{1.79}$$

REMARK 1.10 (Role of master element stiffness matrix) It was shown in Paragraph 1.3.2 that stiffness integrals from the variational formulation, transformed from an element $K_i \in \mathcal{T}_{h,p}$ to the reference interval K_a, keep their original form up to the multiplication by an element-dependent constant (which was the inverse Jacobian of the affine map x_{K_i}). Thus, all integrals, which are needed for the assembly of the global stiffness matrix, are available in the master element stiffness matrix. This essentially reduces the cost of the computation.

Unfortunately the master element stiffness matrix cannot be exploited in this way when the operator in the variational formulation is explicitly space-dependent (consider, e.g., the operator $\tilde{L}(u) = ((1+x)u')'$ instead of the Laplace operator $L(u) = u''$).

Master element stiffness matrices can also be utilized in the same way in 2D and 3D when the reference maps are affine. Stiffness contributions of elements which are equipped with other than affine reference maps must be physically integrated on every mesh element. $\square$

The relation (1.78) implies that the master element stiffness matrix $\hat{\boldsymbol{S}}$ of the order p for the Lobatto shape functions $l_0, l_1, \ldots, l_p$ looks like

$$\hat{\boldsymbol{S}} = \begin{pmatrix} \hat{s}_{11} & \hat{s}_{12} & 0 & 0 & \ldots & 0 \\ \hat{s}_{21} & \hat{s}_{22} & 0 & 0 & & \vdots \\ 0 & 0 & \hat{s}_{33} & 0 & & \\ 0 & 0 & 0 & \ddots & & \\ \vdots & & & & & 0 \\ 0 & \ldots & & & 0 & \hat{s}_{p+1,p+1} \end{pmatrix}. \tag{1.80}$$

The only nonzero nondiagonal entries correspond to products of the first-order shape functions l_0, l_1. This sparse structure is what makes the Lobatto shape functions so popular for the discretization of problems involving the Laplace operator.

DEFINITION 1.10 (Condition number) *Let $\boldsymbol{M}$ be a regular $n \times n$ matrix. The product*

$$\varkappa(\boldsymbol{M}) = \|\boldsymbol{M}\|\|\boldsymbol{M}^{-1}\|, \tag{1.81}$$

where $\|\,.\,\|$ is a matrix norm, is called the condition number *of the matrix $\boldsymbol{M}$ (relative to the norm $\|\,.\,\|$).*

REMARK 1.11 (Spectral condition number) The matrix norm $\|.\|$ in Definition 1.10 can be chosen in many different ways. Most commonly used are the *Euclidean (Frobenius) norm*

$$\|\boldsymbol{M}\| = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} m_{ij}^2} \tag{1.82}$$

and the *spectral norm*

$$\|\boldsymbol{M}\| = \sqrt{\rho(\boldsymbol{M}\boldsymbol{M}^T)}, \tag{1.83}$$

where $\rho(\boldsymbol{M}\boldsymbol{M}^T)$ is the *spectral radius* (i.e., the largest eigenvalue) of the symmetric positive definite matrix $\boldsymbol{M}\boldsymbol{M}^T$.

Using the spectral norm (1.83), one arrives at the frequently used *spectral (Todd) condition number*

$$1 \le \varkappa^*(\boldsymbol{M}) = \frac{\max_{\lambda\in\sigma(\boldsymbol{M})} |\lambda|}{\min_{\lambda\in\sigma(\boldsymbol{M})} |\lambda|}, \tag{1.84}$$

(see, e.g., [195]) where $\sigma(\boldsymbol{M})$ is the *spectrum* (set of all eigenvalues) of the matrix $\boldsymbol{M}$. It holds

$$\varkappa^*(\boldsymbol{M}) \le \varkappa(\boldsymbol{M})$$

for any other matrix norm. It is a widely known fact that the performance of solvers for systems of linear algebraic equations improves in accordance with lower condition numbers of the solved matrices. $\square$

Example 1.5 (Conditioning properties of various types of shape functions)
Conditioning of the master element stiffness matrix is used as an orientation factor for the selection of optimal shape functions. Let us return to the Laplace operator in one spatial dimension. To illustrate the importance of a good choice of shape functions, in addition to the nodal shape functions (1.76) and Lobatto shape functions (1.49) consider a set of hierarchic shape functions of the simple form

$$\begin{aligned} \theta_1(\xi) &= \frac{1-\xi}{2}, \\ \theta_2(\xi) &= \frac{1+\xi}{2}, \\ \theta_k(\xi) &= \frac{(1-\xi)^{k-1}(1+\xi)}{4}. \end{aligned} \tag{1.85}$$

Figure 1.22 compares the conditioning of the corresponding master element stiffness matrices. More precisely, for $p = 2, 3, \ldots, 10$ we depict the spectral condition number of the submatrix of the master element stiffness matrix, corresponding to the bubble functions only (the master element stiffness matrix itself corresponds to the solution of the original problem using one single element and no boundary conditions, and obviously it is singular).

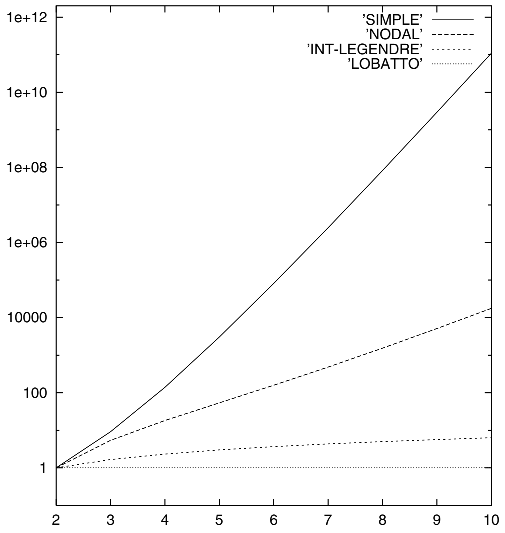

FIGURE 1.22: Conditioning of master element stiffness matrices for the simple shape functions (1.85), nodal shape functions (1.76), integrated Legendre polynomial (without normalization) and finally Lobatto shape functions (1.49), in this order. Notice that the scale is decimal logarithmic.

For the sake of completeness let us add that, unfortunately, conditioning properties of bubble functions get worse in higher spatial dimensions. The case $\varkappa \equiv 1$ is idealistic and only true for the Lobatto shape functions in 1D. $\square$

REMARK 1.12 (Invariance of the condition number) The condition numbers of (bubble-submatrices of) the master element stiffness matrices, shown in Figure 1.22, are invariant with respect to permutation of indices of bubble functions. To see this, it is sufficient to consider a permutation that exchanges the indices k, l in a pair of bubble functions θ_k and θ_l only. In this case it easily follows from the definition that all eigenvalues of the new matrix will be the same, and the new eigenvectors will be obtained from the original ones by switching their kth and lth components. $\square$

1.3.4 Design of basis functions

Both sets of the nodal and hierarchic higher-order shape functions that we have introduced in Paragraph 1.3.3 consist of *vertex* and *bubble* shape functions. The vertex shape functions are nonzero at one of the endpoints of the reference interval $K_a = (-1, 1)$ and vanish at the other (in the hierarchic case these are l_0 and l_1, in the nodal case θ_1 and θ_{p+1}). The rest are bubble shape functions that vanish at both endpoints. Accordingly, the basis functions of the space $V_{h,p}$ will be split into vertex and bubble functions.

Vertex basis functions

The vertex functions are related to grid points and their support consists of two neighboring elements. Consider a grid point x_{i+1} and the adjacent elements K_i, K_{i+1}. In the hierarchic case the corresponding vertex basis function v_i is defined as

$$v_i(x) = \begin{cases} (l_1 \circ x_{K_i}^{-1})(x), & x \in K_i, \\ \\ (l_0 \circ x_{K_{i+1}}^{-1})(x), & x \in K_{i+1}. \end{cases} \tag{1.86}$$

These functions are sometimes called "hat functions" (see Figure 1.23).

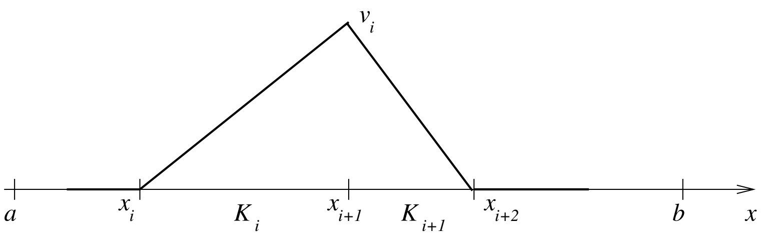

FIGURE 1.23: Vertex basis functions v_i of the space $V_{h,p}$ in the hierarchic case.

In the context of nodal elements the vertex basis functions are defined as

$$v_i(x) = \begin{cases} (\theta_{p+1} \circ x_{K_i}^{-1})(x), & x \in K_i, \\ \\ (\theta_1 \circ x_{K_{i+1}}^{-1})(x), & x \in K_{i+1}. \end{cases} \tag{1.87}$$

Here p is the polynomial order associated with both the elements K_i and K_{i+1}. An example of a quadratic nodal vertex basis function is depicted in Figure 1.24.

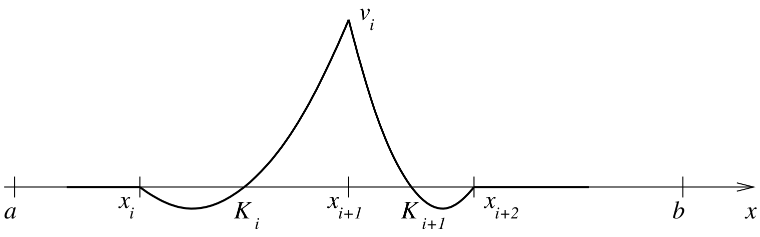

FIGURE 1.24: Vertex nodal basis functions v_i of the space $V_{h,p}$ for piecewise-quadratic approximations.

Bubble basis functions

Bubble basis functions for an element K_i of pth order are defined analogously as a composition of bubble shape functions and the inverse map $x_{K_i}^{-1}$. In the hierarchic case one uses the bubble shape functions $l_2, l_3, \ldots, l_p$, and in the nodal case the bubble shape functions $\theta_2, \theta_3, \ldots, \theta_p$. Examples of hierarchic quadratic and cubic bubble functions are shown in Figures 1.25 and 1.26.

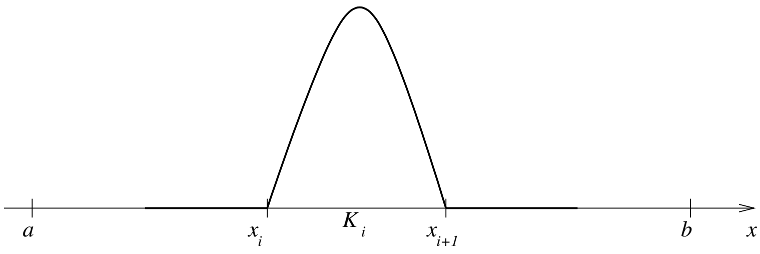

FIGURE 1.25: An example of a hierarchic quadratic bubble basis function.

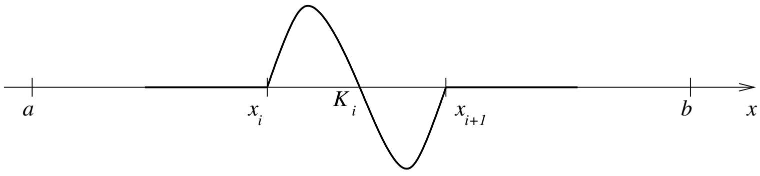

FIGURE 1.26: An example of a hierarchic cubic bubble basis function.

1.3.5 Sparsity structure and connectivity

It is widely known that when indexing the grid points in the interval $I = (a, b)$ consecutively from the left to the right, the global stiffness matrix for piecewise-linear approximations is tridiagonal, analogously to finite difference schemes. This analogy between first-order finite elements and finite differences extends to 2D and 3D when using Cartesian grids. However, higher-order finite elements in 1D and already first-order FE discretizations on unstructured meshes in 2D and 3D yield general sparse structures.

Enumeration of basis functions of $V_{h,p}$

The sparsity structure of the stiffness matrix is uniquely determined by the ordering of the basis functions of the space $V_{h,p}$. Generally one is free to index the basis functions in any way she/he wishes; however, from the point of view of compatibility between first- and higher-order approximations it is reasonable to put the vertex functions according to consecutively denumerated grid points first. After that, all higher-order basis functions can be denumerated consecutively within elements according to their polynomial order, one element at a time.

Connectivity information

The connectivity information is a data construction which for each element K_i links the master element shape functions $l_0, l_1, l_2, \dots$ to the basis functions $v_1, v_2, \dots, v_N \subset V_{h,p}$ in the finite element mesh (let us work with Lobatto shape functions for instance). These data are stored elementwise and their actual amount depends on the needs (and sophistication) of the assembling algorithm. The simplest option, which surely is not the most efficient one possible, is to store for each mesh element K_i an integer index array $\boldsymbol{c}_i$ of the length $p_i + 1$, where p_i is the order of approximation on K_i. This array is filled with the indices of basis functions of $V_{h,p}$, which are related to shape functions $l_0, l_1, \dots, l_{p_i}$ on K_i:

$$\begin{array}{c} l_0 \dots c_{i,1} \\ l_1 \dots c_{i,2} \\ l_2 \dots c_{i,3} \\ \vdots \\ l_{p_i} \dots c_{i,p_i+1} \end{array} \tag{1.88}$$

We may use here, e.g., -1 instead of an index in order to indicate that a shape function is not related to any basis function because of a Dirichlet boundary condition.

Example 1.6 (Connectivity information)
Consider a mesh $\mathcal{T}_{h,p}$ consisting of three elements K_1, K_2 and K_3 of polynomial orders $p_1 = 3$, $p_2 = 4$ and $p_3 = 2$. Let us look at connectivity data for a discretization with zero Dirichlet boundary conditions on both interval endpoints.

In this case the dimension of the space $V_{h,p}$ is $N = 8$ and the element connectivity arrays look like the following:

$$\begin{aligned} K_1 \ldots \boldsymbol{c}_1 &= \{-1, 1, 3, 4\}, \\ K_2 \ldots \boldsymbol{c}_2 &= \{1, 2, 5, 6, 7\}, \\ K_3 \ldots \boldsymbol{c}_3 &= \{2, -1, 8\}. \end{aligned}$$

The components of these index arrays are related to the shape functions $l_0, l_1, l_2, \ldots$ in this order. In this case $c_{1,1} = c_{3,2} = -1$ means that the shape function l_0 on element K_1 and shape function l_1 on element K_3 are not used due to the Dirichlet boundary conditions. $\square$

Let us show a general algorithm that defines connectivity information in 1D for various types of boundary conditions at the interval endpoints $x_1 = a$ and $x_{M+1} = b$, and for an arbitrary distribution of the polynomial orders p_i, $i = 1, 2, \ldots, M$.

ALGORITHM 1.1 (Preparing connectivity data in 1D)

```
1. counter := 1
2. First-order basis functions of the element $K_1$:
if(Dirichlet boundary condition at a) then $c_{1,1} := -1$
else $c_{1,1} :=$ counter, counter := counter + 1
Shape function $l_1$ for $K_1$ corresponds to $v_{counter}$:
$c_{1,2} :=$ counter
3. Loop over elements $K_2, K_3, \ldots, K_{M-1}$ indexing hat functions:
for($k = 2$ to $M - 1$) do {
   (global index for $l_0$:) $c_{k,1} :=$ counter, counter := counter + 1
   (global index for $l_1$:) $c_{k,2} :=$ counter
}
4. First-order basis functions of the element $K_M$:
Shape function $l_0$ for $K_M$ corresponds to $v_{counter}$:
$c_{M,1} :=$ counter, counter := counter + 1
if(Dirichlet boundary condition at b) then $c_{M,2} := -1$
else $c_{M,2} :=$ counter, counter := counter + 1
5. Loop over all elements indexing higher-order basis functions:
for($k = 1$ to $M$) do {
  for($p = 2$ to $p_k$) do {
    (global index for $l_p$:) $c_{k,p+1}$ := counter, counter := counter + 1
  }
}
```

Treatment of connectivity information in higher spatial dimensions will be discussed in detail in Chapter 3.

1.3.6 Assembling algorithm

In higher spatial dimensions we need to store a flag for each element specifying whether it lies on the boundary or not, and a link to the appropriate boundary data. This is not necessary in 1D since we know that if -1 is the first entry in the connectivity array, we are on element K_1, and we are on element K_M if -1 is its second entry. The algorithm consists basically of *one single loop* over all mesh elements $K_1, K_2, \ldots, K_M$. This is an essential difference with respect to first-order discretizations where it is sufficient to loop over grid vertices – the first-order approach indeed does not generalize to higher-order schemes.

ALGORITHM 1.2 (Assembling algorithm)

Evaluate the master element stiffness matrix $\hat{\boldsymbol{S}}$ corresponding to the highest polynomial order in the mesh. If this is not possible, for example because the operator is explicitly space-dependent, one will have to integrate stiffness terms analogous to (1.79) on each mesh element.
Store the Jacobian of the reference map x_{K_i} for each element in the mesh.
Set the matrix $\boldsymbol{S} = \{s_{ij}\}_{i,j=1}^N$ zero.
Set the right-hand side vector $\boldsymbol{F} = (F_1, F_2, \ldots, F_N)^T$ zero.

```
(element loop:) for $k = 1, 2, \ldots, M$ do {
  (first loop over shape (test) functions:) for $i = 1, 2, \ldots, p_k + 1$ do {
    (second loop over shape (basis) functions:) for $j = 1, 2, \ldots, p_k + 1$ do {
      put $m_1 = c_{k,i}$ (if not $-1$, this is the global index of a test function $v_{m_1} \in V_{h,p}$, i.e., row in the global stiffness matrix)
      put $m_2 = c_{k,j}$ (if not $-1$, this is the global index of a basis function $v_{m_2} \in V_{h,p}$, i.e., column in the global stiffness matrix)
      if($m_1 \neq -1$ and $m_2 \neq -1$) then put $s_{m_1,m_2} = s_{m_1,m_2} + \frac{1}{J_{K_k}}\hat{s}_{i,j}$
      else { (beginning of treatment of Dirichlet bdy. conditions)
        if($m_1 \neq -1$ and $m_2 == -1$) then { (the condition $m_2 == -1$ means that the shape functions $l_0, l_1$ can represent the Dirichlet lift $u_{h,p}^*$. By $m_1 \neq -1$ we do not allow them to represent test functions from the space
$V_{h,p}$ – recall that all functions from the space $V_{h,p} \subset H_0^1(a,b)$ must respect homogeneous Dirichlet bdy. conditions)
          **if**($j == 1$) **then** { (we are on $K_1$ and use $g_a l_0(\xi)$ for the transformed Dirichlet lift)
            **put** $F_{m_1} = F_{m_1} - \int_{K_a} \frac{1}{J_{K_1}} g_a l_0'(\xi) l_{i-1}'(\xi)\, d\xi = F_{m_1} - \frac{1}{J_{K_1}} g_a \hat{s}_{1,i}$ (extra contribution to the right-hand side)
          }
          **else** { (now $j == 2$, we are on $K_M$ and use $g_b l_1(\xi)$ for the transformed Dirichlet lift)
            **put** $F_{m_1} = F_{m_1} - \int_{K_a} \frac{1}{J_{K_M}} g_b l_1'(\xi) l_{i-1}'(\xi)\, d\xi = F_{m_1} - \frac{1}{J_{K_1}} g_b \hat{s}_{2,i}$ (extra contribution to the right-hand side)
          }
        }
      } (end of treatment of Dirichlet bdy. conditions)
    } (end of second loop over shape (basis) functions)
    **if**($m_1 \neq -1$) **then put** $F_{m_1} = F_{m_1} + \int_{K_a} J_{K_k} \tilde{f}^{(k)}(\xi) l_{i-1}(\xi)\, d\xi$ (regular contribution to the right-hand side)
  } (end of first loop over shape (test) functions)
} (end of element loop)
(where again $\tilde{f}^{(k)}(\xi) = f(x_{K_k}(\xi))$).
```

The assembling algorithm for 2D and 3D approximations will be introduced in Chapter 3.

1.3.7 Compressed representation of sparse matrices

There is a well-established compressed format for the representation of sparse matrices (*Compressed Sparse Row (CSR)*: see, e.g., [72, 156, 169, 193]). Once one adapts to this format, she/he will be able to find many software packages that will precondition and solve the discrete problem. Let N be the rank of the matrix $\boldsymbol{S}$ (i.e., the number of unknown coefficients of the discrete problem) and by NNZ denote the number of nonzero entries in $\boldsymbol{S}$. Virtually all sparse matrix solvers require that the matrix $\boldsymbol{S}$ is written in the form of three arrays:

1. Array A of length NNZ: this is a real-valued array containing all nonzero entries of the matrix $\boldsymbol{S}$ listed from the left to the right, starting with the first and ending with the last row.

2. Array IA of length $N+1$: this is an integer array, $IA[1] = 1$. $IA[k+1] = IA[k] + nnz_k$ where nnz_k is the number of nonzero entries in the kth row.

3. Array JA of length NNZ: this is an integer array containing the row-positions of all entries from array A.

Chapter 2

Hierarchic master elements of arbitrary order

The first step in the technology of hierarchic higher-order finite element methods is to design suitable master elements of arbitrary polynomial order. We will consider the most commonly used reference domains, equip them with appropriate scalar and vector-valued polynomial spaces and define hierarchic higher-order shape functions. Some of the constructions are actually quite exciting, particularly in vector-valued spaces in higher spatial dimensions, but in each case this chapter is intended merely as a database of formulae rather than information for systematic study. The reader may find it interesting to read about the De Rham diagram (Section 2.1) which relates the spaces H^1, $\boldsymbol{H}(\mathrm{curl})$, $\boldsymbol{H}(\mathrm{div})$ and L^2 by means of differential operators, since finite elements in these spaces have to respect the diagram as well. Then she/he may visit a paragraph that discusses a particular finite element of interest.

The procedure of design of hierarchic elements is a little dull, but one has to go through the exercise once. The hierarchic shape functions form families that have to be constructed separately for each reference domain and each function space, and each time one has to make sure that the shape functions really constitute a basis. Although very interesting relations among some of these families exist (see [115]), we confine ourselves to our goal, which is to provide a database of formulae suitable for computer implementation (see also [183, 184]).

Symbol Description

In the following we will encounter numerous shape functions related to various types of reference domains and function spaces. It seems that the only reasonable way to keep this number of definitions clear is to establish a consistent index notation, which is generous enough to cover all significant differences. Sometimes we will have to attach multiple upper and lower indices to a single symbol. This is done consistently throughout the chapter and the rest of the book.

K	reference domain,	λ	affine coordinates,
$\mathcal{K}$	master element (finite element on a reference domain),	φ	H^1-hierarchic scalar shape functions,

ψ $\boldsymbol{H}$(curl)-hierarchic vector-valued shape functions,
γ $\boldsymbol{H}$(div)-hierarchic vector-valued shape functions,
ω scalar shape functions for L^2-conforming approximations,
$\boldsymbol{t}$ unitary tangential vector,
$\boldsymbol{n}$ unitary normal vector,
$\mathcal{P}$ scalar polynomial space on reference domains K_a, K_t, K_T (related to one-dimensional, triangular and tetrahedral elements),
$\mathcal{Q}$ scalar polynomial space on reference domains K_q, K_B (related to quadrilateral and brick elements),
W scalar polynomial space on master elements (H^1-conforming case),
$\boldsymbol{Q}$ vector-valued polynomial space on master elements ($\boldsymbol{H}$(curl)-conforming case),
$\boldsymbol{V}$ vector-valued polynomial space on master elements ($\boldsymbol{H}$(div)-conforming case),
X scalar polynomial space on master elements (L^2-conforming case).

Indices – letters:
a indicates relation to the reference interval K_a,
t indicates relation to the reference triangle K_t,
q indicates relation to the reference quadrilateral K_q,
T indicates relation to the reference tetrahedron K_T,
B indicates relation to the reference brick K_B (unless specified otherwise),
P indicates relation to the reference prism K_P,
v_1, v_2, v_3 indicates relation to reference domain vertices,
e_1, e_2, e_3 indicates relation to reference domain edges,
s_1, s_2, s_3 indicates relation to reference domain faces,
b indicates relation to reference domain interior.

Indices – numbers:
$1, 2, 3$ in the lower index: direction of approximation,
n (one single number) in the lower index: enumeration of affine coordinates λ, accompanied by another index indicating the element type,
n (one single number) in the lower index: polynomial order identifying a shape function if there is exactly one shape function for each n,
n_1, n_2 (two numbers) in the lower index: numbers identifying a shape function if there are more of them for each order,
n_1, n_2, n_3 (three numbers) in the lower index: numbers identifying a shape function if there are more of them for each order.

Basic terminology

By *locally nonuniform distribution of order of polynomial approximation* we mean that mesh elements adjacent to each other carry different orders of polynomial approximation. A necessary condition for the implementation of this feature, which is essential for *hp*-adaptivity, is the separation of degrees of freedom into *internal* (associated with element interior) and *external* (associated with element interfaces), and their hierarchic structure. This feature makes an essential difference with respect to *nodal* higher-order elements (see Example 1.3).

Anisotropic p-refinement of an element means a p-refinement that results in different orders of polynomial approximation in various directions. This will be relevant for quadrilaterals, bricks and prisms (our only elements with product structure). If the solution, transformed to the reference domain, exhibits major changes in one axial direction, it is not necessary to increase both directional orders of approximation – this feature allows for efficient resolution of boundary and internal layers.

By *constrained approximation* we mean approximation on *irregular meshes* (meshes with hanging nodes in the sense of Paragraph 1.1.3). In combination with anisotropic p-adaptivity, constrained approximation capability is essential for efficient implementation of automatic hp-adaptivity. An introduction to automatic hp-adaptivity will be given in Chapter 6.

2.1 De Rham diagram

The *De Rham diagram* is a scheme that relates the function spaces H^1, $\boldsymbol{H}(\mathrm{curl})$, $\boldsymbol{H}(\mathrm{div})$ and L^2, as well as finite elements in these spaces, by means of differential operators. Its essential importance for finite element methods in the spaces $\boldsymbol{H}(\mathrm{curl})$ and $\boldsymbol{H}(\mathrm{div})$ has been noticed only recently – first probably by Bossavit [33]. In addition to its role in the design of vector-valued finite elements, the diagram forms a mathematical foundation for stability and convergence analysis for Maxwell's equations, problems of acoustics and various mixed formulations. In particular, there is a strong connection between the good behavior of edge and face elements and the commuting properties of the diagram. The diagram has the form

$$H^1 \xrightarrow{\nabla} \boldsymbol{H}(\mathrm{curl}) \xrightarrow{\nabla\times} L^2, \tag{2.1}$$

or, alternatively,

$$H^1 \xrightarrow{\nabla\times} \boldsymbol{H}(\mathrm{div}) \xrightarrow{\nabla\cdot} L^2, \tag{2.2}$$

and extends to

$$H^1 \xrightarrow{\nabla} \boldsymbol{H}(\mathrm{curl}) \xrightarrow{\nabla\times} \boldsymbol{H}(\mathrm{div}) \xrightarrow{\nabla\cdot} L^2 \tag{2.3}$$

in three spatial dimensions.

REMARK 2.1 (Operator ∇) We use the operator ∇ (nabla) in the standard sense,

$$\nabla = \nabla_x = \left(\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_d}\right)^T, \tag{2.4}$$

where d is the spatial dimension. Recall that

- the inner product of nabla with a vector function $\boldsymbol{v}$ (denoted by $\nabla \cdot \boldsymbol{v}$) yields the *divergence* of $\boldsymbol{v}$ (a scalar quantity),
- ∇ applied to a scalar function v yields its *gradient* (a vector quantity),
- the cross product of nabla with a vector function $\boldsymbol{v}$ (denoted by $\nabla \times \boldsymbol{v}$) yields its *curl* (a vector in 3D and a scalar $\partial v_2/\partial x_1 - \partial v_1/\partial x_2$ in 2D).

In addition, in 2D we also define the vector-valued **curl** of a scalar function v as $\mathbf{curl} = \nabla \times v = (-\partial v/\partial \xi_2, \partial v/\partial \xi_1)$. $\square$

In the diagram, the range of each of the operators exactly coincides with the null space of the next operator in the sequence, and the last map is a surjection. All the above versions of the diagram can be restricted to functions satisfying the homogeneous Dirichlet conditions. In reality the scheme is even more complex, relating the exact sequences of spaces H^1, $\boldsymbol{H}(\mathrm{curl})$, $\boldsymbol{H}(\mathrm{div})$ and L^2 on both continuous and discrete levels by means of appropriate *interpolation operators* Π^1, Π^{curl}, Π^{div} and P (P is simply an L^2-projection), which we will introduce later in Section 3.1.

Recent results by Demkowicz and others [61, 67, 59, 68] show that finite elements have to be understood in a more general sense, as *sequences of scalar and vector-valued elements* satisfying the De Rham diagram on the discrete level. Commutativity of the diagram between the continuous and discrete levels therefore has an essential influence on stability of finite element discretizations in vector-valued spaces.

The De Rham diagram will be discussed in more detail in Chapter 3, when all the necessary machinery is in place. However, the differential operators from the diagram will be used already in Sections 2.3 and 2.4 to design higher-order shape functions for $\boldsymbol{H}(\mathrm{curl})$- and $\boldsymbol{H}(\mathrm{div})$-conforming finite elements.

2.2 H^1-conforming approximations

The construction of finite elements of arbitrary order for H^1-conforming approximations is relatively well known, and various options of hierarchic shape functions for all commonly used reference domains can be found in several textbooks (see, e.g., [18, 122, 191]) and numerous articles ([8, 20, 22, 15, 19, 21, 67, 62, 66, 64, 162, 185] and others). However, the question of the *optimal design* of shape functions is extremely difficult (already the formulation of optimality criteria is not at all trivial), and very few results stating any kind of optimality are available. The conditioning of the master element stiffness and/or mass matrix is a good indicator of quality of the shape functions, and we will adopt the same approach in our case as well.

Another reason why we revisit the construction of scalar hierarchic shape functions once again is that they play an important role in several parts of the higher-order finite element technology – they will facilitate the design of vector-valued finite elements of arbitrary order in spaces $\boldsymbol{H}(\mathrm{curl})$ and $\boldsymbol{H}(\mathrm{div})$, we will exploit them in Chapter 3 to give a comprehensive definition of *projection-based interpolation operators* on *hp*-meshes, they will be used for the construction of *reference maps*, etc. Moreover, we find it useful to provide the reader with some graphic and geometric intuition, which standard journal papers do not usually include.

2.2.1 One-dimensional master element $\mathcal{K}_a^1$

Let us recall the one-dimensional reference interval $K_a = (-1, 1)$ we dealt with in Section 1.3. In this case the only relevant local order of approximation is the order $1 \le p^b$ in element interior. The master element $\mathcal{K}_a^1 = (K_a, W_a, \Sigma_a^1)$ will be equipped with polynomial space

$$W_a = \mathcal{P}_{p^b}(K_a). \tag{2.5}$$

By $\mathcal{P}_p(e)$ we denote the space of polynomials of the order of at most p, defined on a one-dimensional interval e. The hierarchic basis in W_a consists of *vertex functions*

$$\begin{aligned} \varphi_a^{v_1}(\xi) = \lambda_{2,a}(\xi) &= l_0(\xi), \\ \varphi_a^{v_2}(\xi) = \lambda_{1,a}(\xi) &= l_1(\xi), \end{aligned} \tag{2.6}$$

where $\lambda_{1,a}$ and $\lambda_{2,a}$ are one-dimensional affine coordinates, and *bubble functions*

$$\varphi_{k,a}^b = l_k, \quad 2 \le k \le p^b \tag{2.7}$$

that were defined in (1.49). For future reference let us mention that the bubble functions can be written in the form

$$\varphi_{k,a}^b = \lambda_{1,a}\lambda_{2,a}\phi_{k-2}(\lambda_{1,a} - \lambda_{2,a}), \quad k = 2, 3, \ldots, p^b, \tag{2.8}$$

using the kernel functions $\phi_0, \phi_1, \ldots$ defined in (1.52). The latter version will extend more naturally to triangles and tetrahedra, while (2.7) will be more suitable for quadrilaterals and bricks. Prisms will require a combination of both.

PROPOSITION 2.1
Both sets of functions defined in (2.6), (2.8), and in (2.7) represent a hierarchic basis for the space W_a, defined in (2.5).

PROOF The Lobatto shape functions $l_0, l_1, \dots$ are linearly independent and their number is equal to the dimension of the polynomial space. $\square$

2.2.2 Quadrilateral master element $\mathcal{K}_q^1$

In this paragraph we will design a master element of arbitrary order $\mathcal{K}_q^1$ on the reference quadrilateral domain

$$K_q = \{\boldsymbol{\xi} \in \mathbb{R}^2; -1 < \xi_1, \xi_2 < 1\}, \tag{2.9}$$

depicted in Figure 2.1.

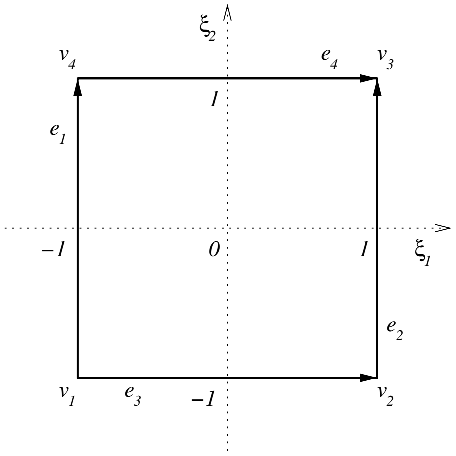

FIGURE 2.1: The reference quadrilateral K_q.

The reason for the choice (2.9) is that $[-1, 1]$ is the natural interval of definition of Jacobi polynomials. We will use one-dimensional affine coordinates $\lambda_{j,q}$, $j = 1, \dots, 4$ of the form

$$\begin{aligned} \lambda_{1,q}(\xi_1, \xi_2) &= \frac{\xi_1 + 1}{2}, \ \lambda_{2,q}(\xi_1, \xi_2) = \frac{1 - \xi_1}{2}, \\ \lambda_{3,q}(\xi_1, \xi_2) &= \frac{\xi_2 + 1}{2}, \ \lambda_{4,q}(\xi_1, \xi_2) = \frac{1 - \xi_2}{2}. \end{aligned} \tag{2.10}$$

To allow for *anisotropic p-refinement* of quadrilateral elements, we consider two local directional polynomial orders of approximation $p^{b,1}, p^{b,2}$ in element interior, corresponding to axial directions ξ_1 and ξ_2, respectively. The edges $e_1, \dots, e_4$ will be assigned local orders of approximation $p^{e_1}, \dots, p^{e_4}$.

The local orders of approximation $p^{b,1}, p^{b,2}, p^{e_1}, \dots, p^{e_4}$ originate in the physical mesh, where they obey the *minimum rule* for H^1-conforming approximations (the polynomial order assigned to an edge e in the physical mesh is equal to the *minimum* of appropriate directional orders in the interior of adjacent elements).

REMARK 2.2 (Minimum rule for H^1-conforming approximations) The *minimum rule* splits the global piecewise-polynomial space $V_{h,p}$ from the discrete variational formulation (1.22) into a set of local polynomial spaces of order $p_i = p(K_i)$ on all finite elements $K_i \in \mathcal{T}_{h,p}$. The polynomial order of these subspaces coincides with the order of approximation in element interiors. This is essential, since only in this way can one speak about local polynomial spaces on finite elements *independently of a concrete choice of shape functions.* Let us be satisfied with this brief motivation for now – we will return to the minimum rules in more detail in Paragraph 3.5.5. $\square$

On the reference domain the minimum rule imposes that

$$p^{e_1}, p^{e_2} \le p^{b,2} \quad \text{and} \quad p^{e_3}, p^{e_4} \le p^{b,1}.$$

These local orders of approximation determine that a finite element of the form $\mathcal{K}_q^1 = (K_q, W_q, \Sigma_q^1)$ has to be equipped with a polynomial space

$$W_q = \left\{ w \in \mathcal{Q}_{p^{b,1}, p^{b,2}};\ w|_{e_j} \in \mathcal{P}_{p^{e_j}}(e_j), j = 1, \dots, 4 \right\}, \tag{2.11}$$

where

$$\mathcal{Q}_{p,q} = \operatorname{span}\left\{ \xi_1^i \xi_2^j;\ \ (\xi_1, \xi_2) \in K_q,\ i = 0, \dots, p, j = 0, \dots, q \right\}. \tag{2.12}$$

The set of degrees of freedom Σ_q^1 will be uniquely identified by the choice of a basis in W_q.

REMARK 2.3 (Conformity requirements and structure of shape functions) The space H^1 imposes the most severe conformity requirements – global continuity of approximation. This constrains the values of approximation at both the element vertices and on the edges. Only functions that vanish entirely on the element boundary are unconstrained (interior). Therefore, the hierarchic basis of W_q will be composed of *vertex*, *edge* and *bubble* functions. $\square$

Vertex functions $\varphi_q^{v_1}, \ldots, \varphi_q^{v_4}$ are assigned to vertices $v_1, \ldots, v_4$: $\varphi_q^{v_j}$ is equal to one at v_j and it vanishes at all remaining vertices. These functions are chosen bilinear in the form

$$\begin{aligned} \varphi_q^{v_1}(\xi_1, \xi_2) &= l_0(\xi_1) l_0(\xi_2), \\ \varphi_q^{v_2}(\xi_1, \xi_2) &= l_1(\xi_1) l_0(\xi_2), \\ \varphi_q^{v_3}(\xi_1, \xi_2) &= l_1(\xi_1) l_1(\xi_2), \\ \varphi_q^{v_4}(\xi_1, \xi_2) &= l_0(\xi_1) l_1(\xi_2), \end{aligned} \tag{2.13}$$

as illustrated in Figure 2.2.

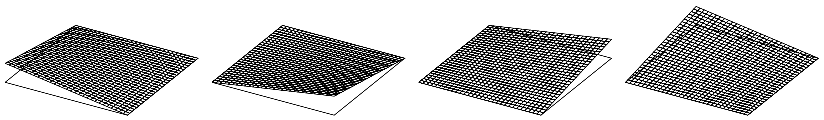

FIGURE 2.2: Vertex functions $\varphi_q^{v_1}, \ldots, \varphi_q^{v_4}$.

Next we add to the basis of W_q *edge functions* $\varphi_{k,q}^{e_j}$, $k = 2, \ldots, p^{e_j}$, $j = 1, \ldots, 4$. Each function $\varphi_{k,q}^{e_j}$ is associated with the corresponding (oriented) edge e_j in such a way that a) its trace on e_j exactly coincides with the Lobatto shape function l_k and b) its trace vanishes on all remaining edges. We define them as

$$\begin{aligned} \varphi_{k,q}^{e_1}(\xi_1, \xi_2) &= l_0(\xi_1) l_k(\xi_2), \quad 2 \le k \le p^{e_1}, \\ \varphi_{k,q}^{e_2}(\xi_1, \xi_2) &= l_1(\xi_1) l_k(\xi_2), \quad 2 \le k \le p^{e_2}, \\ \varphi_{k,q}^{e_3}(\xi_1, \xi_2) &= l_k(\xi_1) l_0(\xi_2), \quad 2 \le k \le p^{e_3}, \\ \varphi_{k,q}^{e_4}(\xi_1, \xi_2) &= l_k(\xi_1) l_1(\xi_2), \quad 2 \le k \le p^{e_4}. \end{aligned} \tag{2.14}$$

REMARK 2.4 (Decoupling of polynomial orders) This remark is related to Remark 2.2. Notice that the order of edge functions is limited by the local orders of approximation $p^{e_1}, \ldots, p^{e_4}$ assigned to edges (i.e., neither by $p^{b,1}$ nor by $p^{b,2}$). Hence the hierarchic structure of shape functions decouples the orders of polynomial approximation in the element interior and on the edges, and adjacent finite elements in the mesh will be allowed to coexist with different polynomial orders. $\square$

Edge functions present in the basis of the space W_q when all $p^{e_1} = \ldots = p^{e_4} = 6$ are depicted in Figures 2.3 – 2.7.

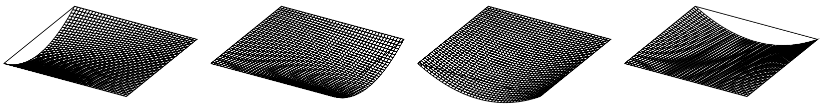
FIGURE 2.3: Quadratic edge functions $\varphi^{e_1}_{2,q}, \ldots, \varphi^{e_4}_{2,q}$.

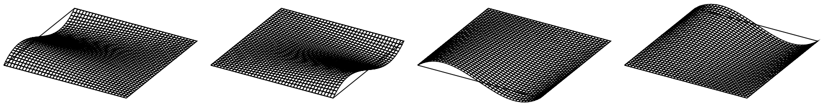
FIGURE 2.4: Cubic edge functions $\varphi^{e_1}_{3,q}, \ldots, \varphi^{e_4}_{3,q}$.

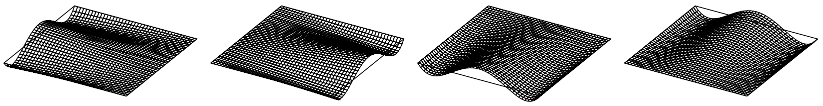
FIGURE 2.5: Fourth-order edge functions $\varphi^{e_1}_{4,q}, \ldots, \varphi^{e_4}_{4,q}$.

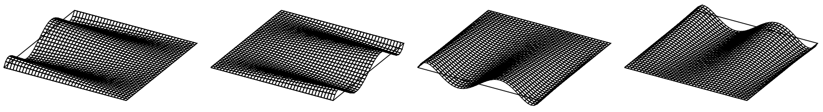
FIGURE 2.6: Fifth-order edge functions $\varphi^{e_1}_{5,q}, \ldots, \varphi^{e_4}_{5,q}$.

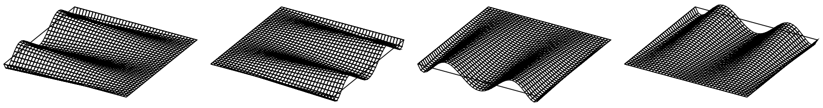
FIGURE 2.7: Sixth-order edge functions $\varphi^{e_1}_{6,q}, \ldots, \varphi^{e_4}_{6,q}$.

REMARK 2.5 The present choice of orientation of edges, following [64], is advantageous since it minimizes the number of sign factors in the formulae for edge functions. However, any choice of orientation of edges would be equally good from the point of view of computer implementation. ▯

REMARK 2.6 In Chapter 3 we will introduce *reference maps*, which geometrically relate the reference element with quadrilaterals in the physical mesh. Each physical mesh edge will then be assigned a unique orientation, and all edges of physical mesh quadrilaterals will be equipped with an *orientation flag*, indicating whether the image of the corresponding edge of the reference domain through the reference map has the same or opposite orientation. Orientation issues both in 2D and 3D will be discussed in more detail in Chapter 3. ▯

The hierarchic basis of W_q will be completed by adding *bubble functions*

$$\varphi^b_{n_1,n_2,q}(\xi_1,\xi_2) = l_{n_1}(\xi_1)l_{n_2}(\xi_2), \quad 2 \le n_1 \le p^{b,1},\ 2 \le n_2 \le p^{b,2}, \tag{2.15}$$

that vanish everywhere on the boundary of the reference domain. A few examples of bubble functions are shown in Figures 2.8 – 2.12.

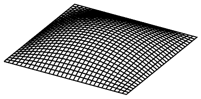

FIGURE 2.8: Quadratic bubble function $\varphi^b_{2,2,q}$.

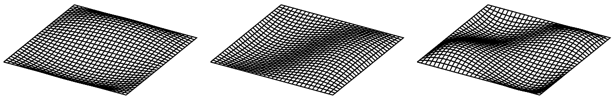

FIGURE 2.9: Cubic bubble functions $\varphi^b_{2,3,q}$, $\varphi^b_{3,2,q}$, $\varphi^b_{3,3,q}$.

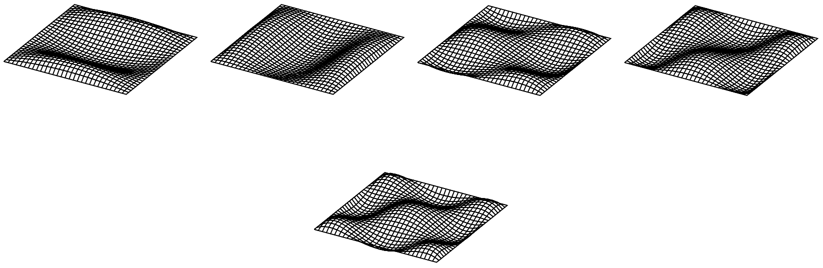

FIGURE 2.10: Fourth-order bubble functions $\varphi^b_{2,4,q}$, $\varphi^b_{4,2,q}$, $\varphi^b_{3,4,q}$, $\varphi^b_{4,3,q}$, $\varphi^b_{4,4,q}$.

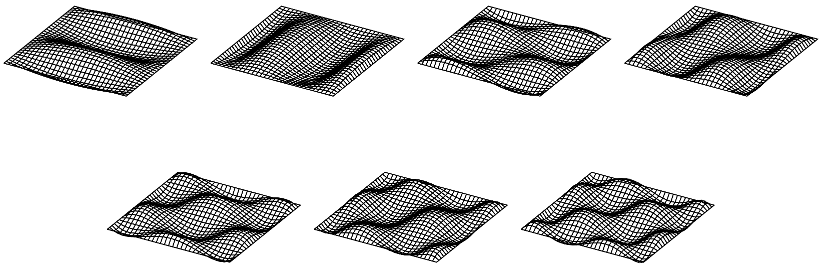

FIGURE 2.11: Fifth-order bubble functions $\varphi^b_{2,5,q}$, $\varphi^b_{5,2,q}$, $\varphi^b_{3,5,q}$, $\varphi^b_{5,3,q}$, $\varphi^b_{4,5,q}$, $\varphi^b_{5,4,q}$, $\varphi^b_{5,5,q}$.

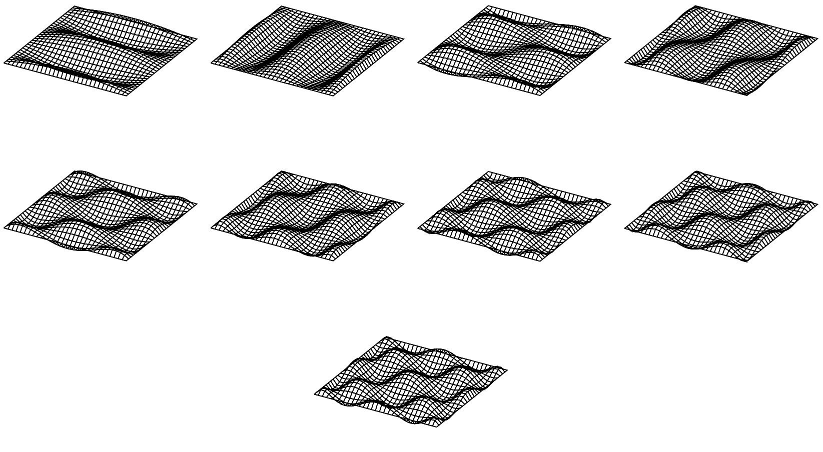

FIGURE 2.12: Sixth-order bubble functions $\varphi^b_{2,6,q}$, $\varphi^b_{6,2,q}$, $\varphi^b_{3,6,q}$, $\varphi^b_{6,3,q}$, $\varphi^b_{4,6,q}$, $\varphi^b_{6,4,q}$, $\varphi^b_{5,6,q}$, $\varphi^b_{6,5,q}$, $\varphi^b_{6,6,q}$.

Numbers of scalar hierarchic shape functions associated with the master quadrilateral $\mathcal{K}_q^1$ are summarized in Table 2.1.

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

Node type	**Polynomial order**	**Number of shape functions**	**Number of nodes**
Vertex	always	1	4
Edge	$2 \le p^{e_j}$	$p^{e_j} - 1$	4
Interior	$2 \le p^{b,1}, p^{b,2}$	$(p^{b,1} - 1)(p^{b,2} - 1)$	1

PROPOSITION 2.2
Shape functions (2.13), (2.14) and (2.15) constitute a hierarchic basis of the space W_q, defined in (2.11).

PROOF Although this is a very simple case, let us introduce a proof for future reference. All functions (2.13), (2.14) and (2.15) are obviously linearly independent. Consider a function $u \in W$. It is our aim to find a unique set of coefficients identifying u as a linear combination of these functions. First, construct vertex interpolant

$$u^v = \sum_{i=1}^{4} \alpha^{v_i} \varphi_q^{v_i},$$

such that $u^v(v_i) = u(v_i)$ for all $i = 1, \dots, 4$. The coefficients α^{v_i} are unique. Obviously, $(u - u^v)$ vanishes at all vertices, and its trace $(u - u^v)|_{e_j} \in \mathcal{P}_{p^{e_j}}(e_j)$ for all $j = 1, \dots, 4$.

Next construct edge interpolants

$$u^{e_j} = \sum_{k=2}^{p^{e_j}} \alpha_{k,q}^{e_j} \varphi_{k,q}^{e_j},$$

such that $u^{e_j}|_{e_j} = (u - u^v)|_{e_j}$, for all $j = 1, \dots, 4$. This is easy because traces of edge functions $\varphi_{k,q}^{e_j}$, $2 \le k \le p^{e_j}$ generate the space $\mathcal{P}_{p^{e_j},0}(e_j)$ (space of one-dimensional polynomials vanishing at endpoints of e_j). The coefficients $\alpha_{k,q}^{e_j}$ are again unique for all $k = 2, \dots, p^{e_j}$, $j = 1, \dots, 3$. Define the final edge interpolant u^e, summing up edge contributions,

$$u^e = \sum_{j=1}^{4} u^{e_j}.$$

The difference $u - u^v - u^e$ vanishes on all edges. The definition (2.11) of the space W_q implies that $u - u^v - u^e$ can be uniquely expressed in terms of bubble functions $\varphi^b_{n_1,n_2,q}$,

$$u - u^v - u^e = \sum_{n_1=2}^{p^{b,1}} \sum_{n_2=2}^{p^{b,2}} \alpha^b_{n_1,n_2,q} \varphi^b_{n_1,n_2,q},$$

which concludes the proof. $\square$

REMARK 2.7 When the distribution of order of polynomial approximation in the finite element mesh is *uniform*, we have $p = p^{b,1} = p^{b,2} = p^{e_1} = \ldots = p^{e_4}$, and the above introduced basis of W_q reduces to a basis of the standard space $\mathcal{Q}_{p,p}$: $l_i(\xi_1)l_j(\xi_2),\ 0 \le i, j \le p$, of cardinality $\mathrm{card}(\mathcal{Q}_{p,p}) = (p+1)^2$. $\square$

REMARK 2.8 It is easy to write all shape functions in terms of affine coordinates $\lambda_{1,q}, \ldots, \lambda_{4,q}$, introduced in (2.10), using the transformation

$$\xi_1 = \lambda_{1,q} - \lambda_{2,q},\ \xi_2 = \lambda_{3,q} - \lambda_{4,q}.$$

The advantage of shape functions formulated in terms of affine coordinates rather than by means of spatial variables is that they are invariant with respect to affine transformations of the reference domain. In other words, we do not need to change the shape functions when adjusting the reference geometry. In the case of product geometries (quadrilateral, brick) this applies to one-dimensional affine changes in the axial directions only. This aspect becomes more strongly pronounced in the case of n-simplices (in our case triangles, tetrahedra). $\square$

2.2.3 Triangular master element $\mathcal{K}^1_t$

Next in our series of master elements of arbitrary order, $\mathcal{K}^1_t$, will be associated with the reference triangular domain

$$K_t = \{\boldsymbol{\xi} \in \mathbb{R}^2;\ -1 < \xi_1, \xi_2;\ \xi_1 + \xi_2 < 0\}, \tag{2.16}$$

shown in Figure 2.13.

REMARK 2.9 Although other reasonable choices exist (e.g., triangles $[0,0]$, $[1,0], [0,1]$ and $[-1,0], [1,0], [0, \sqrt{3}/2]$), we prefer (2.16) since it respects the interval of definition of the Jacobi polynomials in both axial directions. The fact that the geometry contains two edges which are perpendicular to each other will be advantageous in the $\boldsymbol{H}$(curl)- and $\boldsymbol{H}$(div)-conforming cases, where tangential and normal vectors to edges are part of vector-valued shape functions. $\square$

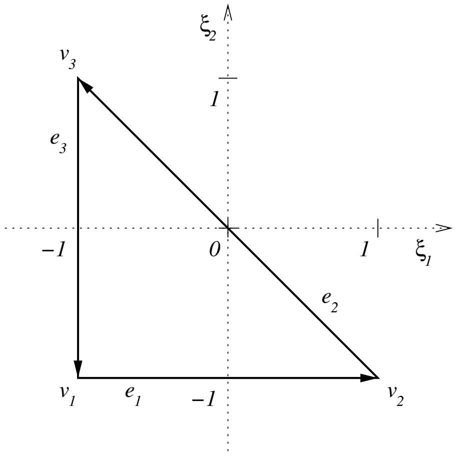

FIGURE 2.13: The reference triangle K_t.

The reference geometry (2.16) is equipped with affine coordinates

$$\lambda_{1,t}(\xi_1, \xi_2) = \frac{\xi_2 + 1}{2}, \ \lambda_{2,t}(\xi_1, \xi_2) = -\frac{\xi_1 + \xi_2}{2}, \ \lambda_{3,t}(\xi_1, \xi_2) = \frac{\xi_1 + 1}{2}. \quad (2.17)$$

Anisotropic p-refinement of triangular elements has no practical application, and therefore we consider one local order of approximation p^b in the element interior only. Edges $e_1, \ldots, e_3$, will be assigned local polynomial orders $p^{e_1}, \ldots, p^{e_3}$. Again, these nonuniform local orders of approximation originate in the physical mesh, where they have to obey the minimum rule for H^1-conforming approximations (Remark 2.2), which on the reference domain translates into

$$p^{e_j} \le p^b, \quad j = 1, \ldots, 3.$$

The local orders $p^b, p^{e_1}, \ldots, p^{e_3}$, suggest that a finite element of the form $\mathcal{K}_t^1 = (K_t, W_t, \Sigma_t^1)$ should carry a polynomial space

$$W_t = \left\{ w \in \mathcal{P}_{p^b}(K_t); \ w|_{e_j} \in \mathcal{P}_{p^{e_j}}(e_j), \ j = 1, \ldots, 3 \right\}, \quad (2.18)$$

where

$$\mathcal{P}_p(K_t) = \operatorname{span}\left\{ \xi_1^i \xi_2^j; \ (\xi_1, \xi_2) \in K_t; \ i, j = 0, \ldots, p; \ i + j \le p \right\}. \quad (2.19)$$

Again the set of degrees of freedom Σ_t^1 will be uniquely identified by a choice of basis of the space W_t. Hierarchic basis of W_t will consist of *vertex*, *edge* and *bubble* functions.

Vertex functions $\varphi_t^{v_1}, \dots, \varphi_t^{v_3}$ are assigned to vertices $v_1, \dots, v_3$: $\varphi_t^{v_j}$ is equal to one at v_j and it vanishes at the remaining two vertices. These functions are chosen linear in the form illustrated in Figure 2.14,

$$\begin{aligned} \varphi_t^{v_1}(\xi_1, \xi_2) &= \lambda_{2,t}(\xi_1, \xi_2), \\ \varphi_t^{v_2}(\xi_1, \xi_2) &= \lambda_{3,t}(\xi_1, \xi_2), \\ \varphi_t^{v_3}(\xi_1, \xi_2) &= \lambda_{1,t}(\xi_1, \xi_2). \end{aligned} \tag{2.20}$$

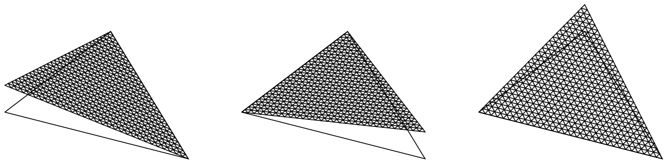

FIGURE 2.14: Vertex functions $\varphi_t^{v_1}, \dots, \varphi_t^{v_3}$.

Next we define *edge functions* $\varphi_{k,q}^{e_j}$, $k = 2, \dots, p^{e_j}$, $j = 1, \dots, 3$. Again, traces of edge functions $\varphi_{k,q}^{e_j}$, $k = 2, \dots, p^{e_j}$, will coincide with the Lobatto shape functions $l_2, l_3, \dots$ on the edge e_j, and vanish on all remaining edges. We can write them in the form

$$\begin{aligned} \varphi_{k,t}^{e_1} &= \lambda_{2,t}\lambda_{3,t}\phi_{k-2}(\lambda_{3,t} - \lambda_{2,t}), \quad 2 \le k \le p^{e_1}, \\ \varphi_{k,t}^{e_2} &= \lambda_{3,t}\lambda_{1,t}\phi_{k-2}(\lambda_{1,t} - \lambda_{3,t}), \quad 2 \le k \le p^{e_2}, \\ \varphi_{k,t}^{e_3} &= \lambda_{1,t}\lambda_{2,t}\phi_{k-2}(\lambda_{2,t} - \lambda_{1,t}), \quad 2 \le k \le p^{e_3}. \end{aligned} \tag{2.21}$$

REMARK 2.10 Here we arrive at a point where the definition of the kernel functions $\phi_0, \phi_1, \dots$ from (1.52) is motivated – they make the edge functions coincide exactly with the Lobatto shape functions $l_2, l_3, \dots$ on the edges, and keep them constant along lines parallel to the line connecting the edge-midpoint with the opposite vertex. The product of the two vertex functions keeps the Lobatto shape functions zero on the remaining edges. $\square$

A few examples of edge functions are shown in Figures 2.15 – 2.19.

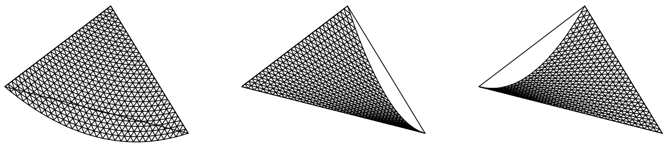

FIGURE 2.15: Quadratic edge functions $\varphi_{2,t}^{e_1}, \dots, \varphi_{2,t}^{e_3}$.

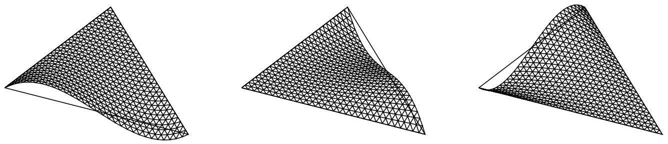

FIGURE 2.16: Cubic edge functions $\varphi_{3,t}^{e_1}, \dots, \varphi_{3,t}^{e_3}$.

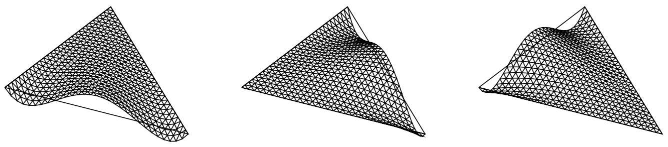

FIGURE 2.17: Fourth-order edge functions $\varphi_{4,t}^{e_1}, \dots, \varphi_{4,t}^{e_3}$.

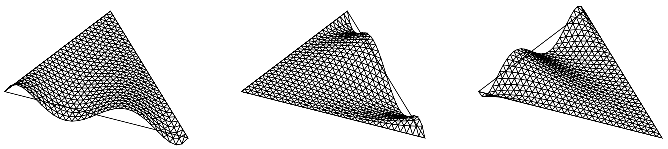

FIGURE 2.18: Fifth-order edge functions $\varphi_{5,t}^{e_1}, \dots, \varphi_{5,t}^{e_3}$.

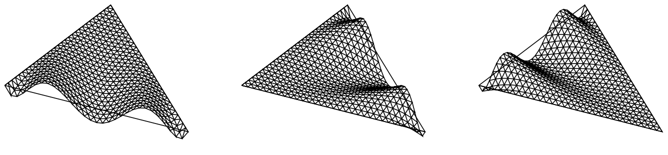

FIGURE 2.19: Sixth-order edge functions $\varphi_{6,t}^{e_1}, \dots, \varphi_{6,t}^{e_3}$.

REMARK 2.11 Traces of vertex and edge functions, associated with the master elements $\mathcal{K}_q^1$ and $\mathcal{K}_t^1$, coincide in both cases with the functions $l_0, l_1, l_2, \ldots$ from (1.49). This allows for the combination of quadrilateral and triangular elements in *hybrid meshes* in Chapter 3. $\square$

The hierarchic basis of W_t will be completed by defining *bubble functions* that vanish entirely on the element boundary. These functions are internal, and their choice does not affect the compatibility of triangular elements with other element types in hybrid meshes. A standard approach is to simply combine affine coordinates with varying powers,

$$\varphi^b_{n_1,n_2,t} = \lambda_{1,t}(\lambda_{2,t})^{n_1}(\lambda_{3,t})^{n_2}, \quad 1 \le n_1, n_2;\ n_1 + n_2 \le p^b - 1. \qquad (2.22)$$

These bubble functions, up to the sixth order, are shown in Figures 2.20 – 2.21.

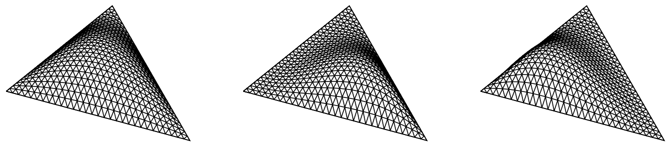

FIGURE 2.20: Standard cubic bubble function $\varphi^b_{1,1,t}$, given by (2.22), and fourth-order bubble functions $\varphi^b_{1,2,t}$ and $\varphi^b_{2,1,t}$ from (2.22).

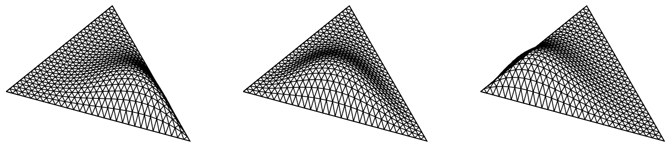

FIGURE 2.21: Standard fifth-order bubble functions $\varphi^b_{1,3,t}$, $\varphi^b_{2,2,t}$ and $\varphi^b_{3,1,t}$ given by (2.22).

REMARK 2.12 The reader may notice that bubble functions (2.22) are constructed in a similar way as the simple hierarchic shape functions (1.85) in 1D. Their conditioning properties are similarly bad, as we show in Figure 2.25. $\square$

This fact motivates us to define a new set of bubble functions

$$\varphi^b_{n_1,n_2,t} = \lambda_{1,t}\lambda_{2,t}\lambda_{3,t}\phi_{n_1-1}(\lambda_{3,t} - \lambda_{2,t})\phi_{n_2-1}(\lambda_{2,t} - \lambda_{1,t}), \tag{2.23}$$

$1 \le n_1, n_2;\ n_1 + n_2 \le p^b - 1$. We use the kernel functions (1.52) in order to incorporate the Lobatto shape functions into their shape. The bubble functions (2.23) are depicted in Figures 2.22 – 2.24, and the conditioning properties of the two mentioned sets of bubble functions are compared in Figure 2.25.

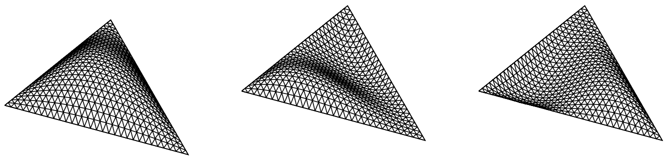

FIGURE 2.22: New cubic bubble function (same as standard) and new fourth-order bubble functions (2.23) with improved conditioning properties.

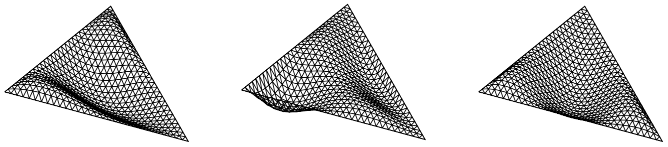

FIGURE 2.23: New fifth-order bubble functions (2.23) with improved conditioning properties.

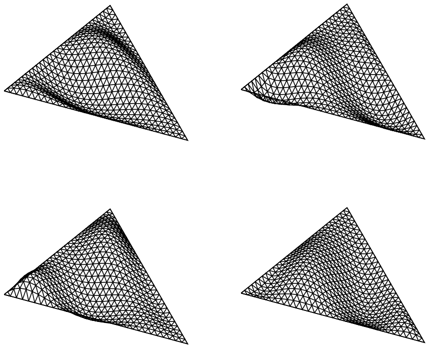

FIGURE 2.24: New sixth-order bubble functions (2.23) with improved conditioning properties.

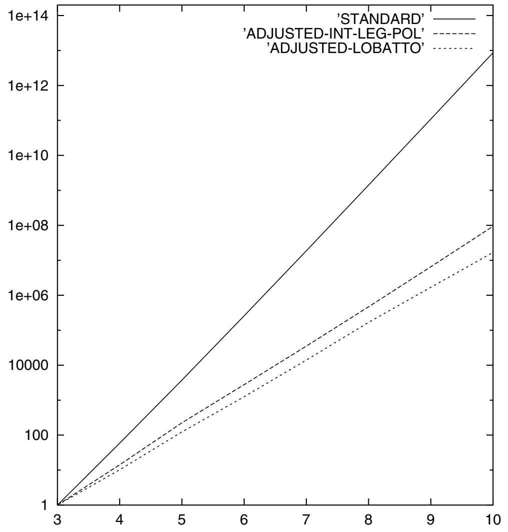

FIGURE 2.25: Conditioning of master element stiffness matrix (of the Laplace operator) for the standard bubble functions (2.22) and the new bubble functions (2.23) in decimal logarithmic scale. The curve in between corresponds to the new bubble functions, using integrated Legendre polynomials instead of the Lobatto shape functions (i.e., with the normalization constant in (1.49) neglected). The curve indicates that the role of the normalization becomes significant as the polynomial order grows.

REMARK 2.13 For future reference, let us mention that the bubble functions (2.23) can also be viewed as *oriented.* They can be written as

$$\varphi^b_{n_1,n_2,t} = \lambda_A \lambda_B \lambda_C \phi_{n_1-1}(\lambda_B - \lambda_A)\phi_{n_2-1}(\lambda_A - \lambda_C),$$

$1 \leq n_1, n_2;\ n_1 + n_2 \leq p^b - 1$, where $\lambda_A, \lambda_B, \lambda_C$ are affine coordinates, ordered in such a way that $\lambda_A(v_1) = \lambda_B(v_2) = \lambda_C(v_3) = 1$. Such orientation will be imposed to triangular faces in 3D, in order to facilitate the construction of globally conforming basis functions in physical tetrahedral and hybrid tetrahedral/prismatic meshes. Algorithmic treatment of orientation information will be discussed in more detail in Chapter 3. $\square$

Table 2.2 quantifies numbers of hierarchic shape functions in the basis of the space W_t.

PROPOSITION 2.3
Shape functions (2.20), (2.21) and (2.23) provide a hierarchic basis of the space W_t, defined in (2.18).

TABLE 2.2: Scalar hierarchic shape functions of $\mathcal{K}_t^1$.

Node type	Polynomial order	Number of shape functions	Number of nodes
Vertex	always	1	3
Edge	$2 \le p^{e_j}$	$p^{e_j} - 1$	3
Interior	$3 \le p^b$	$(p^b - 1)(p^b - 2)/2$	1

PROOF The proof is very similar to the previous quadrilateral case. Verify

1. that all shape functions are linearly independent,
2. that they all belong to the space W_t,
3. and finally that their number matches the dimension of the space W_t.

This accomplishes the proof. □

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

The first three-dimensional master element of arbitrary order, $\mathcal{K}_B^1$, will be associated with the reference brick domain

$$K_B = \{\boldsymbol{\xi} \in \mathbb{R}^3;\ -1 < \xi_1, \xi_2, \xi_3 < 1\}, \tag{2.24}$$

depicted in Figure 2.26.

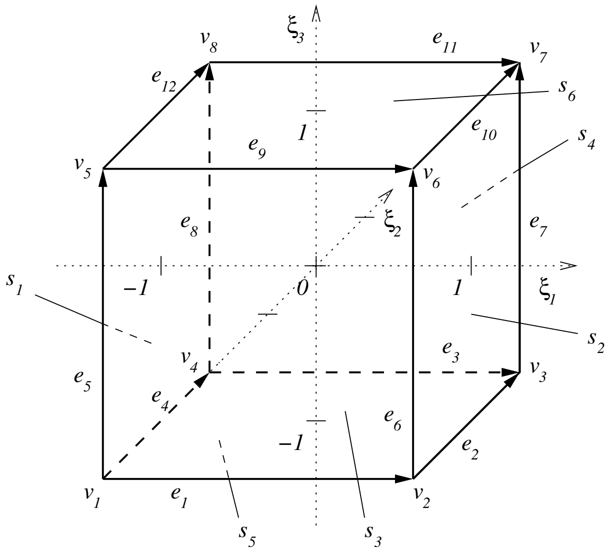

FIGURE 2.26: The reference brick K_B.

REMARK 2.14 This geometry is convenient for our purposes since it respects the interval of definition of the Jacobi polynomials in all three spatial directions. The one-dimensional affine coordinates appropriate for the geometry (2.24) have the form

$$\begin{aligned}\lambda_{1,B}(\xi_1,\xi_2,\xi_3) &= \frac{\xi_1+1}{2},\ \lambda_{2,B}(\xi_1,\xi_2,\xi_3) = \frac{1-\xi_1}{2},\\ \lambda_{3,B}(\xi_1,\xi_2,\xi_3) &= \frac{\xi_2+1}{2},\ \lambda_{4,B}(\xi_1,\xi_2,\xi_3) = \frac{1-\xi_2}{2},\\ \lambda_{5,B}(\xi_1,\xi_2,\xi_3) &= \frac{\xi_3+1}{2},\ \lambda_{6,B}(\xi_1,\xi_2,\xi_3) = \frac{1-\xi_3}{2}.\end{aligned} \tag{2.25}$$

$\square$

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

In 3D there is the added possibility of anisotropic p-refinement of *faces*, for which we need to assign two local directional orders of approximation $p^{s_i,1}, p^{s_i,2}$ to each face s_i, $i = 1, \ldots, 6$. These directional orders are associated with a *local two-dimensional system of coordinates* on each face, which matches an appropriate pair of global coordinate axes in lexicographic order. With this choice, based on [63], local coordinate axes on faces have the same orientation as the corresponding global ones, which simplifies sign-related issues in the formulae for face functions. Edges will be equipped as usual with local orders of approximation $p^{e_1}, \ldots, p^{e_{12}}$, and their orientation will be used for the construction of edge functions only.

REMARK 2.15 (Minimum rules in 3D) In 3D the minimum rule (Remark 2.2) limits the local orders of approximation on both edges and faces. Local (directional) orders on mesh faces are not allowed to exceed the minimum of the (appropriate directional) orders of approximation associated with the interior of the adjacent elements. Local orders of approximation on mesh edges are limited by the minimum of all (appropriate directional) orders corresponding to faces sharing that edge. $\square$

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

$$\begin{aligned}W_B = \big\{ & w \in \mathcal{Q}_{p^{b,1},p^{b,2},p^{b,3}};\ w|_{s_i} \in \mathcal{Q}_{p^{s_i,1},p^{s_i,2}}, w|_{e_j} \in \mathcal{P}_{p^{e_j}}(e_j),\\ & i = 1, \ldots, 6,\ j = 1, \ldots, 12\big\}.\end{aligned} \tag{2.26}$$

Here,

$$\mathcal{Q}_{p,q,r} = \operatorname{span}\left\{\xi_1^i\xi_2^j\xi_2^k;\ (\xi_1,\xi_2,\xi_3) \in K_B,\right. \qquad (2.27)$$
$$\left. i = 0,\dots,p,\ j = 0,\dots,q,\ k = 0,\dots,r\right\}.$$

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

Vertex functions $\varphi_B^{v_j}$, $j = 1,2,\dots,8$, are associated with element vertices, and they provide the complete basis of a space W_B for lowest-order approximation. Recall functions $l_0, l_1, \dots$ from (1.49). Vertex functions will be chosen in a conventional way, i.e., trilinear in the form

$$\varphi_B^{v_j} = l_{d_1}(\xi_1)l_{d_2}(\xi_2)l_{d_3}(\xi_3), \qquad (2.28)$$

where

$$\boldsymbol{d} = (d_1,d_2,d_3) \qquad (2.29)$$

is a vector index, whose components are related to axial directions ξ_1, ξ_2 and ξ_3, respectively. It is defined as follows: consider edges $e_{j_1}, e_{j_2}, e_{j_3}$, containing the vertex v_j, and lying in axial directions ξ_1,ξ_2,ξ_3, respectively. We put $d_k = 0$ if v_j lies on the left of edge e_{j_k} (with respect to the axial direction ξ_k), and $d_k = 1$ otherwise. Notice that the vertex functions $\varphi_B^{v_j}$ are equal to one at the vertex v_j, and vanish at all seven remaining vertices. Their traces are linear on all edges. The construction of vertex functions is illustrated in Figure 2.27.

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

$$\varphi_{k,B}^{e_j} = l_{d_1}(\xi_1)l_{d_2}(\xi_2)l_{d_3}(\xi_3), \qquad (2.30)$$

as illustrated in Figure 2.28.

Face functions $\varphi_{n_1,n_2,B}^{s_i}$, $2 \le n_1 \le p^{s_i,1}$, $2 \le n_2 \le p^{s_i,2}$, corresponding to a face s_i, $i = 1,\dots,6$, will be constructed to have a trace of directional polynomial orders n_1, n_2 on the face s_i (with respect to its local coordinate

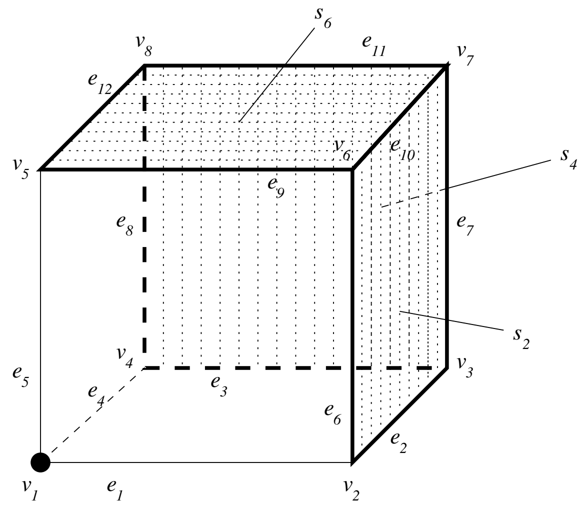

FIGURE 2.27: Vertex function $\varphi_B^{v_1} = l_0(\xi_1)l_0(\xi_2)l_0(\xi_3)$, associated with the vertex v_1 (in this case $\boldsymbol{d} = (0,0,0)$), equals one at v_1. It vanishes completely on the faces s_2, s_4 and s_6, and thus in particular at all remaining vertices as well.

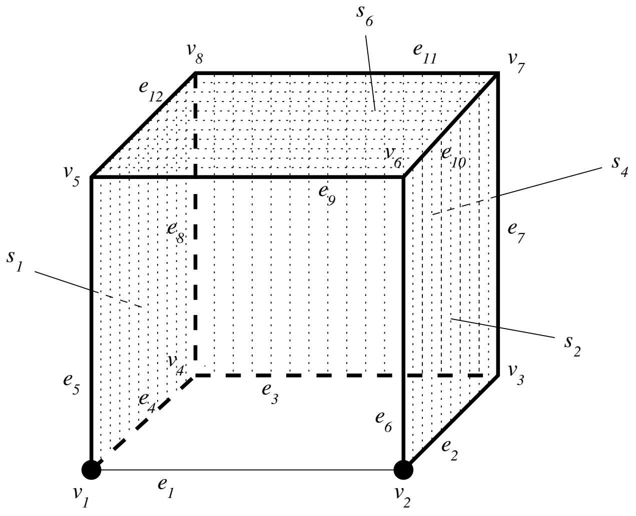

FIGURE 2.28: Edge function $\varphi_{n,B}^{e_1} = l_n(\xi_1)l_0(\xi_2)l_0(\xi_3)$, $n \geq 2$, associated with edge e_1 (in this case $\boldsymbol{d} = (n,0,0)$), coincides with the Lobatto shape function $l_n(\xi_1)$ on the edge e_1. It vanishes completely on faces s_1, s_2, s_4, s_6, and thus in particular also on all remaining edges.

system specified above), and to vanish on the five remaining faces. Appropriate components of the index $\boldsymbol{d} = (d_1, d_2, d_3)$ now contain directional orders n_1, n_2, and the remaining component is set to either zero or one, depending on whether the face s_i lies on the left or right side of the reference brick with respect to the remaining axial direction. We define

$$\varphi^{s_i}_{n_1,n_2,B} = l_{d_1}(\xi_1) l_{d_2}(\xi_2) l_{d_3}(\xi_3), \tag{2.31}$$

and illustrate the construction in Figure 2.29.

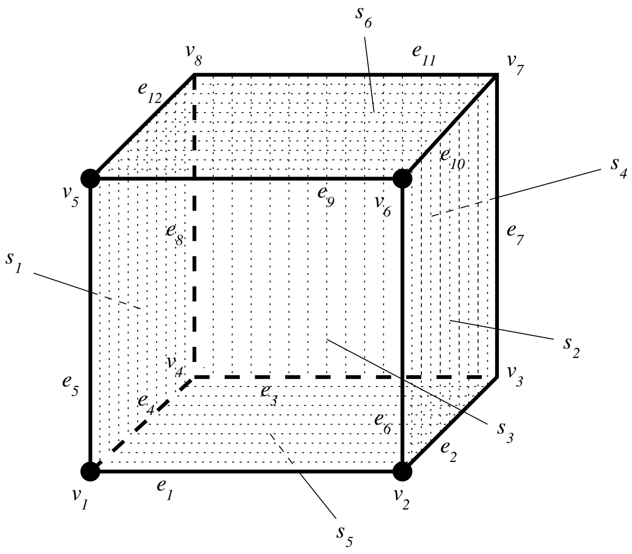

FIGURE 2.29: Consider the face s_3, and local orders of approximation $2 \leq n_1, n_2$, in directions ξ_1, ξ_3. To s_3 we attach a local coordinate system, specified by axial directions ξ_1, ξ_3. In this case it is $\boldsymbol{d} = (n_1, 1, n_2)$. The trace of the face function $\varphi^{s_3}_{n_1,n_2,B}$ is nonzero on the face e_1, and vanishes on all remaining faces (and obviously on all edges and vertices).

REMARK 2.16 Notice that all face functions sharing the same face s_j are linearly independent, and obviously linearly independent of face functions corresponding to other faces. Moreover, all of the aforementioned face functions are linearly independent of edge and vertex functions. $\square$

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

$$\varphi^b_{n_1,n_2,n_3,B} = l_{d_1}(\xi_1)l_{d_2}(\xi_2)l_{d_3}(\xi_3), \quad 2 \leq d_j \leq p^{b,j},\ j = 1, \ldots, 3. \qquad (2.32)$$

Numbers of hierarchic shape functions in the basis of the space W_B are presented in Table 2.3.

TABLE 2.3: Scalar hierarchic shape functions of $\mathcal{K}^1_B$.

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

PROPOSITION 2.4
Shape functions (2.28), (2.30), (2.31) and (2.32) constitute a hierarchic basis of the space W_B, defined in (2.26).

PROOF All functions given by (2.28), (2.30), (2.31) and (2.32) are obviously linearly independent. It is easy to see that they generate the space W_B. Consider a function $u \in W_B$, and express it as a unique linear combination of the basis functions. In other words, begin with constructing a unique vertex interpolant u^v, followed by a unique edge interpolant u^e of $u - u^v$, and then a unique face interpolant u^s of $u - u^v - u^e$. Finally, show that function $u - u^v - u^e - u^s$ can be uniquely expressed by means of bubble functions (2.32). $\square$

REMARK 2.17 When the distribution of order of polynomial approximation in the finite element mesh is *uniform* ($p = p^{b,i} = p^{s_j,k} = p^{e_m}$ for all i, j, k, m), the above introduced basis of W_B reduces to a basis

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

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

All of the above hierarchic shape functions can be expressed in terms of affine coordinates (2.25), using the transformation $\xi_1 = \lambda_{1,B} - \lambda_{2,B}, \xi_2 = \lambda_{3,B} - \lambda_{4,B}, \xi_3 = \lambda_{5,B} - \lambda_{6,B}$. $\square$

2.2.5 Tetrahedral master element $\mathcal{K}_T^1$

Next we design a master element of arbitrary order $\mathcal{K}_T^1$ on the reference tetrahedral domain

$$K_T = \{\boldsymbol{\xi} \in \mathbb{R}^3;\ -1 < \xi_1, \xi_2, \xi_3;\ \xi_1 + \xi_2 + \xi_3 < -1\}, \tag{2.33}$$

shown in Figure 2.30.

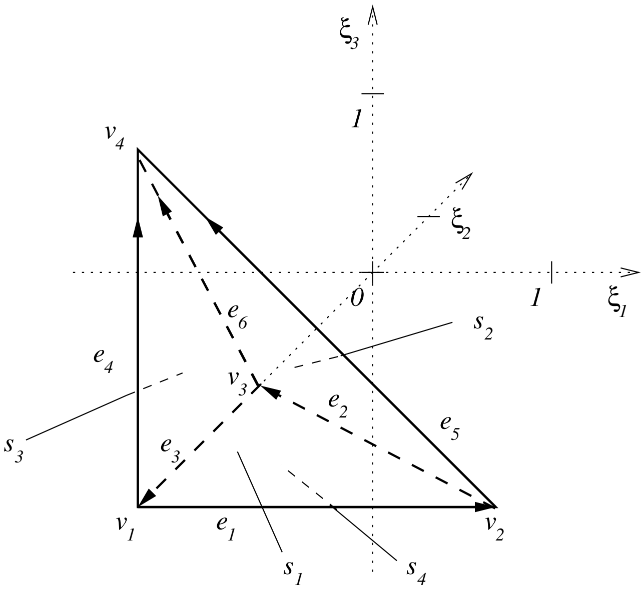

FIGURE 2.30: The reference tetrahedron K_T.

REMARK 2.18 The reference geometry (2.33) is optimal from the point of view that its faces s_2, s_3 and s_4 exactly match the geometry of the reference triangle K_t (Figure 2.13), and, moreover, are perpendicular to each other. Corresponding affine coordinates have the form

$$\begin{aligned}
\lambda_{1,T}(\xi_1,\xi_2,\xi_3) &= \frac{\xi_2+1}{2}, \\
\lambda_{2,T}(\xi_1,\xi_2,\xi_3) &= -\frac{1+\xi_1+\xi_2+\xi_3}{2}, \\
\lambda_{3,T}(\xi_1,\xi_2,\xi_3) &= \frac{\xi_1+1}{2}, \\
\lambda_{4,T}(\xi_1,\xi_2,\xi_3) &= \frac{\xi_3+1}{2}.
\end{aligned} \tag{2.34}$$

$\square$

Consider a local order of approximation p^b in the element interior, local polynomial orders p^{s_i}, $i = 1, \dots, 4$, associated with each face, and standard local orders p^{e_j}, $j = 1, \dots, 6$, associated with each edge. These nonuniform local orders of approximation again have to be compatible with the *minimum rule* for H^1-conforming approximations in 3D introduced in Paragraph 2.2.4.

The local orders $p^b, p^{s_1}, \dots, p^{s_4}, p^{e_1}, \dots, p^{e_4}$, suggest that a finite element of the form $\mathcal{K}_T^1 = (K_T, W_T, \Sigma_T^1)$ will be equipped with polynomial space

$$W_T = \{w \in \mathcal{P}_{p^b}(K_T);\ w|_{s_i} \in \mathcal{P}_{p^{s_i}}(s_i),\ w|_{e_j} \in \mathcal{P}_{p^{e_j}}(e_j), \\ i = 1, \dots, 4,\ j = 1, \dots, 6\}, \tag{2.35}$$

where

$$\mathcal{P}_p(K_T) = \text{span}\left\{\xi_1^i \xi_2^j \xi_3^k;\ i, j, k = 0, \dots, p;\ i + j + k \le p\right\}. \tag{2.36}$$

The set of degrees of freedom Σ_T^1 will be uniquely identified by a choice of basis of the space W_T. Recall affine coordinates $\lambda_{1,T}, \dots, \lambda_{4,T}$ from (2.34), functions $l_0, l_1, \dots$ from (1.49), and kernel functions $\phi_0, \phi_1, \dots$, defined in (1.52). Hierarchic basis of W_T will again comprise *vertex*, *edge*, *face* and *bubble* functions.

Vertex functions $\varphi_T^{v_1}, \dots, \varphi_T^{v_4}$ are associated with the vertices $v_1, \dots, v_4$: $\varphi_T^{v_j}$ is equal to one at the v_j and vanishes at the remaining three vertices. These functions are chosen linear in the form

$$\varphi_T^{v_j} = \lambda_{j_1,T}, \quad j = 1, \dots, 4. \tag{2.37}$$

The index j_1 corresponds to the only face s_{j_1} that does not contain the vertex v_j. Traces of vertex functions are linear both on the reference tetrahedron faces and edges, and their construction is illustrated in Figure 2.31.

Edge functions $\varphi_{k,T}^{e_j}$, $k = 2, \dots, p^{e_j}$, $j = 1, \dots, 6$, appear in the basis of W_T if $2 \le p^{e_j}$, and as usual will be constructed so that their traces match the Lobatto shape functions $l_2, \dots, l_{p^{e_j}}$ on edge e_j, and vanish on the five remaining edges. Orientation of edges will be incorporated into their definition. Let us consider an oriented edge $e_j = v_{i_1} v_{i_2}$. By s_{j_1}, s_{j_2}, denote faces of the reference domain that share with the edge e_j a single vertex v_{i_1} and v_{i_2}, respectively, and define

$$\varphi_{k,T}^{e_j} = \lambda_{j_1,T} \lambda_{j_2,T} \phi_{k-2}(\lambda_{j_1,T} - \lambda_{j_2,T}), \quad 2 \le k \le p^{e_j}. \tag{2.38}$$

The construction of edge functions is illustrated in Figure 2.32.

Face functions associated with faces s_i, $i = 1, \dots, 4$, will be present in the basis of W_T if $3 \le p^{s_i}$. They will be constructed to have nonzero traces of

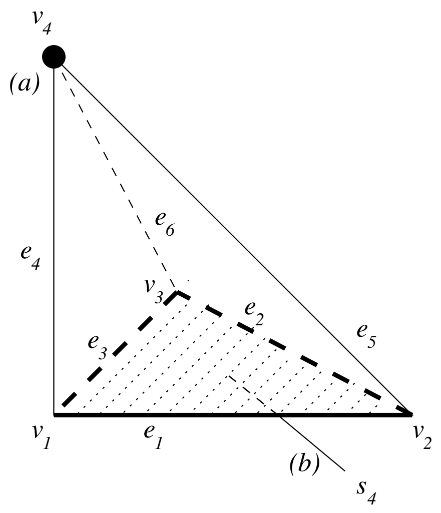

FIGURE 2.31: Vertex function $\varphi_T^{v_4}$ (***a***) is equal to one at the vertex v_4, and (***b***) vanishes everywhere on the face s_4 (and thus in particular at vertices v_1, v_2, v_3).

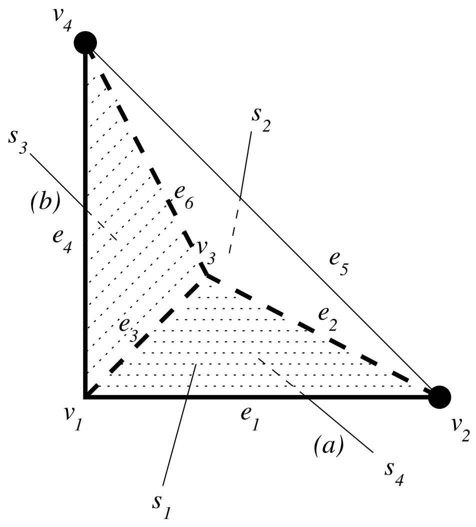

FIGURE 2.32: Edge functions $\varphi_{n,T}^{e_5}$, $2 \le n$, (***a, b***) vanish on the faces s_4, s_3 ($\lambda_{4,T}\lambda_{3,T} \equiv 0$ on s_3, s_4), and thus in particular also along all edges except for e_5.

polynomial orders $3 \le k \le p^{s_i}$ on s_i, and to vanish on all remaining faces. Before dealing with faces, however, we need to define for each of them a unique orientation (which will be independent of the orientation of the edges). This can be done, e.g., by defining a triad of vertices $v_A, v_B, v_C \in s_i$ (*first, second and third* vertex of face s_i) in such a way that v_A, v_C have the lowest and highest local index, respectively.

This also means that for each face s_i we have three affine coordinates $\lambda_A, \lambda_B, \lambda_C$, such that $\lambda_A(v_A) = \lambda_B(v_B) = \lambda_C(v_C) = 1$, and can define $(p^{s_i} - 2)(p^{s_i} - 1)/2$ (oriented) face functions

$$\varphi^{s_i}_{n_1,n_2,T} = \lambda_A \lambda_B \lambda_C \phi_{n_1-1}(\lambda_B - \lambda_A)\phi_{n_2-1}(\lambda_A - \lambda_C), \qquad (2.39)$$

$1 \le n_1, n_2;\ n_1 + n_2 \le p^{s_i} - 1$. According to Remark 2.13, traces of these functions coincide with bubble functions of the master triangle $\mathcal{K}_t^1$ (possibly up to an affine transformation). Their construction is illustrated in Figure 2.33.

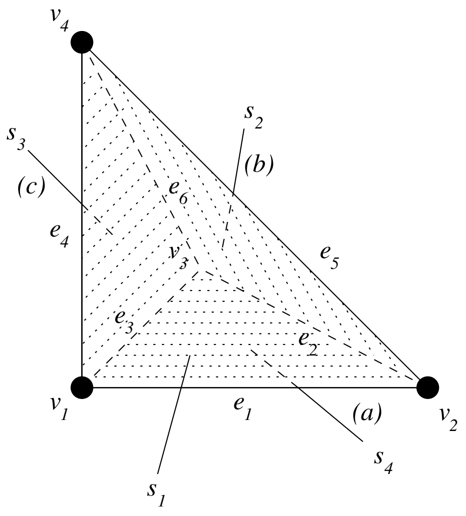

FIGURE 2.33: Face functions $\varphi^{s_1}_{n_1,n_2,T}$, $1 \le n_1, n_2$; (**a, b, c**) vanish on faces s_4, s_2, s_3 ($\lambda_A \lambda_B \lambda_C \equiv 0$ on s_2, s_3, s_4, where $\lambda_A = \lambda_{2,T}, \lambda_B = \lambda_{3,T}, \lambda_C = \lambda_{4,T}$). They have nonzero traces on the face s_1.

Bubble functions, vanishing everywhere on the boundary of the reference tetrahedron, appear in the basis of W_T if $4 \le p^b$. The simplest way to obtain them is as products of vertex functions with varying powers,

$$\varphi^b_{n_1,n_2,n_3,T} = \lambda_{1,T} \lambda^{n_1}_{2,T} \lambda^{n_2}_{3,T} \lambda^{n_3}_{4,T},$$

$1 \le n_1, n_2, n_3$; $n_1+n_2+n_3 \le p^b-1$. However, similarly to the triangular case, we can improve the conditioning properties of bubble functions by choosing

$$\varphi^b_{n_1,n_2,n_3,T} = \phi_{n_1-1}(\lambda_{1,T}-\lambda_{2,T})\phi_{n_2-1}(\lambda_{3,T}-\lambda_{2,T})\phi_{n_3-1}(\lambda_{4,T}-\lambda_{2,T})\prod_{i=1}^{4}\lambda_{i,T}, \tag{2.40}$$

$1 \le n_1, n_2, n_3$; $n_1+n_2+n_3 \le p^b-1$ (we refer back to Figure 2.25).

REMARK 2.19 (Nonsymmetry of bubble functions) The fact that the bubble functions are not symmetric with respect to vertices is widely known, and we have observed this already in the triangular case. This effect is obviously not very pleasant from the algorithmic point of view, but it does not influence the approximation properties of the shape functions. $\square$

Numbers of scalar hierarchic shape functions in the basis of the space W_T are summarized in Table 2.4.

TABLE 2.4: Scalar hierarchic shape functions of $\mathcal{K}^1_T$.

Node type	**Polynomial order**	**Number of shape functions**	**Number of nodes**
Vertex	always	1	4
Edge	$2 \le p^{e_j}$	$p^{e_j}-1$	6
Face	$3 \le p^{s_i}$	$(p^{s_i}-2)(p^{s_i}-1)/2$	4
Interior	$4 \le p^b$	$(p^b-3)(p^b-2)(p^b-1)/6$	1

PROPOSITION 2.5
Shape functions (2.37), (2.38), (2.39) and (2.40) constitute a hierarchic basis of the space W_T, defined in (2.35).

PROOF The same as in the previous cases. Perform three steps: verify that all shape functions lie in the space W_T, that they are linearly independent, and that their number is equal to the dimension of W_T. $\square$

REMARK 2.20 Traces of edge functions corresponding to master elements $\mathcal{K}^1_B$ and $\mathcal{K}^1_T$ on edges of the corresponding reference domains K_B and K_T in both cases coincide with the Lobatto shape functions $l_2, l_3, \ldots$ given by (1.49). This, together with the linearity of vertex functions along element edges, is a good starting point for combining bricks and tetrahedra in hybrid meshes.

However, still missing is an additional finite element capable of matching *both quadrilateral and triangular faces.* This will be presented in the next paragraph. $\square$

2.2.6 Prismatic master element $\mathcal{K}_P^1$

Prismatic elements are most commonly used to connect bricks and tetrahedra in hybrid meshes. We choose a reference prismatic geometry in the product form $K_P = K_t \times K_a$,

$$K_P = \{\boldsymbol{\xi} \in \mathbb{R}^3;\ -1 < \xi_1, \xi_2, \xi_3;\ \xi_1 + \xi_2 < 0;\ \xi_3 < 1\}, \tag{2.41}$$

shown in Figure 2.34.

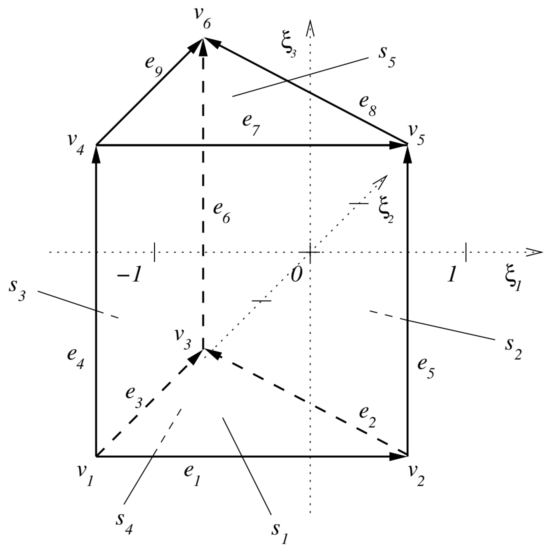

FIGURE 2.34: The reference prism K_P.

REMARK 2.21 The corresponding two- and one-dimensional affine coordinates have the form

$$\lambda_{1,P}(\xi_1,\xi_2,\xi_3) = \frac{\xi_2+1}{2},\ \lambda_{2,P}(\xi_1,\xi_2,\xi_3) = -\frac{\xi_1+\xi_2}{2}, \tag{2.42}$$
$$\lambda_{3,P}(\xi_1,\xi_2,\xi_3) = \frac{\xi_1+1}{2},\ \lambda_{4,P}(\xi_1,\xi_2,\xi_3) = \frac{\xi_3+1}{2},$$

$$\lambda_{5,P}(\xi_1,\xi_2,\xi_3) = \frac{1-\xi_3}{2},$$

with $\lambda_{1,P},\dots,\lambda_{3,P}$ compatible with $\lambda_{1,t},\dots,\lambda_{3,t}$ from the triangular case. $\square$

We consider the possibility of anisotropic p-refinement of prismatic elements, and therefore assign two local directional orders of approximation $p^{b,1}, p^{b,2}$ to the element interior. The order $p^{b,1}$ corresponds to the plane $\xi_1\xi_2$ (we will designate this the *planar direction*), and $p^{b,2}$ to the *vertical direction* ξ_3. There are three quadrilateral faces s_i, $i = 1,\dots,3$, which will be equipped with local directional orders of approximation $p^{s_i,1}, p^{s_i,2}$ (in planar and vertical direction, respectively). Triangular faces s_4, s_5 come with one local order of approximation p^{s_i} only, $i = 4,5$, and local polynomial orders $p^{e_1},\dots,p^{e_9}$ are assigned to edges. As usual, let us mention that these local orders of approximation originate in the physical mesh, and have to obey the *minimum rule* for H^1-conforming approximations.

These local orders suggest that a finite element of the form $\mathcal{K}_P^1 = (K_P, W_P, \Sigma_P^1)$ will be assigned polynomial space

$$\begin{aligned} W_P = \{w \in \mathcal{R}_{p^{b,1},p^{b,2}}(K_P);\ w|_{s_i} \in \mathcal{Q}_{p^{s_i,1},p^{s_i,2}}(s_i) \text{ for } i = 1,2,3; \\ w|_{s_i} \in \mathcal{P}_{p^{s_i}}(s_i) \text{ for } i = 4,5;\ w|_{e_j} \in \mathcal{P}_{p^{e_j}}(e_j),\ j = 1,\dots,9\}. \end{aligned} \tag{2.43}$$

Here

$$\begin{aligned} \mathcal{R}_{m_1,m_2}(K_P) = \operatorname{span}\{\xi_1^{n_1}\xi_2^{n_2}\xi_3^{n_3};\ (\xi_1,\xi_2,\xi_3) \in K_t \times K_a; \\ 0 \le n_1, n_2;\ n_1 + n_2 \le m_1;\ 0 \le n_3 \le m_2\}. \end{aligned} \tag{2.44}$$

Vertex, *edge*, *face* and *bubble* functions will be used to define a suitable H^1-hierarchic basis in the space W_P.

Vertex functions $\varphi_P^{v_j}$, $j = 1,2,\dots,6$, are, as usual, associated with element vertices, and they provide a complete basis of W_P for lowest-order approximation. This time we define them as

$$\varphi_P^{v_j} = \lambda_{j_1,P}\lambda_{j_2,P}. \tag{2.45}$$

The indices j_1, j_2 correspond to the only two faces s_{j_1}, s_{j_2} of the reference prism K_P that *do not* contain the vertex v_j (recall that an affine coordinate is associated with the face where it entirely vanishes). In the standard sense, vertex functions $\varphi_P^{v_j}$ are equal to one at v_j and vanish at all remaining vertices. Their traces are linear on all edges. Construction of the vertex functions is illustrated in Figure 2.35.

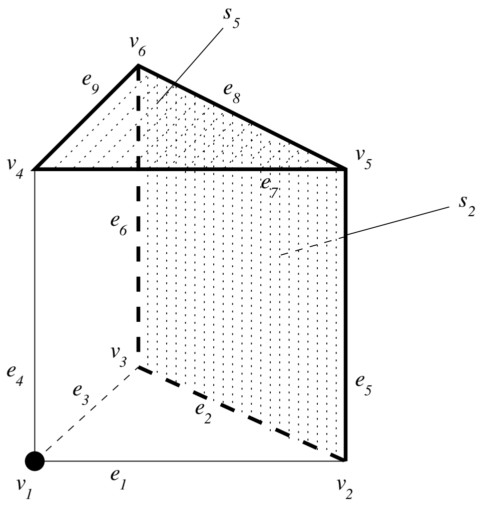

FIGURE 2.35: Vertex function $\varphi_P^{v_1}$ is equal to one at the vertex v_1, and vanishes entirely on the faces s_2, s_5. Thus it vanishes at all remaining vertices.

Edge functions $\varphi_{k,P}^{e_j}$, $j = 1, \dots, 9$, $k = 2, \dots, p^{e_j}$, will be designed to coincide with the Lobatto shape functions $l_2, \dots, l_{p^{e_j}}$ on edges e_j, $j = 1, 2, \dots, 9$, and will vanish on all remaining edges. Let us choose an (oriented) edge $e_j = v_{i_1}v_{i_2}$. By s_{j_1}, s_{j_2} we denote the faces of the reference domain K_P that share a single vertex v_{i_1} or v_{i_2} with the edge e_j, respectively. Further by s_{j_3} we denote the only face of K_P that does not share any vertex with the edge e_j. We add to the basis of W_P (oriented) edge functions

$$\varphi_{k,P}^{e_j} = \lambda_{j_1,P}\lambda_{j_2,P}\phi_{k-2}(\lambda_{j_1,P} - \lambda_{j_2,P})\lambda_{j_3,P}, \quad 2 \le k \le p^{e_j}. \tag{2.46}$$

Use relation (1.52) to recognize the Lobatto shape functions in this definition. Construction of the edge functions is illustrated in Figure 2.36.

Triangular face functions, associated with faces s_i, $i = 4, 5$, will be designed to have on s_i a nonzero trace of local polynomial order k, $3 \le k \le p^{s_i}$, $1 \le n_1, n_2$; $n_1 + n_2 = k - 1$, and will vanish on all remaining faces. The construction is practically the same as for tetrahedra. First we equip each triangular face with a local orientation – we select three vertices $v_A, v_B, v_C \in s_i$ in such a way that v_A, v_C have the lowest and highest local index, respectively. For each face s_i we therefore have three affine coordinates $\lambda_A, \lambda_B, \lambda_C$, such that $\lambda_A(v_A) = \lambda_B(v_B) = \lambda_C(v_C) = 1$. By λ_D we denote affine coordinate corresponding to the other triangular face s_D, and write $(p^{s_i} - 2)(p^{s_i} - 1)/2$ face functions

$$\varphi_{n_1,n_2,T}^{s_i} = \lambda_A\lambda_B\lambda_C\phi_{n_1-1}(\lambda_B - \lambda_A)\phi_{n_2-1}(\lambda_A - \lambda_C)\lambda_D, \tag{2.47}$$

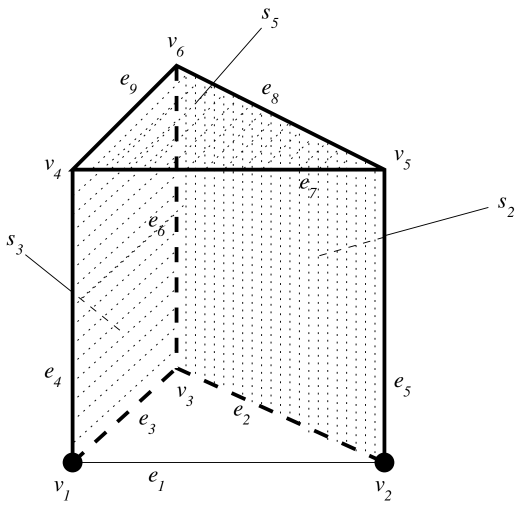

FIGURE 2.36: Traces of edge functions $\varphi^{e_1}_{n,P}$, $2 \le n$, coincide with the Lobatto shape functions $l_2, l_3, \dots$ on the edge e_1, and vanish identically on all faces where the edge e_1 is not contained. Thus they also vanish along all remaining edges.

$1 \le n_1, n_2;\ n_1 + n_2 \le p^{s_i} - 1$.

Quadrilateral face functions, corresponding to faces s_i, $i = 1, \dots, 3$, will be constructed to have on s_i a trace of local directional polynomial orders n_1, n_2, $2 \le n_1 \le p^{s_j,1}$, $2 \le n_2 \le p^{s_j,2}$, and will vanish on all remaining faces. Local coordinate axes on the faces are now chosen to share direction with the corresponding horizontal and vertical edges (see Figure 2.34).

There is a unique pair of edges belonging to the face s_i, both of which are parallel to the plane $\xi_1\xi_2$. From this pair we select the (oriented) edge $e_j = v_{j_1}v_{j_2}$ belonging to the bottom face s_4. Further, by s_{i_1}, s_{i_2} we denote the pair of faces that share a single vertex v_{j_1} or v_{j_2} with the edge e_j, respectively.

We can define face functions

$$\varphi^{s_i}_{n_1,n_2,P} = \lambda_{i_1,P}\lambda_{i_2,P}\lambda_{4,P}\lambda_{5,P}\phi_{n_1-2}(\lambda_{i_1,P} - \lambda_{i_2,P})\phi_{n_2-2}(\lambda_{4,P} - \lambda_{5,P}), \quad (2.48)$$

$2 \le n_1 \le p^{s_i,1}$, $2 \le n_2 \le p^{s_i,2}$. The construction of face functions is illustrated in Figure 2.37.

Bubble functions as usual vanish everywhere on the boundary of the reference domain. It will probably come as no surprise that we construct them as products of bubble functions corresponding to the master triangle $\mathcal{K}^1_t$, and the Lobatto shape functions $l_2, l_3, \dots, l_{p^{b,2}}$ in ξ_3:

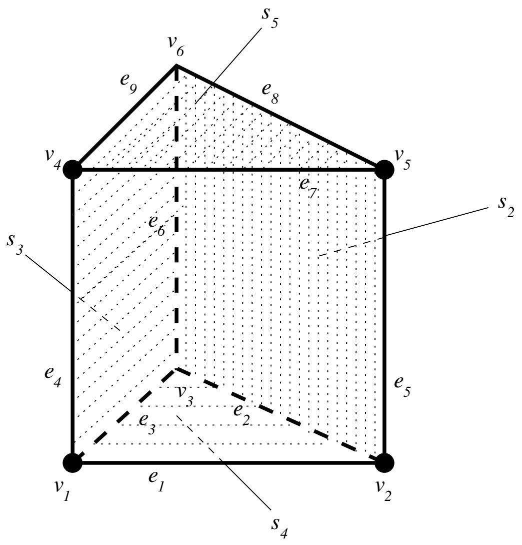

FIGURE 2.37: Quadrilateral face functions $\varphi^{s_1}_{n_1,n_2,P}$, $2 \le n_1, n_2$ are nonzero on the face s_1 and in the element interior. They vanish on all faces except for s_1 and obviously on all edges and vertices as well.

$$\varphi^b_{n_1,n_2,n_3,P} = \lambda_{1,P}\lambda_{2,P}\lambda_{3,P}\phi_{n_1-1}(\lambda_{3,P} - \lambda_{2,P})\phi_{n_2-1}(\lambda_{2,P} - \lambda_{1,P})l_{n_3}(\xi_3), \tag{2.49}$$

$1 \le n_1, n_2$; $n_1 + n_2 \le p^{b,1} - 1$; $2 \le n_3 \le p^{b,2}$. Hence the number of bubble functions is

$$(p^{b,1} - 2)(p^{b,1} - 1)(p^{b,2} - 1)/2. \tag{2.50}$$

Numbers of scalar hierarchic shape functions in the basis of the space W_P are summarized in Table 2.5.

PROPOSITION 2.6

Shape functions (2.45), (2.46), (2.47), (2.48) and (2.49) constitute a hierarchic basis of the space W_P, defined in (2.43).

PROOF Analogous as in the previous cases. $\square$

REMARK 2.22 Edge functions associated with master elements $\mathcal{K}^1_B$, $\mathcal{K}^1_T$ and $\mathcal{K}^1_P$, restricted to edges of these elements, coincide with the Lobatto shape

TABLE 2.5: Scalar hierarchic shape functions of $\mathcal{K}_P^1$.

Node type	**Polynomial order**	**Number of shape functions**	**Number of nodes**
Vertex	always	1	6
Edge	$2 \le p^{e_j}$	$p^{e_j} - 1$	9
Triangular face	$3 \le p^{s_i}$	$(p^{s_i} - 2)(p^{s_i} - 1)/2$	2
Quadrilateral face	$2 \le p^{s_i,1}, p^{s_i,2}$	$(p^{s_i,1} - 1)(p^{s_i,2} - 1)$	3
Interior	$3 \le p^{b,1}, 2 \le p^{b,2}$	(2.50)	1

functions $l_2, l_3, \dots$ from (1.49). Quadrilateral face functions of $\mathcal{K}_P^1$ are compatible with face functions of $\mathcal{K}_B^1$, and triangular face functions of $\mathcal{K}_P^1$ match these of $\mathcal{K}_T^1$. Hence, prismatic elements can be used as interfaces between the hexahedral and tetrahedral element in hybrid meshes. Construction of hybrid meshes, in both two and three spatial dimensions, will be discussed in more detail in Chapter 3. $\square$

2.3 $\boldsymbol{H}$(curl)-conforming approximations

$\boldsymbol{H}$(curl)-conforming finite elements attracted the attention of the Maxwell's computational community after it turned out that vector-valued finite elements, whose components span H^1-conforming polynomial subspaces, are inappropriate (see, e.g, [35, 107, 138, 141, 142, 189]). This has led to the application of *Whitney elements* [204] that constitute a lowest-order approximation over the element with constant tangential components on the edges.

However, there is an increasingly widespread interest in the use of higher-order finite element schemes. Cubic finite elements were constructed in [4, 89, 202] for triangular meshes. A basis that allows for arbitrary order of approximation on triangles and tetrahedra can be found in [203]. Other two-dimensional finite elements of variable order have been proposed in [160]. For a more theoretical discussion of degrees of freedom on hybrid meshes with uniform order of approximation see [115, 116, 139]. The problem of deriving bases for arbitrary order approximation of $\boldsymbol{H}$(curl) and $\boldsymbol{H}$(div) was addressed mainly in more recent works [5, 6, 67, 65, 183].

To our knowledge the first three-dimensional *hp*-adaptive code for electromagnetics, based on hexahedral elements of variable order, has been presented in [161]. There is an ongoing research effort by Prof. U. Langer's group in Linz aimed at coupling *hp*-FEM with parallel multigrid algorithms (see [103] and other recent papers).

In this section we will present an overview of $\boldsymbol{H}$(curl)-conforming finite elements of arbitrary order associated with reference domains K_q, K_t, K_B, K_T and K_P. As in the previous section, attention will be paid to the possibility of a locally nonuniform distribution of the order of polynomial approximation and anisotropic p-refinement. All master elements will be constructed in harmony with the De Rham diagram, i.e., as descendants of appropriate H^1-conforming elements.

2.3.1 De Rham diagram and finite elements in $\boldsymbol{H}(\mathrm{curl})$

The relation

$$H^1 \xrightarrow{\nabla} \boldsymbol{H}(\mathrm{curl}),$$

which is present in both the 2D and 3D versions of the De Rham diagram, indicates that every $\boldsymbol{H}(\mathrm{curl})$-conforming element $\mathcal{K}^{\mathrm{curl}} = (K, \boldsymbol{Q}, \Sigma^{\mathrm{curl}})$ should be understood as a descendant of an appropriate scalar finite element $\mathcal{K}^1 = (K, W, \Sigma^1)$, such that

$$W \xrightarrow{\nabla} \boldsymbol{Q}.$$

If the ancestor element $\mathcal{K}^1$ cannot be found and this relation cannot be established, the finite element scheme in $\boldsymbol{H}(\mathrm{curl})$ will not work properly. As a particular consequence of compatibility with the De Rham diagram, the finite element space $\boldsymbol{Q}$ must respect *conformity requirements* for $\boldsymbol{H}(\mathrm{curl})$-conforming approximations (continuity of tangential component across element interfaces: see Lemma 1.3 in Paragraph 1.1.4).

REMARK 2.23 (Reduced conformity requirements in $\boldsymbol{H}(\mathrm{curl})$) In comparison with the space H^1 where the requirement of global continuity of approximation constrained the function values at the vertices and on edges and faces, the space $\boldsymbol{H}(\mathrm{curl})$ has reduced conformity requirements (see Paragraph 1.1.4). This appropriately simplifies the hierarchic structure of master element shape functions – there will be *no vertex functions* in $\boldsymbol{Q}$, as function values at vertices are not constrained in $\boldsymbol{H}(\mathrm{curl})$. $\square$

Recall that scalar edge functions from the previous section, restricted to edges of scalar H^1-conforming elements, coincide with the Lobatto shape functions l_k, $k = 2, 3, \ldots$. We needed scalar edge functions to vanish at vertices, in order not to interfere there with values of vertex functions. In the $\boldsymbol{H}(\mathrm{curl})$-conforming case, tangential components of edge functions do not have to vanish at element vertices anymore, and thus it will be sufficient to use *Legendre polynomials* $L_0, L_1, \ldots$ to generate the corresponding polynomial subspaces. Legendre polynomials are a natural choice suggested also by the De Rham diagram, as they appear in tangential components *of gradients* of scalar edge functions.

Another agreeable aspect of using Legendre polynomials to generate the tangential components on edges is that the first Legendre polynomial, $L_0 \equiv 1$, elegantly incorporates Whitney functions into the hierarchy of edge functions.

As in the previous H^1-conforming case, a finite element will be understood, in the sense of [47], as a triad $\mathcal{K}^{\text{curl}} = (K, \boldsymbol{Q}, \Sigma^{\text{curl}})$, where K is a geometrical domain, $\boldsymbol{Q}$ a finite-dimensional vector-valued polynomial space, and Σ^{curl} a set of degrees of freedom.

2.3.2 Quadrilateral master element $\mathcal{K}_q^{\text{curl}}$

Let us start with the simplest master element of arbitrary order $\mathcal{K}_q^{\text{curl}}$ on the reference quadrilateral domain K_q (Figure 2.1).

To allow for its anisotropic p-refinement, we consider local directional orders $p^{b,1}, p^{b,2}$ in the element interior. There are usual local orders of approximation $p^{e_1}, \ldots, p^{e_4}$, associated with its edges $e_1, \ldots, e_4$.

REMARK 2.24 (Minimum rules in $\boldsymbol{H}$(curl)) These nonuniform local orders of approximation come from a physical mesh element, and at this time they have to obey the minimum rule *for $\boldsymbol{H}$(curl)-conforming approximations* (polynomial orders *of tangential components* of approximation on physical mesh edges are not allowed to exceed the corresponding local directional orders in interior of adjacent elements). ▯

REMARK 2.25 Notice the specific way,

$$\xi_1^{i+1}\xi_2^{j+1} \xrightarrow{\boldsymbol{\nabla}} \left((i+1)\xi_1^i\xi_2^{j+1}, (j+1)\xi_1^{i+1}\xi_2^j\right),$$

in which the gradient operator $\boldsymbol{\nabla}$ transforms scalar monomials from the space W_q to vector-valued ones. ▯

It follows from Remark 2.25 that in order to fit into the De Rham diagram, a finite element of the form $\mathcal{K}_q^{\text{curl}} = (K_q, \boldsymbol{Q}_q, \Sigma_q^{\text{curl}})$ needs to be associated with polynomial space

$$\boldsymbol{Q}_q = \left\{\boldsymbol{E} \in \mathcal{Q}_{p^{b,1},p^{b,2}+1} \times \mathcal{Q}_{p^{b,1}+1,p^{b,2}};\ \boldsymbol{E}\cdot\boldsymbol{t}|_{e_j} \in \mathcal{P}_{p^{e_j}}(e_j),\ j = 1,\ldots,4\right\}, \tag{2.51}$$

which is a natural descendant of the scalar polynomial space (2.11),

$$W_q = \left\{w \in \mathcal{Q}_{p^{b,1}+1,p^{b,2}+1};\ w|_{e_j} \in \mathcal{P}_{p^{e_j}+1}(e_j), \jmath = 1,\ldots,4\right\}. \tag{2.52}$$

What remains to be done is to define a suitable hierarchic basis of the local polynomial space $\boldsymbol{Q}_q$. To follow $\boldsymbol{H}$(curl)-conformity requirements, we split the hierarchic shape functions into *edge functions* and *bubble functions*.

Edge functions $\psi_{k,q}^{e_j}$, associated with edges e_j, $j = 1,\ldots,4$, with $k = 0,\ldots,p^{e_j}$, will be defined simply as

$$\begin{aligned}
\psi^{e_1}_{k,q} &= l_0(\xi_1)L_k(\xi_2)\boldsymbol{\xi}_2, \quad 0 \le k \le p^{e_1},\\
\psi^{e_2}_{k,q} &= l_1(\xi_1)L_k(\xi_2)\boldsymbol{\xi}_2, \quad 0 \le k \le p^{e_2},\\
\psi^{e_3}_{k,q} &= L_k(\xi_1)l_0(\xi_2)\boldsymbol{\xi}_1, \quad 0 \le k \le p^{e_3},\\
\psi^{e_4}_{k,q} &= L_k(\xi_1)l_1(\xi_2)\boldsymbol{\xi}_1, \quad 0 \le k \le p^{e_4},
\end{aligned} \tag{2.53}$$

where $\boldsymbol{\xi}_1, \boldsymbol{\xi}_2$ are canonical vectors corresponding to axes ξ_1 and ξ_2, respectively. Notice that the trace of the tangential component of the functions $\psi^{e_j}_{k,q}$ coincides with the Legendre polynomials $L_0, \ldots, L_{p^{e_j}}$ on edge e_j, and vanishes on all remaining edges. As in the scalar case, these functions are implicitly *oriented* accordingly to the corresponding edges (recall Figure 2.1).

The lowest-order functions $\psi^{e_j}_{0,q}$, $j = 1, \ldots, 4$, whose tangential components are constant on all edges e_j, form a complete element (*Whitney element*, [204]), and are often called *Whitney functions.* The construction is illustrated in Figure 2.38.

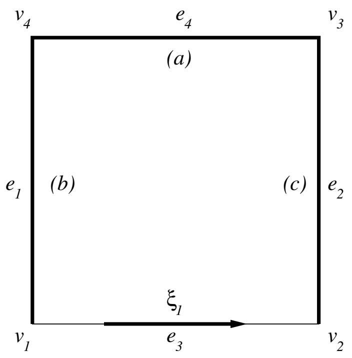

FIGURE 2.38: Tangential components of edge functions $\psi^{e_3}_{k,q}$, $k = 0, \ldots, p^{e_3}$, coincide with the Legendre polynomials on the edge e_3, and vanish on all remaining edges; ***(a)*** $l_0(\xi_2) \equiv 0$ on e_4, and ***(b), (c)*** $\boldsymbol{\xi}_1 \cdot \boldsymbol{t} \equiv 0$ on e_1, e_2.

REMARK 2.26 (Intuition for the design of bubble functions) Notice that tangential components of gradients of the scalar bubble functions (2.15), restricted to edges, are nothing but their *tangential derivatives* along edges. Hence it is clear that gradients of scalar bubble functions are bubble functions in the $\boldsymbol{H}$(curl)-conforming sense. An analogous intuition relates scalar edge functions (2.14) and vector-valued edge functions (2.53). It is however not sufficient to take gradients of scalar bubble functions as bubble functions for $\boldsymbol{Q}_q$, because the space $\boldsymbol{H}$(curl) is larger than $\nabla(H^1)$. Observe that gradients of scalar bubble functions (2.15) (of the form $l_i(\xi_1)l_j(\xi_2)$, $2 \le i,j$), corresponding to the ancestor space (2.52), can be written as

$$\nabla\left(l_i(\xi_1)l_j(\xi_2)\right) = L_{i-1}(\xi_1)l_j(\xi_2)\boldsymbol{\xi}_1 + l_i(\xi_1)L_{j-1}(\xi_2)\boldsymbol{\xi}_2 .$$

$\square$

We define vector-valued *bubble functions* by

$$\begin{aligned} \psi^{b,1}_{n_1,n_2,q} &= L_{n_1}(\xi_1)l_{n_2}(\xi_2)\boldsymbol{\xi}_1, \quad 0 \le n_1 \le p^{b,1},\ 2 \le n_2 \le p^{b,2}+1, \\ \psi^{b,2}_{n_1,n_2,q} &= l_{n_1}(\xi_1)L_{n_2}(\xi_2)\boldsymbol{\xi}_2, \quad 2 \le n_1 \le p^{b,1}+1,\ 0 \le n_2 \le p^{b,2}. \end{aligned} \tag{2.54}$$

REMARK 2.27 Notice that the shape functions (2.54) are *internal* in the $\boldsymbol{H}$(curl)-conforming sense, i.e., they are not restricted by the minimum rule and their polynomial order climbs up all the way to the local directional orders of approximation $p^{b,1}, p^{b,2}$ in element interior. $\square$

Numbers of vector-valued shape functions in the hierarchic basis of the space $\boldsymbol{Q}_q$ are summarized in Table 2.6.

TABLE 2.6: Vector-valued hierarchic shape functions of $\mathcal{K}^{\mathrm{curl}}_q$.

Node type	**Polynomial order**	**Number of shape functions**	**Number of nodes**
Edge	always	$p^{e_j}+1$	4
Interior	$1 \le p^{b,1}$ or $1 \le p^{b,2}$	$(p^{b,1}+1)p^{b,2} + p^{b,1}(p^{b,2}+1)$	1

PROPOSITION 2.7
Vector-valued shape functions (2.53) and (2.54) constitute a hierarchic basis of the space $\boldsymbol{Q}_q$, defined in (2.51).

PROOF It is easy to see that all the functions in (2.53) and (2.54) lie in the space $\boldsymbol{Q}_q$, and that they are linearly independent. We conclude by verifying that the number of edge functions, $(p^{e_1}+1)+(p^{e_2}+1)+(p^{e_3}+1)+(p^{e_4}+1)$, plus the number of bubble functions, $(p^{b,1}+1)(p^{b,2})+(p^{b,1})(p^{b,2}+1)$, is equal to the dimension of space $\boldsymbol{Q}_q$, $(p^{b,1}+1)(p^{b,2}+2)+(p^{b,1}+2)(p^{b,2}+1)-(p^{b,2}-p^{e_1})-(p^{b,2}-p^{e_2})-(p^{b,1}-p^{e_3})-(p^{b,1}-p^{e_4})$. $\square$

REMARK 2.28 For uniform distribution of order of approximation in physical mesh we have $p = p^{b,1} = p^{b,2} = p^{e_1} = \ldots = p^{e_4}$, and the basis from Proposition 2.7 reduces to a basis of the space $\mathcal{Q}_{p,p+1} \times \mathcal{Q}_{p+1,p}$,

$$\begin{aligned} &L_i(\xi_1)l_j(\xi_2)\boldsymbol{\xi}_1, \\ &l_j(\xi_1)L_i(\xi_2)\boldsymbol{\xi}_2, \end{aligned}$$

$i = 0, \ldots, p$, $j = 0, \ldots, p+1$. This basis has been analyzed in [7], and exhibited very good conditioning properties for Maxwell's equations. $\square$

REMARK 2.29 The basis from Proposition 2.7 can again easily be expressed in terms of affine coordinates (2.10), using relations

$$\begin{aligned} \xi_1 &= \lambda_{1,q} - \lambda_{2,q}, \\ \xi_2 &= \lambda_{3,q} - \lambda_{4,q}. \end{aligned}$$

$\square$

2.3.3 Triangular master element $\mathcal{K}_t^{\mathrm{curl}}$

Next on our list of vector-valued master elements of arbitrary order is $\mathcal{K}_t^{\mathrm{curl}}$, associated with the reference triangular domain K_t (Figure 2.13).

We consider a local order of approximation p^b in element interior, and usual local orders p^{e_j} for edges e_j, $j = 1, \ldots, 3$. The minimum rule for $\boldsymbol{H}(\mathrm{curl})$-conforming approximations, enforced in the physical mesh, locally on the reference domain translates into $p^{e_j} \le p^b$ for all $j = 1, \ldots, 3$.

The polynomial space $\mathcal{P}_{p^b+1}(K_t)$ of the form (2.19) does not have a product structure that would allow the gradient operator $\boldsymbol{\nabla}$ to degrade the polynomials in one direction at a time. Hence, the right polynomial space for a finite element of the form $\mathcal{K}_t^{\mathrm{curl}} = (K_t, \boldsymbol{Q}_t, \Sigma_t^{\mathrm{curl}})$ is now simpler,

$$\boldsymbol{Q}_t = \left\{ \boldsymbol{E} \in (\mathcal{P}_{p^b})^2(K_t);\ \boldsymbol{E} \cdot \boldsymbol{t}|_{e_j} \in \mathcal{P}_{p^{e_j}}(e_j),\ j = 1, \ldots, 3 \right\}. \qquad (2.55)$$

In the sense of the De Rham diagram (2.1), an appropriate scalar ancestor space is

$$W_t = \left\{ w \in \mathcal{P}_{p^b+1}(K_t);\ w|_{e_j} \in \mathcal{P}_{p^{e_j}+1}(e_j),\ j = 1, \ldots, 3 \right\}.$$

REMARK 2.30 Recall the relation between affine coordinates $\lambda_{1,t}, \ldots, \lambda_{3,t}$, and unitary normal vectors to edges,

$$\boldsymbol{n}_{i,t} = \frac{\nabla \lambda_{i,t}}{|\nabla \lambda_{i,t}|}, \quad i = 1, \ldots, 3. \qquad (2.56)$$

Hierarchic basis of $\boldsymbol{Q}_t$ will comprise again *edge* and *bubble functions.*

Edge functions $\psi_{k,t}^{e_j}$, $j = 1,\dots,3$, $k = 0,\dots,p^{e_j}$, will be constructed so that traces of their tangential component coincide with the Legendre polynomials $L_0, L_1, \dots, L_{p^{e_j}}$ on the edge e_j, and vanish on all remaining edges. We start with Whitney functions,

$$\begin{aligned}
\psi_{0,t}^{e_1} &= \frac{\lambda_{3,t}\boldsymbol{n}_{2,t}}{\boldsymbol{n}_{2,t}\cdot\boldsymbol{t}_{1,t}} + \frac{\lambda_{2,t}\boldsymbol{n}_{3,t}}{\boldsymbol{n}_{3,t}\cdot\boldsymbol{t}_{1,t}},\\
\psi_{0,t}^{e_2} &= \frac{\lambda_{1,t}\boldsymbol{n}_{3,t}}{\boldsymbol{n}_{3,t}\cdot\boldsymbol{t}_{2,t}} + \frac{\lambda_{3,t}\boldsymbol{n}_{1,t}}{\boldsymbol{n}_{1,t}\cdot\boldsymbol{t}_{2,t}},\\
\psi_{0,t}^{e_3} &= \frac{\lambda_{2,t}\boldsymbol{n}_{1,t}}{\boldsymbol{n}_{1,t}\cdot\boldsymbol{t}_{3,t}} + \frac{\lambda_{1,t}\boldsymbol{n}_{2,t}}{\boldsymbol{n}_{2,t}\cdot\boldsymbol{t}_{3,t}},
\end{aligned} \tag{2.57}$$

which are always present in the basis of $\boldsymbol{Q}_t$, and form a complete basis for lowest-order approximations (*Whitney element*). To get some intuition for these formulae, consider for a moment only the second term $(\lambda_{2,t}\boldsymbol{n}_{3,t})/(\boldsymbol{n}_{3,t}\cdot\boldsymbol{t}_{1,t})$ in the definition of function $\psi_{0,t}^{e_1}$, and look at Figure 2.39.

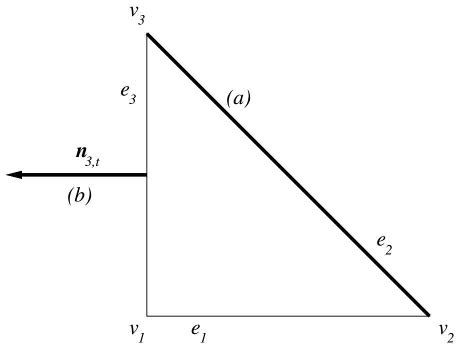

FIGURE 2.39: Trace of the tangential component of $(\lambda_{2,t}\boldsymbol{n}_{3,t})/(\boldsymbol{n}_{3,t}\cdot\boldsymbol{t}_{1,t})$ on edge e_1 matches $\lambda_{2,t}$, and vanishes ***(a)*** on edge e_2 ($\lambda_{2,t}\equiv 0$), and ***(b)*** on edge e_3 ($\boldsymbol{n}_{3,t}\cdot\boldsymbol{t}_{3,t}\equiv 0$).

Thus traces of tangential components of the functions $(\lambda_{3,t}\boldsymbol{n}_{2,t})/(\boldsymbol{n}_{2,t}\cdot\boldsymbol{t}_{1,t})$ and $(\lambda_{2,t}\boldsymbol{n}_{3,t})/\ (\boldsymbol{n}_{3,t}\cdot\boldsymbol{t}_{1,t})$ coincide with the values of (scalar vertex) functions $\lambda_{3,t}$ and $\lambda_{2,t}$ on edge e_1, respectively. Now we see that when summed up, these two parts give rise to a Whitney function ($\lambda_{3,t}+\lambda_{2,t}\equiv 1$ on e_1). Analogously we proceed when constructing *linear edge functions* $\psi_{1,t}^{e_j}$, $j = 1,\dots,3$,

$$\begin{aligned}\psi^{e_1}_{1,t} &= \frac{\lambda_{3,t}\boldsymbol{n}_{2,t}}{\boldsymbol{n}_{2,t}\cdot\boldsymbol{t}_{1,t}} - \frac{\lambda_{2,t}\boldsymbol{n}_{3,t}}{\boldsymbol{n}_{3,t}\cdot\boldsymbol{t}_{1,t}}, \quad p^{e_1} \geq 1, \\ \psi^{e_2}_{1,t} &= \frac{\lambda_{1,t}\boldsymbol{n}_{3,t}}{\boldsymbol{n}_{3,t}\cdot\boldsymbol{t}_{2,t}} - \frac{\lambda_{3,t}\boldsymbol{n}_{1,t}}{\boldsymbol{n}_{1,t}\cdot\boldsymbol{t}_{2,t}}, \quad p^{e_2} \geq 1, \\ \psi^{e_3}_{1,t} &= \frac{\lambda_{2,t}\boldsymbol{n}_{1,t}}{\boldsymbol{n}_{1,t}\cdot\boldsymbol{t}_{3,t}} - \frac{\lambda_{1,t}\boldsymbol{n}_{2,t}}{\boldsymbol{n}_{2,t}\cdot\boldsymbol{t}_{3,t}}, \quad p^{e_3} \geq 1.\end{aligned} \tag{2.58}$$

The trace of the tangential component of the function $\psi^{e_j}_{1,t}$ on edge e_j matches the Legendre polynomial $L_1(\zeta) = \zeta$ on e_j ($\zeta \in (-1,1)$ being the parametrization of this edge), and vanishes on all remaining edges. Linear edge functions are present in the basis of the space $\boldsymbol{Q}_t$ only if the corresponding local order p^{e_j}, associated with edge e_j, is equal to at least one. They form, together with the Whitney functions (2.57), a complete basis of $\boldsymbol{Q}_t$ for linear approximations.

Higher-order edge functions appear in the basis of $\boldsymbol{Q}_t$ if some of the local orders p^{e_j} associated with edges are greater than or equal to two. They can be defined by exploiting the recurrent definition of Legendre polynomials from (1.40),

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

which, after incorporating Whitney and linear edge functions, translates into

$$\begin{aligned}\psi^{e_1}_{k,t} &= \frac{2k-1}{k}L_{k-1}(\lambda_{3,t}-\lambda_{2,t})\psi^{e_1}_{1,t} - \frac{k-1}{k}L_{k-2}(\lambda_{3,t}-\lambda_{2,t})\psi^{e_1}_{0,t}, \\ &\quad 2 \leq k \leq p^{e_1}, \\ \psi^{e_2}_{k,t} &= \frac{2k-1}{k}L_{k-1}(\lambda_{1,t}-\lambda_{3,t})\psi^{e_2}_{1,t} - \frac{k-1}{k}L_{k-2}(\lambda_{1,t}-\lambda_{3,t})\psi^{e_2}_{0,t}, \\ &\quad 2 \leq k \leq p^{e_2}, \\ \psi^{e_3}_{k,t} &= \frac{2k-1}{k}L_{k-1}(\lambda_{2,t}-\lambda_{1,t})\psi^{e_3}_{1,t} - \frac{k-1}{k}L_{k-2}(\lambda_{2,t}-\lambda_{1,t})\psi^{e_3}_{0,t}, \\ &\quad 2 \leq k \leq p^{e_3}.\end{aligned} \tag{2.59}$$

The trace of the tangential component of the function $\psi^{e_j}_{k,t}$ on edge e_j now matches the higher-order Legendre polynomial $L_k(\zeta)$ ($\zeta \in (-1,1)$ being the parametrization of this edge as before) and vanishes on all other edges.

REMARK 2.31 Notice that the edge functions (2.57), (2.58) and (2.59) already suffice to generate tangential component of the approximation up to the orders $p^{e_1},\ldots,p^{e_3}$ on edges. $\square$

Bubble functions, whose tangential components vanish everywhere on the element boundary, can be split (see, e.g., [203]) into two groups:

- *edge-based bubble functions* (sometimes called *normal functions*), which vanish everywhere on the boundary except for one edge, by the normal vector to which they are multiplied,
- *genuine bubble functions*, which vanish on all edges.

A quick way to define *edge-based bubble functions* $\psi_{k,t}^{b,e_j}$, $k = 2, \dots, p^b$, would be to multiply scalar edge functions $\varphi_{k,t}^{e_j}$ by normal vector $\boldsymbol{n}_{j,t}$ to the corresponding edge e_j,

$$\psi_{k,t}^{b,e_j} = \varphi_{k,t}^{e_j} \boldsymbol{n}_{j,t}, \quad k = 2, \dots, p^b.$$

However, it is advantageous to explicitly involve Legendre polynomials, in order to improve conditioning properties of the hierarchic basis. We define

$$\begin{aligned} \psi_{k,t}^{b,e_1} &= \lambda_{3,t}\lambda_{2,t}L_{k-2}(\lambda_{3,t} - \lambda_{2,t})\boldsymbol{n}_{1,t}, \quad 2 \le k \le p^b, \\ \psi_{k,t}^{b,e_2} &= \lambda_{1,t}\lambda_{3,t}L_{k-2}(\lambda_{1,t} - \lambda_{3,t})\boldsymbol{n}_{2,t}, \quad 2 \le k \le p^b, \\ \psi_{k,t}^{b,e_3} &= \lambda_{2,t}\lambda_{1,t}L_{k-2}(\lambda_{2,t} - \lambda_{1,t})\boldsymbol{n}_{3,t}, \quad 2 \le k \le p^b. \end{aligned} \tag{2.60}$$

The geometrical intuition behind this construction is given in Figure 2.40.

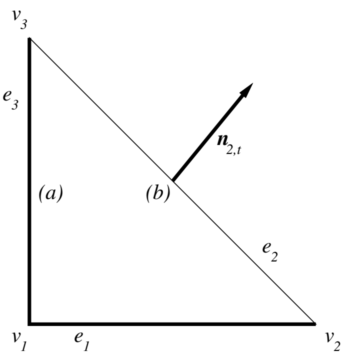

FIGURE 2.40: Multiplied by the normal vector $\boldsymbol{n}_{2,t}$, the functions $\lambda_{1,t}\lambda_{3,t}L_{k-2}(\lambda_{1,t} - \lambda_{3,t})$, $2 \le k \le p^b$, give rise to edge-based bubble functions $\psi_{k,t}^{b,e_2}$, that (**a**) completely vanish on edges e_1, e_3 ($\lambda_{1,t}\lambda_{3,t} \equiv 0$), and (***b***) have a tangential component zero on the edge e_2 ($\boldsymbol{n}_{2,t} \cdot \boldsymbol{t}_{2,t} = 0$).

Finally we design *genuine* bubble functions $\psi_{n_1,n_2,t}^{b,i}$, $k = 2, \dots, p^b$, $i = 1, 2$, $1 \le n_1, n_2$; $n_1 + n_2 \le p^b - 1$. Again, an easy way to define them would be to multiply scalar bubble functions $\varphi_{n_1,n_2,t}^{b}$, by canonical vectors $\boldsymbol{\xi}_1, \boldsymbol{\xi}_2$:

$$\psi^{b,1}_{n_1,n_2,t} = \varphi^{b}_{n_1,n_2,t}\boldsymbol{\xi}_1,$$
$$\psi^{b,2}_{n_1,n_2,t} = \varphi^{b}_{n_1,n_2,t}\boldsymbol{\xi}_2,$$

but for the same reason as before we define

$$\begin{aligned}\psi^{b,1}_{n_1,n_2,t} &= \lambda_{1,t}\lambda_{2,t}\lambda_{3,t}L_{n_1-1}(\lambda_{3,t}-\lambda_{2,t})L_{n_2-1}(\lambda_{2,t}-\lambda_{1,t})\boldsymbol{\xi}_1, \quad (2.61)\\ \psi^{b,2}_{n_1,n_2,t} &= \lambda_{1,t}\lambda_{2,t}\lambda_{3,t}L_{n_1-1}(\lambda_{3,t}-\lambda_{2,t})L_{n_2-1}(\lambda_{2,t}-\lambda_{1,t})\boldsymbol{\xi}_2,\end{aligned}$$

$1 \le n_1, n_2;\ n_1 + n_2 \le p^b - 1$.

Numbers of vector-valued shape functions in the hierarchic basis of the space $\boldsymbol{Q}_t$ are summarized in Table 2.7.

TABLE 2.7: Vector-valued hierarchic shape functions of $\mathcal{K}_t^{\mathrm{curl}}$.

Node type	**Polynomial order**	**Number of shape functions**	**Number of nodes**
Edge	always	$p^{e_j}+1$	3
Edge-based interior	$2 \le p^b$	$3(p^b-1)$	1
Genuine interior	$3 \le p^b$	$(p^b-1)(p^b-2)$	1

PROPOSITION 2.8
Whitney functions (2.57), linear edge functions (2.58), higher-order edge functions (2.59), and bubble functions (2.60), (2.61), constitute a hierarchic basis of the space $\boldsymbol{Q}_t$, defined in (2.55).

PROOF It is a little tedious to compute that the number of the shape functions is equal to the dimension of space $\boldsymbol{Q}_t$, but it is easy to see that they all belong to the space $\boldsymbol{Q}_t$, and that they are linearly independent. We encourage the reader to do this exercise by herself/himself, in order to get familiar with the structure of the hierarchic basis. $\square$

REMARK 2.32 Notice that only edge functions determine the compatibility of two-dimensional $\boldsymbol{H}(\mathrm{curl})$-conforming elements. In our case, traces of tangential components of edge functions to both master elements $\mathcal{K}_q^{\mathrm{curl}}$ and $\mathcal{K}_t^{\mathrm{curl}}$ are nothing but the Legendre polynomials $L_0, L_1, \ldots$. Therefore, the presented hierarchic bases of the spaces $\boldsymbol{Q}_q$, $\boldsymbol{Q}_t$ are convenient for combination of quadrilateral and triangular elements in *hybrid* $\boldsymbol{H}(\mathrm{curl})$*-conforming* *quadrilateral-triangular meshes.* Bubble functions, whose tangential components vanish everywhere on the element boundary, have no influence on their compatibility. $\square$

2.3.4 Brick master element $\mathcal{K}_B^{\text{curl}}$

Cartesian geometry of the reference brick K_B will be used to simplify the discussion in this paragraph. As in the H^1-conforming case, we want to allow for anisotropic p-refinement of brick elements, and therefore consider local directional orders of approximation $p^{b,1}, p^{b,2}$ and $p^{b,3}$ in the element interior. Local directional orders $p^{s_i,1}, p^{s_i,2}$ are assigned to faces s_i, $i = 1, \dots, 6$, and usual local orders of approximation $p^{e_1}, \dots, p^{e_{12}}$ to edges. The local polynomial orders on faces are understood in local systems of coordinates, attached to each face, as described in Paragraph 2.2.4. Again, the local polynomial orders of approximation have to obey the *minimum rule* for $\boldsymbol{H}(\text{curl})$-conforming approximations (Remark 2.24).

REMARK 2.33 The gradient operator ∇, representing an adequate portion of the De Rham diagram (2.3), acts on product monomials from the space W_B one spatial variable (one direction) at a time,

$$\xi_1^{i+1}\xi_2^{j+1}\xi_3^{k+1} \xrightarrow{\nabla} \left((i+1)\xi_1^{i}\xi_2^{j+1}\xi_3^{k+1}, (j+1)\xi_1^{i+1}\xi_2^{j}\xi_3^{k+1}, (k+1)\xi_1^{i+1}\xi_2^{j+1}\xi_3^{k}\right).$$

$\square$

According to Remark 2.33, the De Rham diagram suggests that a finite element of the form $\mathcal{K}_B^{\text{curl}} = (K_B, \boldsymbol{Q}_B, \Sigma_B^{\text{curl}})$ should be equipped with polynomial space

$$\begin{aligned} \boldsymbol{Q}_B = \{ &\boldsymbol{E} \in \mathcal{Q}_{p^{b,1},p^{b,2}+1,p^{b,3}+1} \times \mathcal{Q}_{p^{b,1}+1,p^{b,2},p^{b,3}+1} \times \mathcal{Q}_{p^{b,1}+1,p^{b,2}+1,p^{b,3}}; \\ &\boldsymbol{E}_t|_{s_i} \in \mathcal{Q}_{p^{s_i,1},p^{s_i,2}+1}(s_i) \times \mathcal{Q}_{p^{s_i,1}+1,p^{s_i,2}}(s_i); \\ &\boldsymbol{E}\cdot\boldsymbol{t}|_{e_j} \in \mathcal{P}_{p^{e_j}}(e_j);\ i = 1,\dots,6,\ j = 1,\dots,12\}, \end{aligned} \tag{2.62}$$

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. The ancestor space has the form

$$\begin{aligned} W_B = \{ &w \in \mathcal{Q}_{p^{b,1}+1,p^{b,2}+1,p^{b,3}+1}, w|_{s_i} \in \mathcal{Q}_{p^{s_i,1}+1,p^{s_i,2}+1}(s_i), \\ &w|_{e_j} \in \mathcal{P}_{p^{e_j}+1}(e_j),\ i = 1,\dots,6,\ j = 1,\dots,12\}. \end{aligned} \tag{2.63}$$

To follow $\boldsymbol{H}(\text{curl})$-conformity requirements, we split hierarchic shape functions, which will form a basis of the space $\boldsymbol{Q}_B$, into *edge*, *face* and *bubble functions.* All of them will be written in the form of a product of Legendre polynomials $L_0, L_1, \ldots$, and the Lobatto shape functions $l_0, l_1, \ldots$, similarly as in the quadrilateral case.

Traces of tangential components of *edge functions* $\psi^{e_j}_{k,B}$, $j = 1, \ldots, 12$, $k = 0, \ldots, p^{e_j}$, will as usual vanish on all edges except for an edge e_j with which they are associated, and will coincide with the Legendre polynomials $L_0, L_1, \ldots, L_{p^{e_j}}$ on e_j. Probably the best way to define them is simply to list them all. Recall enumeration and orientation of edges from Figure 2.26:

Edges parallel to $\boldsymbol{\xi}_1$:

$$\begin{aligned}
\psi^{e_1}_{k,B} &= L_k(\xi_1)l_0(\xi_2)l_0(\xi_3)\boldsymbol{\xi}_1, \quad 0 \le k \le p^{e_1},\\
\psi^{e_3}_{k,B} &= L_k(\xi_1)l_1(\xi_2)l_0(\xi_3)\boldsymbol{\xi}_1, \quad 0 \le k \le p^{e_3},\\
\psi^{e_9}_{k,B} &= L_k(\xi_1)l_0(\xi_2)l_1(\xi_3)\boldsymbol{\xi}_1, \quad 0 \le k \le p^{e_9},\\
\psi^{e_{11}}_{k,B} &= L_k(\xi_1)l_1(\xi_2)l_1(\xi_3)\boldsymbol{\xi}_1, \quad 0 \le k \le p^{e_{11}}.
\end{aligned} \tag{2.64}$$

Edges parallel to $\boldsymbol{\xi}_2$:

$$\begin{aligned}
\psi^{e_2}_{k,B} &= l_1(\xi_1)L_k(\xi_2)l_0(\xi_3)\boldsymbol{\xi}_2, \quad 0 \le k \le p^{e_2},\\
\psi^{e_4}_{k,B} &= l_0(\xi_1)L_k(\xi_2)l_0(\xi_3)\boldsymbol{\xi}_2, \quad 0 \le k \le p^{e_4},\\
\psi^{e_{10}}_{k,B} &= l_1(\xi_1)L_k(\xi_2)l_1(\xi_3)\boldsymbol{\xi}_2, \quad 0 \le k \le p^{e_{10}},\\
\psi^{e_{12}}_{k,B} &= l_0(\xi_1)L_k(\xi_2)l_1(\xi_3)\boldsymbol{\xi}_2, \quad 0 \le k \le p^{e_{12}}.
\end{aligned} \tag{2.65}$$

Edges parallel to $\boldsymbol{\xi}_3$:

$$\begin{aligned}
\psi^{e_5}_{k,B} &= l_0(\xi_1)l_0(\xi_2)L_k(\xi_3)\boldsymbol{\xi}_3, \quad 0 \le k \le p^{e_5},\\
\psi^{e_6}_{k,B} &= l_1(\xi_1)l_0(\xi_2)L_k(\xi_3)\boldsymbol{\xi}_3, \quad 0 \le k \le p^{e_6},\\
\psi^{e_7}_{k,B} &= l_1(\xi_1)l_1(\xi_2)L_k(\xi_3)\boldsymbol{\xi}_3, \quad 0 \le k \le p^{e_7},\\
\psi^{e_8}_{k,B} &= l_0(\xi_1)l_1(\xi_2)L_k(\xi_3)\boldsymbol{\xi}_3, \quad 0 \le k \le p^{e_8}.
\end{aligned} \tag{2.66}$$

Twelve Whitney functions, corresponding to $k = 0$, again form a complete lowest-order element.

In the next step we add to the basis of space $\boldsymbol{Q}_B$ *face functions* whose tangential component vanishes, in the usual sense, on all faces but one. There will be two sets of linearly independent face functions for each face, associated with two corresponding linearly independent tangential axial directions (local coordinate axes on the face). Face functions related to the face s_1 can be written as

$$\begin{aligned}
\psi^{s_1,1}_{n_1,n_2,B} &= l_0(\xi_1)L_{n_1}(\xi_2)l_{n_2}(\xi_3)\boldsymbol{\xi}_2,\\
&\qquad 0 \le n_1 \le p^{s_1,1},\ 2 \le n_2 \le p^{s_1,2} + 1,
\end{aligned} \tag{2.67}$$

$$\psi^{s_1,2}_{n_1,n_2,B} = l_0(\xi_1) l_{n_1}(\xi_2) L_{n_2}(\xi_3) \boldsymbol{\xi}_3,$$
$$2 \le n_1 \le p^{s_1,1} + 1,\ 0 \le n_2 \le p^{s_1,2},$$

and we leave the rest to the reader as an easy exercise. The number of face functions associated with a face s_i is

$$(p^{s_i,1} + 1)p^{s_i,2} + p^{s_i,1}(p^{s_i,2} + 1). \tag{2.68}$$

Notice that traces of tangential components of these face functions exactly match the scalar face functions from (2.31).

Finally we design *bubble functions*, whose tangential components vanish everywhere on the boundary of the reference brick K_B. Gradients of scalar shape functions $l_i(\xi_1)l_j(\xi_2)l_k(\xi_3)$, $2 \le i, j, k$, corresponding to the ancestor space (2.63), have the form

$$\begin{aligned} &\nabla \left(l_i(\xi_1)l_j(\xi_2)l_k(\xi_3)\right) \\ &= L_{i-1}(\xi_1)l_j(\xi_2)l_k(\xi_3)\boldsymbol{\xi}_1 + l_i(\xi_1)L_{j-1}(\xi_2)l_k(\xi_3)\boldsymbol{\xi}_2 + l_i(\xi_1)l_j(\xi_2)L_{k-1}(\xi_3)\boldsymbol{\xi}_3. \end{aligned}$$

Therefore, the most natural way to define vector-valued bubble functions for the basis of the space $\boldsymbol{Q}_B$, is

$$\begin{aligned} \psi^{b,1}_{n_1,n_2,n_3,B} &= L_{n_1}(\xi_1) l_{n_2}(\xi_2) l_{n_3}(\xi_3) \boldsymbol{\xi}_1, \\ &\quad 0 \le n_1 \le p^{b,1},\ 2 \le n_2 \le p^{b,2} + 1,\ 2 \le n_3 \le p^{b,3} + 1, \\ \psi^{b,2}_{n_1,n_2,n_3,B} &= l_{n_1}(\xi_1) L_{n_2}(\xi_2) l_{n_3}(\xi_3) \boldsymbol{\xi}_2, \\ &\quad 2 \le n_1 \le p^{b,1} + 1,\ 0 \le n_2 \le p^{b,2},\ 2 \le n_3 \le p^{b,3} + 1, \\ \psi^{b,3}_{n_1,n_2,n_3,B} &= l_{n_1}(\xi_1) l_{n_2}(\xi_2) L_{n_3}(\xi_3) \boldsymbol{\xi}_3, \\ &\quad 2 \le n_1 \le p^{b,1} + 1,\ 2 \le n_2 \le p^{b,2} + 1,\ 0 \le n_3 \le p^{b,3}. \end{aligned} \tag{2.69}$$

Counting them up, we obtain

$$(p^{b,1} + 1)p^{b,2}p^{b,3} + p^{b,1}(p^{b,2} + 1)p^{b,3} + p^{b,1}p^{b,2}(p^{b,3} + 1). \tag{2.70}$$

Numbers of vector-valued shape functions in the hierarchic basis of the space $\boldsymbol{Q}_B$ are summarized in Table 2.8.

PROPOSITION 2.9
Vector-valued shape functions (2.64) – (2.66), (2.67) and (2.69) represent a hierarchic basis of the space $\boldsymbol{Q}_B$, defined in (2.62).

PROOF Using the product structure, it is easy to see that the above basis functions are linearly independent, and that they all lie in the space $\boldsymbol{Q}_B$. The

TABLE 2.8: Vector-valued hierarchic shape functions of $\mathcal{K}_B^{\mathrm{curl}}$.

Node type	Polynomial order	Number of shape functions	Number of nodes
Edge	always	$p^{e_j}+1$	12
Face	$1 \le p^{s_i,1}$ or $1 \le p^{s_i,2}$	see (2.68)	6
Interior	$1 \le p^{b,i}, p^{b,j};\ i \ne j$	see (2.70)	1

calculation of the dimension of $\boldsymbol{Q}_B$, and the verification that it is the same as the number of basis functions, is left to the reader as an exercise. $\square$

REMARK 2.34 For uniform distribution of order of approximation in physical mesh we have $p = p^{b,i} = p^{s_j,k} = p^{e_l}$ for each of i,j,k,l, and the basis from Proposition 2.9 reduces to a basis of the standard space $\mathcal{Q}_{p,p+1,p+1} \times \mathcal{Q}_{p+1,p,p+1} \times \mathcal{Q}_{p+1,p+1,p}$,

$$\left.\begin{array}{l} L_i(\xi_1)l_j(\xi_2)l_k(\xi_3)\boldsymbol{\xi}_1, \\ l_j(\xi_1)L_i(\xi_2)l_k(\xi_3)\boldsymbol{\xi}_2, \\ l_j(\xi_1)l_k(\xi_2)L_i(\xi_3)\boldsymbol{\xi}_3, \end{array}\right\} \quad i = 0,\dots,p;\ j,k = 0,\dots,p+1. \tag{2.71}$$

This basis has been analyzed in [7] with very good conditioning results for the discretization of Maxwell's equations. $\square$

2.3.5 Tetrahedral master element $\mathcal{K}_T^{\mathrm{curl}}$

Next let us design a master element of arbitrary order $\mathcal{K}_T^{\mathrm{curl}}$ on the reference tetrahedral domain K_T (Figure 2.30).

This time we return to a single local polynomial order of approximation p^b in element interior, single local orders p^{s_i}, $i = 1,\dots,4$ on faces, and standard local polynomial orders p^{e_j}, $j = 1,\dots,6$, for edges.

REMARK 2.35 The *minimum rule* for $\boldsymbol{H}(\mathrm{curl})$-conforming approximations (2.24) yields that, locally on the reference domain, orders of approximation associated with faces are lower than or equal to p^b, and polynomial orders on edges are limited from above by polynomial orders associated with both adjacent faces. $\square$

The local orders $p^b, p^{s_1},\dots,p^{s_4}, p^{e_1},\dots,p^{e_6}$ suggest that a finite element of the form $\mathcal{K}_T^{\mathrm{curl}} = (K_T, \boldsymbol{Q}_T, \Sigma_T^{\mathrm{curl}})$ needs to be equipped with polynomial space

$$\begin{aligned} \boldsymbol{Q}_T = \big\{ &\boldsymbol{E} \in (\mathcal{P}_{p^b})^3(K_T);\ \boldsymbol{E}_t|_{s_i} \in (\mathcal{P}_{p^{s_i}})^2(s_i); \\ &\boldsymbol{E}\cdot\boldsymbol{t}|_{e_j} \in \mathcal{P}_{p^{e_j}}(e_j),\ i = 1,\dots,4,\ j = 1,\dots,6 \big\}, \end{aligned} \tag{2.72}$$

where again $\boldsymbol{E}_t|_{s_i} = \boldsymbol{E} - \boldsymbol{n}_i(\boldsymbol{E}\cdot\boldsymbol{n}_i)$ is the projection of the vector $\boldsymbol{E}$ on the face s_i. Based on the De Rham diagram (2.3), the appropriate ancestor space is

$$\begin{aligned} W_T = \{ & w \in \mathcal{P}_{p^b+1}(K_T);\ w|_{s_i} \in \mathcal{P}_{p^{s_i}+1}(s_i);\ w|_{e_j} \in \mathcal{P}_{p^{e_j}+1}(e_j), \\ & i = 1,\dots,4,\ j = 1,\dots,6\}\,. \end{aligned}$$

The set of degrees of freedom Σ_T^{curl} will be uniquely identified by a choice of basis of the space $\boldsymbol{Q}_T$. Recall affine coordinates $\lambda_{1,T},\dots,\lambda_{4,T}$ from (2.34), the Lobatto shape functions $l_0, l_1, \dots$ from (1.49), and kernel functions $\phi_0, \phi_1, \dots$, defined in (1.52). Again we have a relation between affine coordinates and unitary normal vectors to faces,

$$\boldsymbol{n}_{i,T} = \frac{\nabla\lambda_{i,T}}{|\nabla\lambda_{i,T}|}, \quad i = 1,\dots,4. \tag{2.73}$$

Edge functions will be constructed, as usual, in such a way that traces of their tangential components vanish on all edges except for the particular one they are assigned to. Recall enumeration and orientation of edges from Figure 2.30. Consider an (oriented) edge $e_j = v_A v_B$, $j = 1,\dots,6$, and a pair of affine coordinates λ_A, λ_B, such that $\lambda_A(v_A) = \lambda_B(v_B) = 1$. Notice that λ_A vanishes on all edges except for e_j and another pair of edges $e_{A,1}, e_{A,2}$. These three edges uniquely identify a face s_A. Analogously, λ_B yields a face s_B. $\boldsymbol{n}_A$ and $\boldsymbol{n}_B$ denote unitary normal vectors to these two faces.

Now we can easily define linear *vertex-based edge functions*

$$\begin{aligned} \psi_A^{e_j} &= \frac{\lambda_A \boldsymbol{n}_A}{\boldsymbol{n}_A \cdot \boldsymbol{t}_{j,T}}, \\ \psi_B^{e_j} &= \frac{\lambda_B \boldsymbol{n}_B}{\boldsymbol{n}_B \cdot \boldsymbol{t}_{j,T}}, \end{aligned} \tag{2.74}$$

where $\boldsymbol{t}_{j,T}$ is a unitary tangential vector to the oriented edge e_j. Tangential components of both functions obviously vanish on all edges except for e_j (see Figure 2.41), and on e_j we have

$$\begin{aligned} (\psi_A^{e_j} \cdot \boldsymbol{t}_{j,T})|_{e_j} &= \lambda_A|_{e_j}, \\ (\psi_B^{e_j} \cdot \boldsymbol{t}_{j,T})|_{e_j} &= \lambda_B|_{e_j}. \end{aligned}$$

Thus we can define for the edge e_j a *Whitney function*,

$$\psi_{0,T}^{e_j} = \psi_A^{e_j} + \psi_B^{e_j}, \tag{2.75}$$

and a *linear edge function*,

$$\psi_{1,T}^{e_j} = \psi_B^{e_j} - \psi_A^{e_j}. \tag{2.76}$$

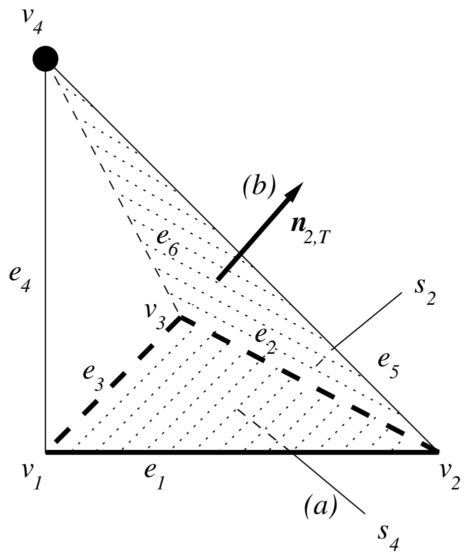

FIGURE 2.41: Consider the edge $e_4 = v_1v_4$ and affine coordinates $\lambda_A = \lambda_{2,T}$ and $\lambda_B = \lambda_{4,T}$. Notice that $\lambda_{4,T}$ vanishes on all edges except for e_4, e_5 and e_6. The edges e_5 and e_6 uniquely identify a face s_2. Hence, multiplied by normal vector $\boldsymbol{n}_{2,T}$, the affine coordinate $\lambda_{4,T}$ yields a vertex-based edge function $\psi_4^{e_4}$. This function (**a**) vanishes completely on face s_4, and (***b***) its tangential component also vanishes everywhere on the face s_2. Thus, the tangential component vanishes on all edges except for e_4.

These functions satisfy

$$(\psi_{0,T}^{e_j} \cdot \boldsymbol{t}_{j,T})|_{e_j} \equiv 1 = L_0(\lambda_B - \lambda_A),$$
$$(\psi_{1,T}^{e_j} \cdot \boldsymbol{t}_{j,T})|_{e_j} \equiv \zeta = L_1(\lambda_B - \lambda_A),$$

where $\zeta \in (-1, 1)$ is a parametrization of the edge e_j. Exploiting the recurrent definition of Legendre polynomials (1.40),

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

similarly as in the triangular case, we can write (oriented) *higher-order edge functions*

$$\psi_{k,T}^{e_j} = \frac{2k-1}{k}L_{k-1}(\lambda_B - \lambda_A)\psi_{1,T}^{e_j} - \frac{k-1}{k}L_{k-2}(\lambda_B - \lambda_A)\psi_{0,T}^{e_j}, \quad (2.77)$$

$$2 \le k \le p^{e_j};\ j = 1, \dots, 6.$$

The tangential component of functions $\psi^{e_j}_{k,T}$ now matches higher-order Legendre polynomials $L_k(\zeta)$ on the edge e_j, and vanishes on all other edges. In this way we add into the basis of $\boldsymbol{Q}_T$ edge functions for all edges e_j, $j = 1, \dots, 6$, up to the corresponding local order of approximation p^{e_j}.

Face functions, whose tangential components vanish in the standard sense on all faces but one, will be split into *edge-based* and *genuine*.

Recall the construction of local orientations on faces from the scalar case in Paragraph 2.2.5 – for each face we select a vertex v_A with the lowest local index, and by v_B, v_C denote its two remaining vertices in increasing order. For each face s_i, these three vertices determine affine coordinates $\lambda_A, \lambda_B, \lambda_C$, such that $\lambda_A(v_A) = \lambda_B(v_B) = \lambda_C(v_C) = 1$.

Let us forget about the original local orientation of edges for the construction of face functions. Consider a face s_i. Starting with its edge $e_j = v_A v_B$, shared by faces s_i, s_D, the product of the two corresponding affine coordinates λ_A, λ_B vanishes on all faces except for s_i, s_D, and gives a quadratic trace on the edge e_j. This trace can be extended to kth-order polynomials by multiplying it with $L_{k-2}(\lambda_B - \lambda_A)$, $k = 2, 3, \dots, p^{s_i}$. Multiplying this product further by the normal vector $\boldsymbol{n}_D$ to the face s_D, we eliminate its tangential component from s_D, and obtain the *edge-based face functions*

$$\psi^{s_i,e_j}_{k,T} = \lambda_A \lambda_B L_{k-2}(\lambda_B - \lambda_A)\alpha_i \boldsymbol{n}_D,\ k = 2, 3, \dots, p^{s_i}, \tag{2.78}$$

$j = 1, 2, \dots, 6$. The real coefficient α_i is chosen in such a way that projection of the normal vector $\alpha_i \boldsymbol{n}_D$, to a plane corresponding to the face s_i, has a unitary length. This normalization is necessary for future compatibility with triangular faces of prismatic elements. We proceed in the same way for the remaining two edges of the face s_i. The construction is illustrated in Figure 2.42.

Using the same notation, we also can easily define *genuine face functions*, which vanish identically on all faces but one:

$$\begin{aligned} \psi^{s_i,1}_{n_1,n_2,T} &= \lambda_A \lambda_B \lambda_C L_{n_1-1}(\lambda_B - \lambda_A) L_{n_2-1}(\lambda_A - \lambda_C) \boldsymbol{t}_{AB}, \\ \psi^{s_i,2}_{n_1,n_2,T} &= \lambda_A \lambda_B \lambda_C L_{n_1-1}(\lambda_B - \lambda_A) L_{n_2-1}(\lambda_A - \lambda_C) \boldsymbol{t}_{CA}, \end{aligned} \tag{2.79}$$

$1 \le n_1, n_2;\ n_1 + n_2 \le p^{s_i} - 1$. The symbols $\boldsymbol{t}_{AB}$, $\boldsymbol{t}_{AC}$ stand for unitary tangential vectors to the edges $e_{AB} = v_A v_B$, $e_{CA} = v_C v_A$, respectively. The construction is illustrated in Figure 2.43.

Hierarchic basis of the space $\boldsymbol{Q}_T$ will be completed by adding *bubble functions*, whose tangential component vanishes everywhere on the surface of the reference domain K_T. They will be again split into two groups – *face-based* and *genuine*. Face-based bubble functions are constructed in the same

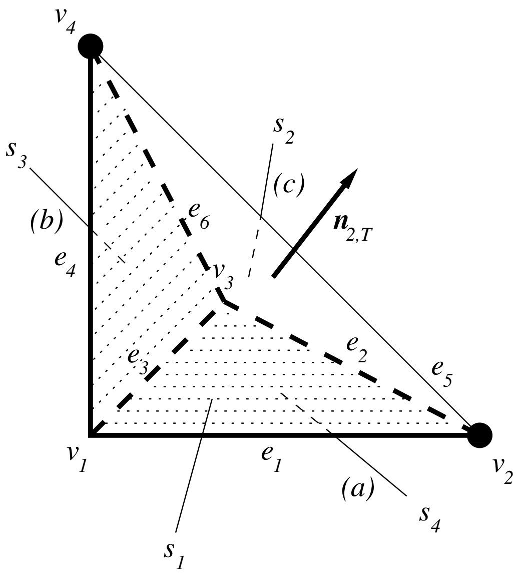

FIGURE 2.42: Consider the face $s_i = s_1$, and its edge $e_5 = v_2v_4$ ($= v_Bv_C$ locally on s_1). It is $\lambda_B = \lambda_{3,T}$, $\lambda_C = \lambda_{4,T}$. In addition to s_1, the edge e_5 also belongs to the face $s_D = s_2$. Multiplied by the normal vector $\boldsymbol{n}_D = \boldsymbol{n}_{2,T}$ and normalized by a real coefficient α_i, the product $\lambda_B\lambda_C L_{k-2}(\lambda_C - \lambda_B)$ yields a set of (oriented) edge-based face functions $\psi_{k,T}^{s_1,e_5}$, $2 \le k \le p^{s_1}$: ***(a)*,*(b)*** it vanishes completely on faces s_3, s_4 ($\lambda_{3,T}\lambda_{4,T} \equiv 0$ on s_3, s_4), and ***(c)*** its tangential component also vanishes everywhere on the face s_2 ($\boldsymbol{n}_{2,T} \cdot \boldsymbol{t} \equiv 0$ on s_2).

way as genuine face functions, except that the product $\lambda_A\lambda_B\lambda_C L_{n_1-1}(\lambda_B - \lambda_A)L_{n_2-1}(\lambda_A - \lambda_C)$ is now multiplied by the *normal vector* $\boldsymbol{n}_{i,T}$ to the face s_i, eliminating the only nonzero tangential component from the surface of the reference domain:

$$\psi_{n_1,n_2,T}^{b,s_i} = \lambda_A\lambda_B\lambda_C L_{n_1-1}(\lambda_B - \lambda_A)L_{n_2-1}(\lambda_A - \lambda_C)\boldsymbol{n}_{i,T}, \tag{2.80}$$

$1 \le n_1, n_2$; $n_1 + n_2 \le p^b - 1$, as depicted in Figure 2.44. Notice that the orientation of the faces no longer matters for bubble functions.

Finally, to the basis of $\boldsymbol{Q}_T$ we add *genuine bubble functions*

$$\psi_{n_1,n_2,n_3,T}^{b,m} = \varphi_{n_1,n_2,n_3,T}^{b}\boldsymbol{\xi}_m, \tag{2.81}$$

$1 \le n_1, n_2, n_3$; $n_1 + n_2 + n_3 \le p^b - 1$; $m = 1, \ldots, 3$, where $\varphi_{n_1,n_2,n_3,T}^{b}$ are scalar bubble functions defined in (2.40).

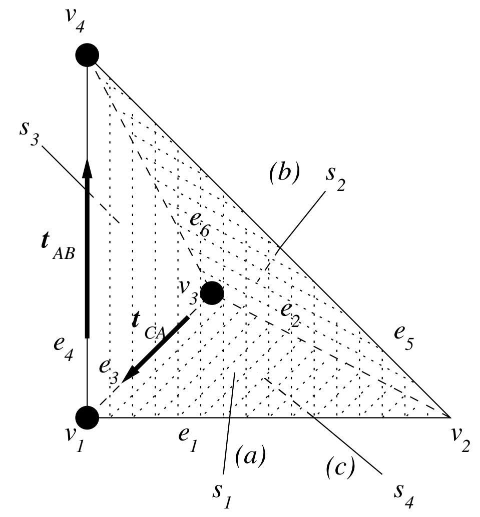

FIGURE 2.43: Consider the face $s_3 = v_1v_3v_4$. Locally on this face, $v_A = v_1, v_B = v_3$, $v_C = v_4$, $\lambda_A = \lambda_{2,T}$, $\lambda_B = \lambda_{1,T}$ and $\lambda_C = \lambda_{4,T}$. The product $\lambda_{2,T}\lambda_{1,T}\lambda_{4,T}L_{n_1-1}(\lambda_{1,t} - \lambda_{2,T})L_{n_2-1}(\lambda_{2,T} - \lambda_{4,T})$, multiplied by tangential vectors $\boldsymbol{t}_{AB}$ and $\boldsymbol{t}_{CA}$, respectively, yields (oriented) genuine face functions $\psi^{s_3,1}_{n_1,n_2,T}$ and $\psi^{s_3,2}_{n_1,n_2,T}$: (**a, b, c**) they vanish completely on the faces s_1, s_2, s_4 ($\lambda_{2,T}\lambda_{1,T}\lambda_{4,T} \equiv 0$ on s_1, s_2, s_4).

Numbers of vector-valued shape functions in the hierarchic basis of the space $\boldsymbol{Q}_T$ are summarized in Table 2.9.

TABLE 2.9: Vector-valued hierarchic shape functions of $\mathcal{K}_T^{\mathrm{curl}}$.

Node type	**Polyn.. order**	**Number of shape functions**	**Number of nodes**
Edge	always	$p^{e_j} + 1$	6
Edge-based face	$2 \le p^{s_i}$	$3(p^{s_i} - 1)$	4
Genuine face	$3 \le p^{s_i}$	$2(p^{s_i} - 2)(p^{s_i} - 1)/2$	4
Face-based interior	$3 \le p^b$	$4(p^b - 2)(p^b - 1)/2$	1
Genuine interior	$4 \le p^b$	$d(p^b - 3)(p^b - 2)(p^b - 1)/6$	1

PROPOSITION 2.10

Edge functions (2.75), (2.76), (2.77), face functions (2.78), (2.79), and bubble functions (2.80), (2.81), provide a complete basis of the space $\boldsymbol{Q}_T$, defined in (2.72).

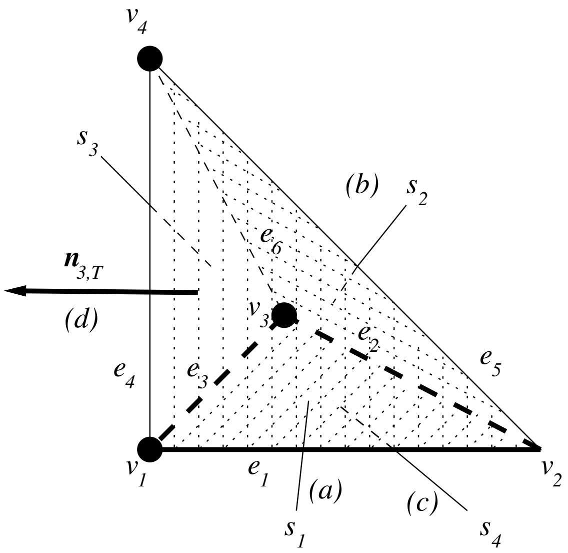

FIGURE 2.44: Consider the face s_3. The product $\lambda_{2,T}\lambda_{4,T}\lambda_{1,T}L_{n_1-1}(\lambda_{4,t} - \lambda_{2,T})L_{n_2-1}(\lambda_{2,T} - \lambda_{1,T})$, multiplied by the normal vector $\boldsymbol{n}_{3,T}$, yields face-based bubble functions $\psi^{b,s_3}_{n_1,n_2,T}$: (**a, b, c**) they vanish completely on the faces s_1, s_2, s_4; (***d***) their tangential component also vanishes everywhere on the face s_3.

PROOF The proof is left to the reader as an exercise. Follow the standard scheme – first check that all of the functions belong to the space $\boldsymbol{Q}_T$, then show that they are linearly independent, and finally that their number is equal to the dimension of $\boldsymbol{Q}_T$. $\square$

2.3.6 Prismatic master element $\mathcal{K}_P^{\mathrm{curl}}$

The last $\boldsymbol{H}$(curl)-conforming master element of arbitrary order $\mathcal{K}_P^{\mathrm{curl}}$ will be associated with the reference prismatic domain K_P (Figure 2.34).

As in the scalar case, we allow for anisotropic p-refinement of prismatic elements, and therefore consider local directional orders of approximation $p^{b,1}, p^{b,2}$ in the interior. The order $p^{b,1}$ corresponds to the plane $\xi_1\xi_2$ (we will designate this the *horizontal direction*), and $p^{b,2}$ to the *vertical direction* ξ_3. We have three quadrilateral faces s_i, $i = 1, \dots, 3$, which will be equipped with local directional orders of approximation $p^{s_i,1}, p^{s_i,2}$ (in horizontal and vertical directions, respectively). Triangular faces s_4, s_5 come with one local order of approximation p^{s_i} per face only. Standard local polynomial orders $p^{e_1}, \dots, p^{e_9}$ will be assigned to edges.

REMARK 2.36 Analogously as in the previous cases, all these nonuniform local orders of approximation come from a physical mesh element, and have to obey the *minimum rule* for $\boldsymbol{H}(\mathrm{curl})$-conforming approximations from Remark 2.24. $\square$

A finite element of arbitrary order on the reference domain K_P will be constructed in the conventional way as a triad $\mathcal{K}_P^{\mathrm{curl}} = (K_P, \boldsymbol{Q}_P, \Sigma_P^{\mathrm{curl}})$. We saw in Paragraph 2.2.6 that scalar polynomials φ on the reference prism $K_P = K_t \times K_a$ have a product form $\varphi \in \mathcal{R}_{m_1,m_2}(K_P)$, defined in (2.44). The way the gradient operator $\boldsymbol{\nabla}$ acts on a space of this form,

$$\mathcal{R}_{m_1+1,m_2+1}(K_P) \xrightarrow{\boldsymbol{\nabla}} \mathcal{R}_{m_1,m_2+1}(K_P) \times \mathcal{R}_{m_1,m_2+1}(K_P) \times \mathcal{R}_{m_1+1,m_2}(K_P), \tag{2.82}$$

determines the choice of an appropriate ancestor space

$$\begin{aligned} W_P = \big\{ & w \in \mathcal{R}_{p^{b,1}+1,p^{b,2}+1}(K_P);\ w|_{s_i} \in \mathcal{Q}_{p^{s_i,1}+1,p^{s_i,2}+1}(s_i) \text{ for } i = 1,2,3; \\ & w|_{s_i} \in \mathcal{P}_{p^{s_i}+1}(s_i) \text{ for } i = 4,5;\ w|_{e_j} \in \mathcal{P}_{p^{e_j}+1}(e_j),\ j = 1,\ldots,9\big\}, \end{aligned}$$

and suggests that a finite element of the form $\mathcal{K}_P^{\mathrm{curl}} = (K_P, \boldsymbol{Q}_P, \Sigma_P^{\mathrm{curl}})$ should be equipped with polynomial space

$$\begin{aligned} \boldsymbol{Q}_P = \big\{ & \boldsymbol{E} \in \mathcal{R}_{p^{b,1},p^{b,2}+1}(K_P) \times \mathcal{R}_{p^{b,1},p^{b,2}+1}(K_P) \times \mathcal{R}_{p^{b,1}+1,p^{b,2}}(K_P); \\ & \boldsymbol{E}_t|_{s_i} \in \mathcal{Q}_{p^{s_i,1},p^{s_i,2}+1}(s_i) \times \mathcal{Q}_{p^{s_i,1}+1,p^{s_i,2}}(s_i) \text{ for } i = 1,\ldots,3; \\ & \boldsymbol{E}_t|_{s_i} \in (\mathcal{P}_{p^{s_i}})^2(s_i) \text{ for } i = 4,5; \\ & \boldsymbol{E}\cdot\boldsymbol{t}|_{e_j} \in \mathcal{P}_{p^{e_j}}(e_j),\ j = 1,\ldots,9\big\}. \end{aligned} \tag{2.83}$$

Here again $\boldsymbol{E}_t|_{s_i} = \boldsymbol{E} - \boldsymbol{n}_i(\boldsymbol{E}\cdot\boldsymbol{n}_i)$ is the projection of the vector $\boldsymbol{E}$ on the face s_i.

REMARK 2.37 The product geometry $K_P = K_t \times K_a$ will facilitate the procedure in which the hierarchic shape functions are defined. Relation (2.82) suggests that the first two vector components may be constructed using products $\psi_t(\xi_1,\xi_2)l(\xi_3)$ of shape functions associated with the master triangle $\mathcal{K}_t^{\mathrm{curl}}$ in ξ_1,ξ_2, and the Lobatto shape functions in ξ_3. The third vector component will be constructed in the form of products $\varphi_t(\xi_1,\xi_2)L(\xi_3)$ of scalar shape functions associated with the master triangle $\mathcal{K}_t^1$ in ξ_1,ξ_2, and *original Legendre polynomials* in ξ_3. To simplify the notation, we will view all two-dimensional vectors corresponding to the master triangles $\mathcal{K}_t^1$ and $\mathcal{K}_t^{\mathrm{curl}}$ (normal and tangential vectors to its edges, vector-valued shape functions, etc.) as *three-dimensional vectors with zero third component.* These vectors will obviously be perpendicular to the third canonical vector $\boldsymbol{\xi}_3$. We will also use the fact that for each quadrilateral face s_i there is a unique matching edge e_i of the reference triangle K_t. $\square$

Edge functions $\psi^{e_j}_{k,P}$, $j = 1,\ldots,9$, $k = 0,\ldots,p^{e_j}$, will as usual be designed so that the tangential component of $\psi^{e_j}_{k,P}$ vanishes on all edges except for e_j, where it matches the Legendre polynomials $L_0, L_1, \ldots, L_{p^{e_j}}$. Recall reference triangle affine coordinates $\lambda_{1,t},\ldots,\lambda_{3,t}$ from (2.17), scalar vertex functions $\varphi^{v_j}_t$ defined in (2.20), and $\boldsymbol{H}$(curl)-conforming edge functions $\psi^{e_j}_{k,t}$, given by (2.57), (2.58) and (2.59).

Using our simplified notation, edge functions corresponding to edges $e_1,\ldots,$ e_3 (bottom of the reference prism K_P) can be written as

$$\psi^{e_j}_{k,P}(\xi_1,\xi_2,\xi_3) = \psi^{e_j}_{k,t}(\xi_1,\xi_2)l_0(\xi_3), \quad j = 1,\ldots,3,\ 0 \le k \le p^{e_j} \tag{2.84}$$

(e_j is used here both for edges of the reference prism K_P and reference triangle K_t). For vertical edges $e_4,\ldots,e_6$ we have edge functions

$$\begin{aligned}\psi^{e_4}_{k,P}(\xi_1,\xi_2,\xi_3) &= \varphi^{v_1}_t(\xi_1,\xi_2)L_k(\xi_3)\boldsymbol{\xi}_3, \quad 0 \le k \le p^{e_4},\\ \psi^{e_5}_{k,P}(\xi_1,\xi_2,\xi_3) &= \varphi^{v_2}_t(\xi_1,\xi_2)L_k(\xi_3)\boldsymbol{\xi}_3, \quad 0 \le k \le p^{e_5},\\ \psi^{e_6}_{k,P}(\xi_1,\xi_2,\xi_3) &= \varphi^{v_3}_t(\xi_1,\xi_2)L_k(\xi_3)\boldsymbol{\xi}_3, \quad 0 \le k \le p^{e_6}.\end{aligned} \tag{2.85}$$

The last three edges $e_7,\ldots,e_9$, corresponding to the top face s_5, are equipped with edge functions

$$\psi^{e_j}_{k,P}(\xi_1,\xi_2,\xi_3) = \psi^{e_{j-6}}_{k,t}(\xi_1,\xi_2)l_1(\xi_3), \quad j = 7,\ldots,9,\ 0 \le k \le p^{e_j}. \tag{2.86}$$

REMARK 2.38 Notice that edge functions corresponding to horizontal edges contribute only to the first two vector components, and edge functions associated with vertical edges only to the third one. All of them are linearly independent. As usual, nine lowest-order edge functions (corresponding to $k = 0$) form a complete lowest-order (Whitney) element. $\square$

Next we add to the basis of $\boldsymbol{Q}_P$ *face functions*. The primary function of prismatic elements is to connect hexahedral and tetrahedral elements in hybrid meshes, and therefore face functions associated with quadrilateral and triangular faces will be designed to be compatible with master elements $\mathcal{K}^{\mathrm{curl}}_B$ and $\mathcal{K}^{\mathrm{curl}}_T$, respectively. Recall the definition of local coordinate systems for quadrilateral and triangular faces from the scalar case.

Face functions for quadrilateral faces:

In the first step we generate face functions, the tangential component of which is nonzero only on a single quadrilateral face s_i, and only in the *horizontal* direction. We define

$$\psi^{s_i,1}_{n_1,n_2,P} = \psi^{e_i}_{n_1,t}(\xi_1,\xi_2)l_{n_2}(\xi_3), \quad 0 \le n_1 \le p^{s_i,1},\ 2 \le n_2 \le p^{s_i,2}+1. \quad (2.87)$$

Here we use the same symbol for two matching edges e_i of the reference triangle and prism. These functions completely vanish on both triangular faces s_4 and s_5, since the Lobatto shape functions $l_i(\pm 1) = 0$, $i = 2, 3, \ldots$. The tangential component of functions $\psi^{s_i,1}_{n_1,n_2,P}$ in the vertical direction ξ_3 vanishes everywhere due to their zero third vector component. The rest immediately follows from properties of $\boldsymbol{H}(\mathrm{curl})$-conforming edge functions $\psi^{e_i}_{n_1,t}$ associated with the master triangle.

Remaining face functions for quadrilateral faces will be designed to have a nonzero tangential component only on a single quadrilateral face s_i, and only in the *vertical* direction:

$$\psi^{s_i,2}_{n_1,n_2,P} = \varphi^{e_i}_{n_1,t}(\xi_1,\xi_2)L_{n_2}(\xi_3)\boldsymbol{\xi}_3, \quad 2 \le n_1 \le p^{s_i,1}+1,\ 0 \le n_2 \le p^{s_i,2}. \quad (2.88)$$

Scalar edge functions on the master triangle $\mathcal{K}^1_t$, $\varphi^{e_i}_{n_1,t}(\xi_1,\xi_2)$, were defined in (2.21). These functions make $\psi^{s_i,2}_{n_1,n_2,P}$ vanish completely on the remaining vertical faces, and $\boldsymbol{\xi}_3$ ensures that their tangential components vanish moreover on horizontal faces s_4, s_5.

REMARK 2.39 Notice that tangential components of face functions belonging to the above two sets (2.87), (2.88) *exactly match* appropriate tangential components of face functions (2.67), corresponding to the master brick $\mathcal{K}^{\mathrm{curl}}_B$. Use relation (1.52) and definition of scalar triangular edge functions (2.21). Also the *numbers* of face functions are the same, when identical directional orders of approximation are considered. Notice, too, that limits of directional polynomial orders of functions defined in (2.87), (2.88) exactly correspond to the definition (2.83) of the space $\boldsymbol{Q}_P$. Hence the face functions are suitable for the design of hybrid tetrahedral-hexahedral meshes. $\square$

Face functions for triangular faces s_4, s_5 will be constructed in a similar way to those for the master tetrahedron $\mathcal{K}^{\mathrm{curl}}_T$. First we assign to these faces the same local orientation as in the scalar case in Paragraph 2.2.6.

There are three *edge-based triangular face functions* for each face s_i, $i = 4, 5$: Put $l(\xi_3) = l_0(\xi_3)$ if $i = 4$, and $l(\xi_3) = l_1(\xi_3)$ otherwise. Let us begin with an edge $e_j = v_A v_B$ of the face s_i, which is also shared by another face s_D. The product $\lambda_A\lambda_B l(\xi_3)$, $\lambda_A(v_A) = \lambda_B(v_B) = 1$ vanishes on all faces except for s_i, s_D, and gives a quadratic trace on e_j. This trace is again extended to kth-order polynomials by multiplying it with $L_{k-2}(\lambda_B - \lambda_A)$, $k = 2, 3, \ldots, p^{s_i}$. We use the normal vector $\boldsymbol{n}_D$ to eliminate the tangential component from the face s_D, and define

$$\psi^{s_i,e_j}_{n_1,P} = \lambda_A \lambda_B L_{n_1-2}(\lambda_B - \lambda_A) l(\xi_3) \boldsymbol{n}_D,\ n_1 = 2, 3, \dots, p^{s_i}. \qquad (2.89)$$

The construction is illustrated in Figure 2.45.

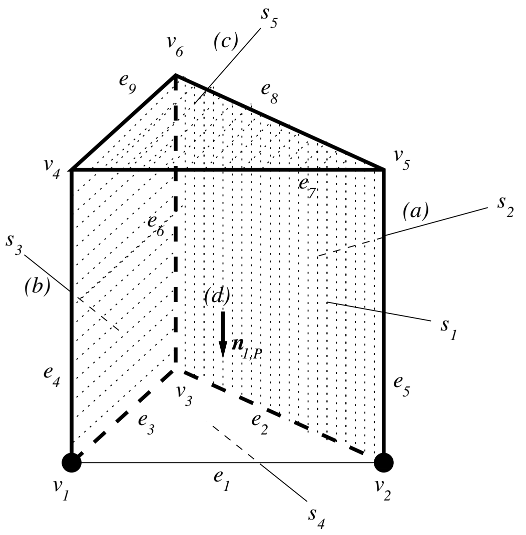

FIGURE 2.45: Consider the triangular face s_4 and its edge e_1, which matches edge e_1 of the reference triangle K_t. Multiplied by $l_0(\xi_3)$, the edge functions $\psi^{e_1}_{n_1,t}$, $2 \le n_1 \le p^{s_4}$, yield a set of edge-based face functions $\psi^{s_4,e_1}_{n_1,P}$, $2 \le n_1 \le p^{s_4}$: ***(a), (b), (c)*** they vanish completely on faces s_2, s_3 ($\psi^{e_1}_{n_1,t} \equiv 0$ on s_2, s_3) and s_5 ($l_0(1) = 0$); ***(d)*** the tangential component vanishes also on face s_1.

Genuine triangular face functions will also be constructed in a way similar to the tetrahedral case:

$$\begin{aligned}
\psi^{s_i,1}_{n_1,n_2,P} &= \lambda_A \lambda_B \lambda_C L_{n_1-1}(\lambda_B - \lambda_A) L_{n_2-1}(\lambda_A - \lambda_C) l(\xi_3) \boldsymbol{t}_{AB}, \qquad (2.90)\\
\psi^{s_i,2}_{n_1,n_2,P} &= \lambda_A \lambda_B \lambda_C L_{n_1-1}(\lambda_B - \lambda_A) L_{n_2-1}(\lambda_A - \lambda_C) l(\xi_3) \boldsymbol{t}_{CA},
\end{aligned}$$

$1 \le n_1, n_2;\ n_1 + n_2 \le p^{s_i} - 1$. The symbols $\boldsymbol{t}_{AB}$, $\boldsymbol{t}_{CA}$ stand for unitary tangential vectors to the edges $e_{AB} = v_A v_B$, $e_{CA} = v_C v_A$, respectively. The construction is illustrated in Figure 2.46.

REMARK 2.40 Notice that tangential components of face functions, belonging to the above two sets (2.89), (2.90), *exactly match* corresponding tan-

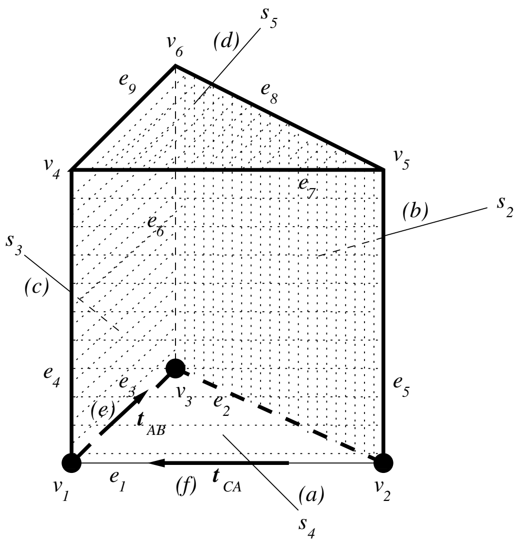

FIGURE 2.46: Again consider face s_4. Multiplied by tangential vectors $\boldsymbol{t}_{AB}$ and $\boldsymbol{t}_{CA}$, the product $\lambda_A\lambda_B\lambda_C L_{n_1-1}(\lambda_B-\lambda_A)L_{n_2-1}(\lambda_A-\lambda_C)l_0(\xi_3)$ gives rise to genuine face functions $\psi^{s_4,1}_{n_1,n_2,P}, \psi^{s_4,2}_{n_1,n_2,P}$, respectively: (**a, b, c, d**) they vanish completely on all faces except for s_4; (***e, f***) their tangential component is generally nonzero on face s_4.

gential components of face functions (2.78), (2.79) of the master tetrahedron $\mathcal{K}_T^{\text{curl}}$. Recall the role of the real parameter α_i, which was introduced in (2.78). In Chapter 3 we will describe in detail how prismatic elements are used to connect tetrahedra and bricks in hybrid $\boldsymbol{H}(\text{curl})$-conforming meshes. $\square$

The basis of the space $\boldsymbol{Q}_P$ will be completed by adding *bubble functions*, whose tangential component vanishes everywhere on the surface of the reference prism K_P.

First let us complete the part of the basis corresponding to the two first vector components. This is done by multiplying edge-based and genuine bubble functions associated with the master triangle $\mathcal{K}_t^{\text{curl}}$ by the Lobatto shape functions in ξ_3. Thus we obtain *quadrilateral-face-based bubble functions*

$$\psi^{b,s_i}_{n_1,n_2,P} = \psi^{b,e_i}_{n_1,t}(\xi_1,\xi_2)l_{n_2}(\xi_3), \quad i=1,\ldots,3, \tag{2.91}$$

$2 \le n_1 \le p^{b,1}$, $2 \le n_3 \le p^{b,2}+1$, and *genuine bubble functions*

$$\psi^{b,m}_{n_1,n_2,n_3,P} = \psi^{b,m}_{n_1,n_2,t}(\xi_1,\xi_2)l_{n_3}(\xi_3), \tag{2.92}$$

$1 \le n_1, n_2$; $n_1+n_2 \le p^{b,1}-1$; $2 \le n_3 \le p^{b,2}+1$; $m=1,2$.

Finally we design *triangular-face-based* bubble functions, which are only nonzero in their third component. This can be done by multiplying scalar bubble functions associated with the master triangle $\mathcal{K}_t^1$ in ξ_1, ξ_2 by original Legendre polynomials in ξ_3:

$$\psi^{b,3}_{n_1,n_2,n_3,P} = \varphi^{b}_{n_1,n_2,t}(\xi_1,\xi_2)L_{n_3}(\xi_3)\boldsymbol{\xi}_3, \tag{2.93}$$

$1 \le n_1, n_2,\ n_1 + n_2 \le p^{b,1},\ 0 \le n_3 \le p^{b,2}$. With this, the basis of the space $\boldsymbol{Q}_P$ is complete, and design of the finite element $\mathcal{K}_P^{\mathrm{curl}}$ finished.

Numbers of vector-valued shape functions in the hierarchic basis of the space $\boldsymbol{Q}_P$ are summarized in Table 2.10.

TABLE 2.10: Vector-valued hierarchic shape functions of $\mathcal{K}_P^{\mathrm{curl}}$.

Node type	**Polynomial order**	**Number of shape functions**	**Num. of nodes**
Edge	always	$p^{e_j} + 1$	9
Quad. face horiz.	$1 \le p^{s_i,2}$	$(p^{s_i,1} + 1)p^{s_i,2}$	3
Quad. face vert.	$1 \le p^{s_i,1}$	$(p^{s_i,2} + 1)p^{s_i,1}$	3
Tri. edge face	$2 \le p^{s_i}$	$3(p^{s_i} - 1)$	2
Tri. face genuine	$3 \le p^{s_i}$	$(p^{s_i} - 1)(p^{s_i} - 2)$	2
Quad. face bubble	$2 \le p^{b,1}, 1 \le p^{b,2}$	$3(p^{b,1} - 1)p^{b,2}$	1
Genuine bubble	$3 \le p^{b,1}, 1 \le p^{b,2}$	$(p^{b,1} - 1)(p^{b,1} - 2)p^{b,2}$	1
Tri. face bubble	$2 \le p^{b,1}$	$(p^{b,1} - 1)p^{b,1}(p^{b,2} + 1)/2$	1

PROPOSITION 2.11
Edge functions (2.84), (2.85), (2.86), face functions (2.87), (2.88), (2.89), (2.90), and bubble functions (2.91), (2.92) and (2.93) constitute a hierarchic basis of the space $\boldsymbol{Q}_P$ defined in (2.83).

PROOF Let us devote more attention to this case, as the master prism $\mathcal{K}_P^{\mathrm{curl}}$ is one of the more complicated master elements that we deal with.

It is easy to see that all of the aforementioned shape functions belong to the vector-valued polynomial space $\boldsymbol{Q}_P$.

Next we verify that all shape functions are linearly independent. Edge functions associated with horizontal edges have a zero third component, while edge functions belonging to vertical edges only have a nonzero third component. Further, edge functions corresponding to horizontal edges e_1, e_2, e_3 on the bottom completely vanish on the top face s_5 and vice versa. Thus, functions from these three groups are linearly independent, and so are functions within each group, since traces of their tangential components match Legendre polynomials on appropriate edges.

Horizontal quadrilateral face functions (2.87) are linearly independent of the *vertical* ones, defined by (2.88), since they again reside in different vector-components. Linear independence within each group follows from the linear independence of the functions used for their definition.

Edge-based triangular face functions are linearly independent of the genuine ones because the latter vanish completely on all quadrilateral faces, with obvious results. Also, linear independence of bubble functions follows logically from the properties of scalar and vector-valued functions used for their definition.

The tedious step, as always, is to verify that the number of basis functions exactly matches the dimension of the space $\boldsymbol{Q}_P$. Let us start with a simplified situation, in which the element is generally anisotropically p-refined, but local orders of approximation on faces and edges are *not reduced* by local nonuniform distribution of the order of approximation in the physical mesh. Thus we have $p^{b,1} = p^{s_1,1} = \ldots = p^{s_3,1} = p^{s_4} = p^{s_5} = p^{e_1} = \ldots = p^{e_3} = p^{e_7} = \ldots = p^{e_9}$, and $p^{b,2} = p^{s_1,2} = \ldots = p^{s_3,2} = p^{e_4} = \ldots = p^{e_6}$. After a brief computation, we obtain that

$$\dim(\boldsymbol{Q}_P) = \underbrace{(p^{b,1}+1)(p^{b,1}+2)}_{A}\underbrace{(p^{b,2}+2)}_{B} + \underbrace{\frac{(p^{b,1}+2)(p^{b,1}+3)}{2}}_{C}\underbrace{(p^{b,2}+1)}_{D},$$

where A is the dimension of polynomial space associated with the master triangle $\mathcal{K}_t^{\mathrm{curl}}$ of order $p^{b,1}$, B is the number of the Lobatto shape functions $l_0, \ldots, l_{p^{b,2}+1}$, C is the dimension of scalar polynomial space associated with the master triangle $\mathcal{K}_t^1$ of order $p^{b,1}+1$, and finally, D corresponds to the dimension of one-dimensional polynomial space generated by Legendre polynomials $L_0, \ldots, L_{p^{b,2}}$. Notice that numbers A, B correspond to functions with zero third components, and C, D to functions whose two first components are zero. Now let us compute the basis functions:

1. Functions with zero third component:
   (a) Horizontal edges contribute $2 \cdot 3(p^{b,1}+1)$ edge functions (2.84), (2.86),
   (b) quadrilateral faces yield $3(p^{b,1}+1)p^{b,2}$ (horizontal) face functions (2.87),
   (c) and we have $2 \cdot 3(p^{b,1}-1)$ edge-based triangular face functions (2.89).
   (d) Further there are $3(p^{b,1}-1)p^{b,2}$ quadrilateral face-based bubble functions (2.91),

(e) $2(p^{b,1}-1)(p^{b,1}-2)$ genuine triangular face functions (2.90), and finally

(f) $(p^{b,1}-1)(p^{b,1}-2)p^{b,2}$ genuine bubble functions (2.92).

2. Functions whose two first components are zero:

   (a) Vertical edges contribute $3(p^{b,2}+1)$ edge functions (2.85),

   (b) quadrilateral faces yield $3(p^{b,1}+1)p^{b,2}$ (vertical) face functions (2.88),

   (c) and there are $(p^{b,1}-1)p^{b,1}(p^{b,2}+1)/2$ triangular face-based bubble functions (2.93).

Summing up entries in the the first part, we arrive at

$$(p^{b,1}+1)(p^{b,1}+2)(p^{b,2}+2).$$

The second part involves

$$\frac{(p^{b,1}+2)(p^{b,1}+3)}{2}(p^{b,2}+1)$$

shape functions. Thus, the number of shape functions exactly matches the dimension of the space $\boldsymbol{Q}_P$.

All that remains to be done is to verify that this is also valid when local orders of approximation on edges and faces reduce. This can already be easily seen, taking one local order of approximation after another, reducing it and observing that the reduction of dimension of the space $\boldsymbol{Q}_P$ exactly corresponds to the reduction of the number of corresponding shape functions. With this, the proof is complete. $\square$

2.4 $\boldsymbol{H}(\mathrm{div})$-conforming approximations

With the experience that we gained during the construction of $\boldsymbol{H}(\mathrm{curl})$-conforming finite elements of arbitrary order in the previous section, the design of $\boldsymbol{H}(\mathrm{div})$-conforming elements will be a simple exercise.

2.4.1 De Rham diagram and finite elements in $\boldsymbol{H}(\mathrm{div})$

Also in $\boldsymbol{H}(\mathrm{div})$ we will consider the finite elements within the general framework of the De Rham diagram (2.2),

$$H^1 \xrightarrow{\nabla\times} \boldsymbol{H}(\mathrm{div}) \xrightarrow{\nabla\cdot} L^2 \quad \text{(2D form)},$$

and (2.3),

$$H^1 \xrightarrow{\nabla} \boldsymbol{H}(\mathrm{curl}) \xrightarrow{\nabla\times} \boldsymbol{H}(\mathrm{div}) \xrightarrow{\nabla\cdot} L^2 \quad \text{(3D form)}.$$

Conformity requirements reduce to *continuity of normal component of approximation across element interfaces.*

REMARK 2.41 (Similarity of spaces $\boldsymbol{H}$(curl) and $\boldsymbol{H}$(div) in 2D) Due to the similarity of operators $\nabla\times = (-\partial/\partial\xi_2, \partial/\partial\xi_1)$ and $\nabla = (\partial/\partial\xi_1, \partial/\partial\xi_2)$ in 2D, the spaces $\boldsymbol{H}$(curl) and $\boldsymbol{H}$(div) have much in common. In particular, normal direction is (up to the sign factor) *the unique complementary direction* to the tangential one and vice versa. Therefore, $\boldsymbol{H}$(div)-conforming shape functions can easily be derived from the appropriate $\boldsymbol{H}$(curl)-conforming ones by switching these two directions. $\square$

REMARK 2.42 (Reduced conformity requirements in 3D) In three spatial dimensions the situation will be easier than in the $\boldsymbol{H}$(curl)-conforming case, since neither vertices nor edges are constrained by conformity requirements in the space $\boldsymbol{H}$(div) (see Paragraph 1.1.4). Hierarchic vector-valued shape functions will contain *neither vertex nor edge functions.* $\square$

2.4.2 Quadrilateral master element $\mathcal{K}_q^{\mathrm{div}}$

Let us begin with the simplest master element of arbitrary order, $\mathcal{K}_q^{\mathrm{div}}$, on the reference quadrilateral domain K_q (Figure 2.1).

As usual, we allow for its anisotropic p-refinement, and therefore consider local directional orders of approximation $p^{b,1}, p^{b,2}$ in the element interior (corresponding to axial directions ξ_1 and ξ_2, respectively). Local orders $p^{e_1}, \ldots, p^{e_4}$, are assigned to edges $e_1, \ldots, e_4$.

REMARK 2.43 (Minimum rules in $\boldsymbol{H}$(div)) These nonuniform local orders of approximation come from a physical mesh element, and at this time they have to obey the *minimum rule for $\boldsymbol{H}$(div)-conforming approximations*: polynomial orders *of normal components* of approximation on physical mesh edges must not exceed appropriate local directional orders in the interior of adjacent elements. $\square$

Locally on the reference domain Remark 2.43 yields that

$$\begin{aligned} p^{e_1}, p^{e_2} &\leq p^{b_2}, \\ p^{e_3}, p^{e_4} &\leq p^{b_1}. \end{aligned}$$

REMARK 2.44 The **curl** operator $\nabla\times$, representing an adequate portion of the De Rham diagram (2.2), transforms scalar monomials from the space W_q by

$$\xi_1^{i+1}\xi_2^{j+1} \overset{\nabla\times}{\longrightarrow} \left(-(j+1)\xi_1^{i+1}\xi_2^{j}, (i+1)\xi_1^{i}\xi_2^{j+1}\right).$$

$\square$

According to Remark 2.44 the De Rham diagram suggests that a finite element of the form $\mathcal{K}_q^{\mathrm{div}} = (K_q, \boldsymbol{V}_q, \Sigma_q^{\mathrm{div}})$ needs to be equipped with polynomial space

$$\boldsymbol{V}_q = \left\{\boldsymbol{v} \in \mathcal{Q}_{p^{b,1}+1,p^{b,2}} \times \mathcal{Q}_{p^{b,1},p^{b,2}+1};\ \boldsymbol{v}\cdot\boldsymbol{n}|_{e_j} \in \mathcal{P}_{p^{e_j}}(e_j),\ j=1,\dots,4\right\}. \tag{2.94}$$

The ancestor scalar finite element space is the same as in the $\boldsymbol{H}$(curl)-conforming case, namely

$$W_q = \left\{w \in \mathcal{Q}_{p^{b,1}+1,p^{b,2}+1};\ w|_{e_j} \in \mathcal{P}_{p^{e_j}+1}(e_j),\ j=1,\dots,4\right\}.$$

As usual let us now accomplish the design of the finite element $\mathcal{K}_q^{\mathrm{div}}$ by identifying the set of degrees of freedom Σ_q^{div} via a suitable hierarchic basis of the space $\boldsymbol{V}_q$. We will see that the connection of this case with the two-dimensional $\boldsymbol{H}$(curl)-conforming case is straightforward. The hierarchic basis will again comprise edge and bubble functions.

Traces of normal component of *edge functions* $\gamma_{k,q}^{e_i}$ will coincide with Legendre polynomials L_k, $k=0,\dots,p^{e_i}$ on the appropriate edge e_i, $i=1,\dots,4$, and vanish on all remaining ones,

$$\begin{aligned}
\gamma_{k,q}^{e_1} &= l_0(\xi_1)L_k(\xi_2)\boldsymbol{\xi}_1, \quad 0\le k\le p^{e_1},\\
\gamma_{k,q}^{e_2} &= l_1(\xi_1)L_k(\xi_2)\boldsymbol{\xi}_1, \quad 0\le k\le p^{e_2},\\
\gamma_{k,q}^{e_3} &= L_k(\xi_1)l_0(\xi_2)\boldsymbol{\xi}_2, \quad 0\le k\le p^{e_3},\\
\gamma_{k,q}^{e_4} &= L_k(\xi_1)l_1(\xi_2)\boldsymbol{\xi}_2, \quad 0\le k\le p^{e_4},
\end{aligned} \tag{2.95}$$

while *bubble functions* will be designed in such a way that their normal component vanishes on all edges,

$$\begin{aligned}
\gamma_{n_1,n_2,q}^{b,1} &= l_{n_1}(\xi_1)L_{n_2}(\xi_2)\boldsymbol{\xi}_1, \quad 2\le n_1\le p^{b,1}+1,\ 0\le n_2\le p^{b,2},\\
\gamma_{n_1,n_2,q}^{b,2} &= L_{n_1}(\xi_1)l_{n_2}(\xi_2)\boldsymbol{\xi}_2, \quad 0\le n_1\le p^{b,1},\ 2\le n_2\le p^{b,2}+1.
\end{aligned} \tag{2.96}$$

REMARK 2.45 Perhaps it is worth mentioning that the edge functions are *oriented* according to canonical vectors $\boldsymbol{\xi}_1, \boldsymbol{\xi}_2$. This fact will be used for the design of globally $\boldsymbol{H}(\mathrm{div})$-conforming edge functions in the physical mesh in Chapter 3. $\square$

The four lowest-order edge functions $\gamma_{0,q}^{e_1}, \dots, \gamma_{0,q}^{e_4}$, (*Whitney functions*) again complete a lowest order element. All of the above shape functions are obviously linearly independent.

Numbers of vector-valued shape functions in the hierarchic basis of the space $\boldsymbol{V}_q$ are summarized in Table 2.11.

TABLE 2.11: Vector-valued hierarchic shape functions of $\mathcal{K}_q^{\mathrm{div}}$.

Node type	**Polynomial order**	**Number of shape functions**	**Number of nodes**
Edge	always	$p^{e_j} + 1$	4
Interior	$1 \le p^{b,1}$ or $1 \le p^{b,2}$	$(p^{b,2}+1)p^{b,1} + p^{b,2}(p^{b,1}+1)$	1

PROPOSITION 2.12

Vector-valued shape functions (2.95) and (2.96) constitute a hierarchic basis of the space $\mathbf{V}_q$, defined in (2.94).

PROOF The same as in the $\boldsymbol{H}(\mathrm{curl})$-conforming case. $\square$

2.4.3 Triangular master element $\mathcal{K}_t^{\mathrm{div}}$

In this paragraph we will design a master element of arbitrary order $\mathcal{K}_t^{\mathrm{div}}$ on the reference triangular domain K_t (Figure 2.13).

Consider a local order of approximation p^b in element interior, and local orders p^{e_j} on edges, $j = 1, \dots, 3$.

REMARK 2.46 The minimum rule for $\boldsymbol{H}(\mathrm{div})$-conforming approximation (Remark 2.43) yields that $p^{e_i} \le p^b$ for all $i = 1, \dots, 3$. $\square$

In harmony with the De Rham diagram (2.2), a finite element of the form $\mathcal{K}_t^{\mathrm{div}} = (K_t, \boldsymbol{V}_t, \Sigma_t^{\mathrm{div}})$ will be equipped with polynomial space

$$\boldsymbol{V}_t = \left\{ \boldsymbol{v} \in (\mathcal{P}_{p^b})^2(K_t);\ \boldsymbol{v} \cdot \boldsymbol{n}|_{e_j} \in \mathcal{P}_{p^{e_j}}(e_j),\ j = 1, \dots, 3 \right\}. \tag{2.97}$$

The ancestor space W_t is the same as in the $\boldsymbol{H}(\mathrm{curl})$-conforming case,

$$W_t = \left\{ w \in \mathcal{P}_{p^b+1}(K_t);\ w|_{e_j} \in \mathcal{P}_{p^{e_j}+1}(e_j)\ j = 1, \dots, 3 \right\}.$$

REMARK 2.47 Recall the relation

$$\boldsymbol{n}_{i,t} = \frac{\nabla\lambda_{i,t}}{|\nabla\lambda_{i,t}|}, \quad i = 1,\dots,3, \tag{2.98}$$

defining unitary normal vectors to edges. □

Exchanging tangential and normal direction in the definition of functions (2.57) from the $\boldsymbol{H}$(curl)-conforming case, we easily arrive at *Whitney functions*

$$\begin{aligned}
\gamma_{0,t}^{e_1} &= \frac{\lambda_{3,t}\boldsymbol{t}_{2,t}}{\boldsymbol{t}_{2,t}\cdot\boldsymbol{n}_{1,t}} + \frac{\lambda_{2,t}\boldsymbol{t}_{3,t}}{\boldsymbol{t}_{3,t}\cdot\boldsymbol{n}_{1,t}}, \\
\gamma_{0,t}^{e_2} &= \frac{\lambda_{1,t}\boldsymbol{t}_{3,t}}{\boldsymbol{t}_{3,t}\cdot\boldsymbol{n}_{2,t}} + \frac{\lambda_{3,t}\boldsymbol{t}_{1,t}}{\boldsymbol{t}_{1,t}\cdot\boldsymbol{n}_{2,t}}, \\
\gamma_{0,t}^{e_3} &= \frac{\lambda_{2,t}\boldsymbol{t}_{1,t}}{\boldsymbol{t}_{1,t}\cdot\boldsymbol{n}_{3,t}} + \frac{\lambda_{1,t}\boldsymbol{t}_{2,t}}{\boldsymbol{t}_{2,t}\cdot\boldsymbol{n}_{3,t}},
\end{aligned} \tag{2.99}$$

(whose normal component vanishes on all edges except for the one where it coincides with the Legendre polynomial $L_0 \equiv 1$). Analogously we obtain *linear edge functions*

$$\begin{aligned}
\gamma_{1,t}^{e_1} &= \frac{\lambda_{3,t}\boldsymbol{t}_{2,t}}{\boldsymbol{t}_{2,t}\cdot\boldsymbol{n}_{1,t}} - \frac{\lambda_{2,t}\boldsymbol{t}_{3,t}}{\boldsymbol{t}_{3,t}\cdot\boldsymbol{n}_{1,t}}, \quad p^{e_1} \ge 1, \\
\gamma_{1,t}^{e_2} &= \frac{\lambda_{1,t}\boldsymbol{t}_{3,t}}{\boldsymbol{t}_{3,t}\cdot\boldsymbol{n}_{2,t}} - \frac{\lambda_{3,t}\boldsymbol{t}_{1,t}}{\boldsymbol{t}_{1,t}\cdot\boldsymbol{n}_{2,t}}, \quad p^{e_2} \ge 1, \\
\gamma_{1,t}^{e_3} &= \frac{\lambda_{2,t}\boldsymbol{t}_{1,t}}{\boldsymbol{t}_{1,t}\cdot\boldsymbol{n}_{3,t}} - \frac{\lambda_{1,t}\boldsymbol{t}_{2,t}}{\boldsymbol{t}_{2,t}\cdot\boldsymbol{n}_{3,t}}, \quad p^{e_3} \ge 1.
\end{aligned} \tag{2.100}$$

Higher-order edge functions

$$\begin{aligned}
\gamma_{k,t}^{e_1} &= \frac{2k-1}{k}L_{k-1}(\lambda_{3,t}-\lambda_{2,t})\gamma_{1,t}^{e_1} - \frac{k-1}{k}L_{k-2}(\lambda_{3,t}-\lambda_{2,t})\gamma_{0,t}^{e_1}, \\
&\quad 2 \le k \le p^{e_1}; \\
\gamma_{k,t}^{e_2} &= \frac{2k-1}{k}L_{k-1}(\lambda_{1,t}-\lambda_{3,t})\gamma_{1,t}^{e_2} - \frac{k-1}{k}L_{k-2}(\lambda_{1,t}-\lambda_{3,t})\gamma_{0,t}^{e_2}, \\
&\quad 2 \le k \le p^{e_2}; \\
\gamma_{k,t}^{e_3} &= \frac{2k-1}{k}L_{k-1}(\lambda_{2,t}-\lambda_{1,t})\gamma_{1,t}^{e_3} - \frac{k-1}{k}L_{k-2}(\lambda_{2,t}-\lambda_{1,t})\gamma_{0,t}^{e_3}. \\
&\quad 2 \le k \le p^{e_3},
\end{aligned} \tag{2.101}$$

are again composed from Whitney and linear edge functions, using the recurrent definition of Legendre polynomials (1.40).

Bubble functions, whose normal component vanishes on all edges, will be constructed as edge-based and genuine. *Edge-based* bubble functions will have the form

$$
\begin{aligned}
\gamma_{k,t}^{b,e_1} &= \lambda_{3,t}\lambda_{2,t}L_{k-2}(\lambda_{3,t}-\lambda_{2,t})\boldsymbol{t}_{1,t}, \quad 2 \le k \le p^b, \\
\gamma_{k,t}^{b,e_2} &= \lambda_{1,t}\lambda_{3,t}L_{k-2}(\lambda_{1,t}-\lambda_{3,t})\boldsymbol{t}_{2,t}, \quad 2 \le k \le p^b, \\
\gamma_{k,t}^{b,e_3} &= \lambda_{2,t}\lambda_{1,t}L_{k-2}(\lambda_{2,t}-\lambda_{1,t})\boldsymbol{t}_{3,t}, \quad 2 \le k \le p^b,
\end{aligned}
\tag{2.102}
$$

analogous to (2.60), and *genuine bubble functions* will be written as

$$
\begin{aligned}
\gamma_{n_1,n_2,t}^{b,1} &= \lambda_{1,t}\lambda_{2,t}\lambda_{3,t}L_{n_1-1}(\lambda_{3,t}-\lambda_{2,t})L_{n_2-1}(\lambda_{2,t}-\lambda_{1,t})\boldsymbol{\xi}_1, \\
\gamma_{n_1,n_2,t}^{b,2} &= \lambda_{1,t}\lambda_{2,t}\lambda_{3,t}L_{n_1-1}(\lambda_{3,t}-\lambda_{2,t})L_{n_2-1}(\lambda_{2,t}-\lambda_{1,t})\boldsymbol{\xi}_2,
\end{aligned}
\tag{2.103}
$$

$1 \le n_1, n_2;\ n_1 + n_2 \le p^b - 1$, as suggested by (2.61).

Numbers of vector-valued shape functions in the hierarchic basis of the space $\boldsymbol{V}_t$ are summarized in Table 2.7.

TABLE 2.12: Vector-valued hierarchic shape functions of $\mathcal{K}_t^{\mathrm{div}}$.

Node type	**Polynomial order**	**Number of shape functions**	**Number of nodes**
Edge	always	$p^{e_j} + 1$	3
Edge-based interior	$2 \le p^b$	$3(p^b - 1)$	1
Genuine interior	$3 \le p^b$	$(p^b - 1)(p^b - 2)$	1

PROPOSITION 2.13

Whitney functions (2.99), linear edge functions (2.100), higher-order edge functions (2.101), and bubble functions (2.102), (2.103), constitute a hierarchic basis of the space $\boldsymbol{V}_t$, defined in (2.97).

PROOF The same as in the $\boldsymbol{H}$(curl)-conforming case. ∎

REMARK 2.48 Notice that traces of normal components of both the edge functions (2.95) associated with the master quadrilateral $\mathcal{K}_q^{\mathrm{div}}$, and edge functions (2.99), (2.100) and (2.101) of the master triangle $\mathcal{K}_t^{\mathrm{div}}$ match the Legendre polynomials $L_0, L_1, \ldots$. Hence, as in the $\boldsymbol{H}$(curl)-conforming case, the hierarchic bases of master elements $\mathcal{K}_q^{\mathrm{div}}$ and $\mathcal{K}_t^{\mathrm{div}}$ offer the possibility of combining quadrilateral and triangular elements in hybrid meshes. This issue will be discussed in more detail in Chapter 3. $\square$

2.4.4 Brick master element $\mathcal{K}_B^{\mathrm{div}}$

In this paragraph we will present a master element of arbitrary order $\mathcal{K}_B^{\mathrm{div}}$ on the reference brick domain K_B (Figure 2.26).

To allow for anisotropic p-refinement of brick elements, we consider standard local orders of approximation $p^{b,1}, p^{b,2}$ and $p^{b,3}$ in the element interior (corresponding to coordinate axes in lexicographic order). Local directional orders $p^{s_i,1}, p^{s_i,2}$ are assigned to faces s_i, $i = 1, \ldots, 6$. No local orders for edges are relevant, as edges are not constrained by $\boldsymbol{H}(\mathrm{div})$-conformity rules.

REMARK 2.49 The minimum rule for $\boldsymbol{H}(\mathrm{div})$-conforming approximations (Remark 2.43) yields that directional polynomial orders of *normal components* of approximation on physical mesh faces are limited by corresponding directional orders in the interior of adjacent elements. $\square$

REMARK 2.50 Observe how the operator $\nabla\times$ acts on vector-valued monomials from the product space $\boldsymbol{Q}_B$:

$$(a\,\xi_1^i\xi_2^{j+1}\xi_3^{k+1}, b\,\xi_1^{i+1}\xi_2^j\xi_3^{k+1}, c\,\xi_1^{i+1}\xi_2^{j+1}\xi_3^k)$$

$$\begin{aligned}\xrightarrow{\nabla\times}\ \Big(&c(j+1)\xi_1^{i+1}\xi_2^j\xi_3^k - b(k+1)\xi_1^{i+1}\xi_2^j\xi_3^k,\\ &a(k+1)\xi_1^i\xi_2^{j+1}\xi_3^k - c(i+1)\xi_1^i\xi_2^{j+1}\xi_3^k,\\ &b(i+1)\xi_1^i\xi_2^j\xi_3^{k+1} - a(j+1)\xi_1^i\xi_2^j\xi_3^{k+1}\Big).\end{aligned}$$

$\square$

Remark 2.50 explains why the De Rham diagram requires that a finite element of the form $\mathcal{K}_B^{\mathrm{div}} = (K_B, \boldsymbol{V}_B, \Sigma_B^{\mathrm{div}})$ is equipped with polynomial space

$$\begin{aligned}\boldsymbol{V}_B = \{&\boldsymbol{v} \in \mathcal{Q}_{p^{b,1}+1,p^{b,2},p^{b,3}} \times \mathcal{Q}_{p^{b,1},p^{b,2}+1,p^{b,3}} \times \mathcal{Q}_{p^{b,1},p^{b,2},p^{b,3}+1};\\ &\boldsymbol{v}\cdot\boldsymbol{n}|_{s_i} \in \mathcal{Q}_{p^{s_i,1},p^{s_i,2}}(s_i),\ i = 1,\ldots,6\},\end{aligned} \tag{2.104}$$

which is a natural descendant of the vector-valued space $\boldsymbol{Q}_B$ of the form (2.62),

$$\begin{aligned}\boldsymbol{Q}_B = \{&\boldsymbol{E} \in \mathcal{Q}_{p^{b,1},p^{b,2}+1,p^{b,3}+1} \times \mathcal{Q}_{p^{b,1}+1,p^{b,2},p^{b,3}+1} \times \mathcal{Q}_{p^{b,1}+1,p^{b,2}+1,p^{b,3}};\\ &\boldsymbol{E}_t|_{s_i} \in \mathcal{Q}_{p^{s_i,1},p^{s_i,2}+1}(s_i) \times \mathcal{Q}_{p^{s_i,1}+1,p^{s_i,2}}(s_i);\\ &\boldsymbol{E}\cdot\boldsymbol{t}|_{e_j} \in \mathcal{P}_{p^{e_j}}(e_j),\ i = 1,\ldots,6,\ j = 1,\ldots,12\}.\end{aligned}$$

Recall the local coordinate systems for faces $s_1, \ldots, s_6$ defined in Paragraph 2.2.4.

The hierarchic basis of space $\boldsymbol{V}_B$ will consist of *face functions*, whose normal component vanishes in the standard sense on all faces but one, and *bubble functions*, whose normal component vanishes identically on all faces. The construction of both of these types of functions will be very much analogous to the $\boldsymbol{H}$(curl)-conforming case.

Face functions for the faces s_1 and s_2 (see Figure 2.26) are defined as

$$\begin{aligned} \gamma^{s_1}_{n_1,n_2,B} &= l_0(\xi_1)L_{n_1}(\xi_2)L_{n_2}(\xi_3)\boldsymbol{\xi}_1, \\ &\qquad 0 \le n_1 \le p^{s_1,1},\ 0 \le n_2 \le p^{s_1,2}; \\ \gamma^{s_2}_{n_1,n_2,B} &= l_1(\xi_1)L_{n_1}(\xi_2)L_{n_2}(\xi_3)\boldsymbol{\xi}_1, \\ &\qquad 0 \le n_1 \le p^{s_2,1},\ 0 \le n_2 \le p^{s_2,2}, \end{aligned} \tag{2.105}$$

leaving the rest to the reader as an easy exercise. Notice that with $n_1 = n_2 = 0$, the above definition encompasses *Whitney functions*, whose nonzero normal component is in the standard sense equal to one. Whitney functions again provide a complete basis for a lowest-order element. For future reference let us mention that all face functions are *oriented* according to the canonical vector involved in their specific formulae.

Bubble functions are considered in the form

$$\begin{aligned} \psi^{b,1}_{n_1,n_2,n_3,B} &= l_{n_1}(\xi_1)L_{n_2}(\xi_2)L_{n_3}(\xi_3)\boldsymbol{\xi}_1, \\ &\qquad 2 \le n_1 \le p^{b,1}+1,\ 0 \le n_2 \le p^{b,2},\ 0 \le n_3 \le p^{b,3}; \\ \psi^{b,2}_{n_1,n_2,n_3,B} &= L_{n_1}(\xi_1)l_{n_2}(\xi_2)L_{n_3}(\xi_3)\boldsymbol{\xi}_2, \\ &\qquad 0 \le n_1 \le p^{b,1},\ 2 \le n_2 \le p^{b,2}+1,\ 0 \le n_3 \le p^{b,3}; \\ \psi^{b,3}_{n_1,n_2,n_3,B} &= L_{n_1}(\xi_1)L_{n_2}(\xi_2)l_{n_3}(\xi_3)\boldsymbol{\xi}_3, \\ &\qquad 0 \le n_1 \le p^{b,1},\ 0 \le n_2 \le p^{b,2},\ 2 \le n_3 \le p^{b,3}+1. \end{aligned} \tag{2.106}$$

The number of bubble functions (2.106) is

$$p^{b,1}(p^{b,2}+1)(p^{b,3}+1)+(p^{b,1}+1)p^{b,2}(p^{b,3}+1)+(p^{b,1}+1)(p^{b,2}+1)p^{b,3}. \tag{2.107}$$

Numbers of vector-valued shape functions in the hierarchic basis of the space $\boldsymbol{V}_B$ are summarized in Table 2.13.

PROPOSITION 2.14

Vector-valued shape functions (2.105) and (2.106) represent a hierarchic basis of the space $\mathbf{V}_B$*, defined in (2.104).*

TABLE 2.13: Vector-valued hierarchic shape functions of $\mathcal{K}_B^{\mathrm{div}}$.

Node type	**Polynomial order**	**Number of shape functions**	**Number of nodes**
Face	always	$(p^{s_i,1}+1)(p^{s_i,2}+1)$	6
Interior	$1 \le p^{b,m}$ for some m	see (2.107)	1

PROOF Consider vector components one at a time. It is easy to see that all functions belong to the space $\boldsymbol{V}_B$, and that they are linearly independent. Counting them up, we obtain the dimension of the space $\boldsymbol{V}_B$,

$$\begin{aligned}\dim(\boldsymbol{V}_B) = \sum_{i=1}^{6}(p^{s_i,1}+1)(p^{s_i,2}+1) + p^{b,1}(p^{b,2}+1)(p^{b,3}+1)\\ +(p^{b,1}+1)p^{b,2}(p^{b,3}+1) + (p^{b,1}+1)(p^{b,2}+1)p^{b,3},\end{aligned}$$

which finishes the proof. $\square$

2.4.5 Tetrahedral master element $\mathcal{K}_T^{\mathrm{div}}$

Next we design a master element of arbitrary order $\mathcal{K}_T^{\mathrm{div}}$, associated with the reference tetrahedral domain K_T (Figure 2.30). This time we consider one local order of approximation p^b in the element interior only, and one local order p^{s_i} per face, $i = 1, \dots, 4$.

REMARK 2.51 The minimum rule for $\boldsymbol{H}(\mathrm{div})$-conforming approximations (Remark 2.43) in this case, locally on the reference domain, translates into

$$p^{s_i} \le p^b.$$

$\square$

It follows from the De Rham diagram (2.3) that a finite element of the form $\mathcal{K}_T^{\mathrm{div}} = (K_T, \boldsymbol{V}_T, \Sigma_T^{\mathrm{div}})$ should carry a vector-valued polynomial space

$$\boldsymbol{V}_T = \left\{\boldsymbol{v} \in (\mathcal{P}_{p^b})^3(K_T);\ \boldsymbol{v}\cdot\boldsymbol{n}|_{s_i} \in \mathcal{P}_{p^{s_i}}(s_i);\ i = 1, \dots, 4\right\}. \qquad (2.108)$$

This space is a natural descendant of the space $\boldsymbol{Q}_T$,

$$\begin{aligned}\boldsymbol{Q}_T = \{\boldsymbol{E} \in (\mathcal{P}_{p^b+1})^3(K_T);\ \boldsymbol{E}_t|_{s_i} \in (\mathcal{P}_{p^{s_i}+1})^2(s_i);\\ \boldsymbol{E}\cdot\boldsymbol{t}|_{e_j} \in \mathcal{P}_{p^{e_j}+1}(e_j);\ i = 1, \dots, 4,\ j = 1, \dots, 6,\}\end{aligned}$$

that was defined in (2.72).

This time, *face functions* will be split into *Whitney*, *linear*, *edge-based* and *genuine*. Before constructing them, recall the relation (2.73),

$$\boldsymbol{n}_{i,T} = \frac{\nabla \lambda_{i,T}}{|\nabla \lambda_{i,T}|}, \quad i = 1, \ldots, 4,$$

defining unitary normal vectors to faces $s_1, \ldots, s_4$. We will also exploit orientation of faces introduced in Paragraph 2.2.5 (recall that for each face we selected a vertex v_A with lowest local index, and by v_B, v_C denoted its two remaining vertices in increasing order. For each face s_i this choice determines three affine coordinates $\lambda_A, \lambda_B, \lambda_C$, such that $\lambda_A(v_A) = \lambda_B(v_B) = \lambda_C(v_C) = 1$).

Whitney face functions will be composed from elementary *vertex-based face functions*. Consider a face s_i, and by v_D denote the element-opposite vertex. Denote $e_A = v_A v_D$, $e_B = v_B v_D$, $e_C = v_C v_D$, and construct unitary tangential vectors

$$\boldsymbol{t}_A = \frac{v_A - v_D}{|v_A - v_D|}, \ \boldsymbol{t}_B = \frac{v_B - v_D}{|v_B - v_D|}, \ \boldsymbol{t}_C = \frac{v_C - v_D}{|v_C - v_D|}.$$

Notice that the normal component of functions

$$\frac{\lambda_A \boldsymbol{t}_A}{\boldsymbol{t}_A \cdot \boldsymbol{n}_{i,T}}, \ \frac{\lambda_B \boldsymbol{t}_B}{\boldsymbol{t}_B \cdot \boldsymbol{n}_{i,T}}, \ \frac{\lambda_C \boldsymbol{t}_C}{\boldsymbol{t}_C \cdot \boldsymbol{n}_{i,T}}, \qquad (2.109)$$

exactly coincides with λ_A, λ_B and λ_C on the face s_i, respectively, and vanishes on all other faces, as illustrated in Figure 2.47.

Now it is easy to define a Whitney function,

$$\gamma_{0,T}^{s_i} = \frac{\lambda_A \boldsymbol{t}_A}{\boldsymbol{t}_A \cdot \boldsymbol{n}_{i,T}} + \frac{\lambda_B \boldsymbol{t}_B}{\boldsymbol{t}_B \cdot \boldsymbol{n}_{i,T}} + \frac{\lambda_C \boldsymbol{t}_C}{\boldsymbol{t}_C \cdot \boldsymbol{n}_{i,T}}, \qquad (2.110)$$

whose normal component is equal to one on the face s_i and vanishes on all other faces. Four Whitney functions, associated with faces $s_1, \ldots, s_4$, form a complete lowest-order element. Elementary functions (2.109) can be further combined to produce *linear face functions*,

$$\begin{aligned} \gamma_{1,T}^{s_i,1} &= \frac{\lambda_B \boldsymbol{t}_B}{\boldsymbol{t}_B \cdot \boldsymbol{n}_{i,T}} - \frac{\lambda_A \boldsymbol{t}_A}{\boldsymbol{t}_A \cdot \boldsymbol{n}_{i,T}}, \\ \gamma_{1,T}^{s_i,2} &= \frac{\lambda_A \boldsymbol{t}_A}{\boldsymbol{t}_A \cdot \boldsymbol{n}_{i,T}} - \frac{\lambda_C \boldsymbol{t}_C}{\boldsymbol{t}_C \cdot \boldsymbol{n}_{i,T}}. \end{aligned} \qquad (2.111)$$

Together with Whitney functions, these functions form a complete first-order element.

Next we construct *edge-based face functions*. Consider an oriented edge $e_j = v_E v_F$, lying in the face s_i, and affine coordinates λ_E, λ_F, such that $\lambda_E(v_E) = \lambda_F(v_F) = 1$. The product $\lambda_E \lambda_F$ vanishes everywhere on the element surface,

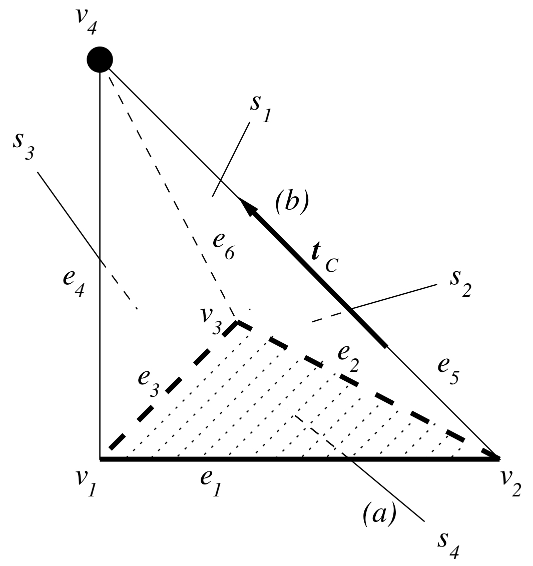

FIGURE 2.47: Consider the face $s_i = s_3$, and its vertex $v_C = v_4$ For this face it is $v_D = v_2$. The elementary vertex-based face function $\lambda_C \boldsymbol{t}_C/(\boldsymbol{t}_C \cdot \boldsymbol{n}_{i,T})$ ***(a)*** vanishes completely on the face s_4 ($\lambda_C \equiv 0$ on s_4), ***(b)*** its normal component vanishes also on each face s_1, s_2 ($\boldsymbol{n}_{1,T} \cdot \boldsymbol{t}_C = \boldsymbol{n}_{2,T} \cdot \boldsymbol{t}_C = 0$). Thus, the normal component is nonzero only on the face s_3, where it coincides with $\lambda_{4,T}$.

except for two faces s_i, s_D. We eliminate the normal component from the face s_D in the standard way, exploiting a suitable tangential direction. This task can be done by, for example, a unitary tangential vector to its edge e_D, $e_D \neq e_j$, which follows after e_j in the local orientation associated with the face s_D. Hence we obtain a set of linearly independent edge-based face functions

$$\gamma_{k,T}^{s_i,e_j} = \lambda_E \lambda_F L_{k-2}(\lambda_F - \lambda_E) \frac{\boldsymbol{t}_D}{\boldsymbol{t}_D \cdot \boldsymbol{n}_{i,T}}, \quad 2 \leq k \leq p^{s_i}. \tag{2.112}$$

The real coefficient $1/(\boldsymbol{t}_D \cdot \boldsymbol{n}_{i,T})$ is introduced for future compatibility with prismatic elements.

$\boldsymbol{H}$(curl)-conforming face-based bubble functions (2.80), whose construction was shown in Figure 2.44, will play the role of *genuine face functions* in the basis of $\boldsymbol{V}_T$:

$$\gamma_{n_1,n_2,T}^{s_i} = \psi_{n_1,n_2,T}^{b,s_i}, \quad 1 \leq n_1, n_2; \; n_1 + n_2 \leq p^{s_i} - 1. \tag{2.113}$$

What remains to be done now is to design *bubble functions* whose normal component vanishes on all faces. As in [6], we will split them into three groups – *edge-based*, *face-based* and *genuine*.

Consider an oriented edge $e_j = v_E v_F$, and affine coordinates λ_E, λ_F, such that $\lambda_E(v_E) = \lambda_F(v_F) = 1$. *Edge-based bubble functions*, associated with the edge e_j, have the form

$$\gamma^{b,e_j}_{n_1,T} = \lambda_E \lambda_F L_{n_1-2}(\lambda_F - \lambda_E)\boldsymbol{t}_{j,T}, \quad 2 \le n_1 \le p^b. \tag{2.114}$$

The construction is illustrated in Figure 2.48.

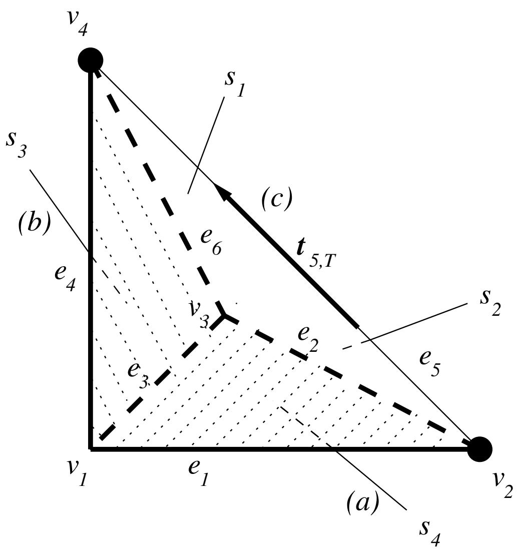

FIGURE 2.48: Consider the edge $e_j = e_5$. Edge-based bubble functions $\gamma^{b,e_5}_{k,T}$ ***(a)***, ***(b)*** vanish completely on faces s_3, s_4 ($\lambda_E\lambda_F \equiv 0$), and ***(c)*** their normal component vanishes also on each face s_1, s_2 ($\boldsymbol{t}_{5,T} \cdot \boldsymbol{n}_{1,T} = \boldsymbol{t}_{5,T} \cdot \boldsymbol{n}_{2,T} = 0$).

$\boldsymbol{H}$(curl)-conforming *genuine face functions* (2.79), whose construction was illustrated in Figure 2.43, can be used as *face-based bubble functions*,

$$\begin{aligned} \gamma^{b,s_i,1}_{n_1,n_2,T} &= \psi^{s_i,1}_{n_1,n_2,T}, \\ \gamma^{b,s_i,2}_{n_1,n_2,T} &= \psi^{s_i,2}_{n_1,n_2,T}, \end{aligned} \tag{2.115}$$

$1 \le n_1, n_2;\ n_1 + n_2 \le p^b - 1$.

Also *genuine bubble functions* can be chosen in the same way as in the $\boldsymbol{H}$(curl)-conforming case,

$$\gamma^{b,m}_{n_1,n_2,n_3,T} = \psi^{b,m}_{n_1,n_2,n_3,T}, \tag{2.116}$$

$1 \le n_1, n_2, n_3;\ n_1 + n_2 + n_3 \le p^b - 1;\ m = 1, \ldots, 3.$

Numbers of vector-valued shape functions in the hierarchic basis of the space $\boldsymbol{V}_T$ are summarized in Table 2.14.

TABLE 2.14: Vector-valued hierarchic shape functions of $\mathcal{K}_T^{\mathrm{div}}$.

Node type	Polyn. order	Number of shape functions	Num. of nodes
Whitney	always	1	4
Linear face	$1 \le p^{s_i}$	2	4
Edge-based face	$2 \le p^{s_i}$	$3(p^{s_i} - 1)$	4
Genuine face	$3 \le p^{s_i}$	$(p^{s_i} - 2)(p^{s_i} - 1)/2$	4
Edge-based bubble	$2 \le p^b$	$6(p^b - 1)$	1
Face-based bubble	$3 \le p^b$	$4(p^b - 2)(p^b - 1)$	1
Genuine bubble	$4 \le p^b$	$d(p^b - 3)(p^b - 2)(p^b - 1)/6$	1

PROPOSITION 2.15

Whitney face functions (2.110), linear face functions (2.111), edge-based face functions (2.112), genuine face functions (2.113), and bubble functions (2.114), (2.115) and (2.116), provide a complete basis of the space $\boldsymbol{V}_T$, defined in (2.108).

PROOF In the standard way: first verify that all shape functions belong to the space $\boldsymbol{V}_T$. It is easy to see that they are linearly independent, and that they generate the whole space $\boldsymbol{V}_T$. $\square$

2.4.6 Prismatic master element $\mathcal{K}_P^{\mathrm{div}}$

Our last master element of arbitrary order, $\mathcal{K}_P^{\mathrm{div}}$, will be associated with the reference prismatic domain K_P (Figure 2.34).

Consider local directional orders of approximation $p^{b,1}, p^{b,2}$ in the element interior. The order $p^{b,1}$ corresponds to the plane $\xi_1\xi_2$ (again, we will designate this the *horizontal direction*), and $p^{b,2}$ to the *vertical direction* ξ_3. Quadrilateral faces s_i, $i = 1, \ldots, 3$, are assigned local directional orders of approximation $p^{s_i,1}, p^{s_i,2}$ (in horizontal and vertical direction, respectively). Triangular faces s_4, s_5 come with one local order of approximation p^{s_i} only, $i = 4, 5$. Edges are not constrained by $\boldsymbol{H}(\mathrm{div})$-conformity requirements. The minimum rule for $\boldsymbol{H}(\mathrm{div})$-conforming approximations (Remark 2.43) applies.

The De Rham diagram (2.3) suggests that a finite element of the form $\mathcal{K}_P^{\mathrm{div}} = (K_P, \boldsymbol{V}_P, \Sigma_P^{\mathrm{div}})$ should be equipped with a vector-valued polynomial space

$$\begin{aligned}\boldsymbol{V}_P = \{&\boldsymbol{v} \in \mathcal{R}_{p^{b,1},p^{b,2}}(K_P) \times \mathcal{R}_{p^{b,1},p^{b,2}}(K_P) \times \mathcal{R}_{p^{b,1}-1,p^{b,2}+1}(K_P);\\ &\boldsymbol{v}\cdot\boldsymbol{n}|_{s_i} \in \mathcal{Q}_{p^{s_i,1},p^{s_i,2}}(s_i) \text{ for } i = 1,\dots,3;\\ &\boldsymbol{v}\cdot\boldsymbol{n}|_{s_i} \in \mathcal{P}_{p^{s_i}}(s_i) \ \text{ for } i = 4,5\}\,,\end{aligned} \tag{2.117}$$

which is defined only if $p^{b,1} \geq 1$. The appropriate ancestor space has the form (2.83),

$$\begin{aligned}\boldsymbol{Q}_P = \{&\boldsymbol{E} \in \mathcal{R}_{p^{b,1},p^{b,2}+1}(K_P) \times \mathcal{R}_{p^{b,1},p^{b,2}+1}(K_P) \times \mathcal{R}_{p^{b,1}+1,p^{b,2}}(K_P);\\ &\boldsymbol{E}_t|_{s_i} \in \mathcal{Q}_{p^{s_i,1},p^{s_i,2}+1}(s_i) \times \mathcal{Q}_{p^{s_i,1}+1,p^{s_i,2}}(s_i) \text{ for } i = 1,\dots,3;\\ &\boldsymbol{E}_t|_{s_i} \in (\mathcal{P}_{p^{s_i}})^2(s_i) \text{ for } i = 4,5;\\ &\boldsymbol{E}\cdot\boldsymbol{t}|_{e_j} \in \mathcal{P}_{p^{e_j}}(e_j);\ \ j = 1,\dots,9\}\,.\end{aligned}$$

The design of the finite element $\mathcal{K}_P^{\mathrm{div}}$ will be accomplished by defining a suitable hierarchic basis of the space $\boldsymbol{V}_P$.

REMARK 2.52 Similarly as in the $\boldsymbol{H}$(curl)-conforming case, we will exploit the product geometry $K_P = K_t \times K_a$ to simplify the construction. The De Rham diagram indicates that the first two vector components of the shape functions should be constructed as products of shape functions associated with the master element $\mathcal{K}_t^{\mathrm{div}}$ in ξ_1, ξ_2 (again formally extended to 3D by adding zero third component), and Legendre polynomials in ξ_3, while the third vector component should have the form of a product of scalar shape functions of the master element $\mathcal{K}_t^1$ in ξ_1, ξ_2, and Legendre polynomials in ξ_3. The construction of shape functions for the master elements $\mathcal{K}_t^1$ and $\mathcal{K}_t^{\mathrm{div}}$ was described in Paragraphs 2.2.3 and 2.4.3. $\square$

The basis of the space $\boldsymbol{V}_P$ will comprise *face functions* whose normal component vanishes in the standard sense on all faces but one, and *bubble functions* whose normal component vanishes on all faces. Face functions for quadrilateral and triangular faces will be constructed to be compatible with face functions of the master brick $\mathcal{K}_B^{\mathrm{div}}$ and master tetrahedron $\mathcal{K}_T^{\mathrm{div}}$, respectively.

Simplifying the notation as explained in Remark 2.52, face functions for quadrilateral faces s_i, $i = 1,\dots,3$, can be written as

$$\gamma^{s_i}_{n_1,n_2,P}(\xi_1,\xi_2,\xi_3) = \gamma^{e_i}_{n_1,t}(\xi_1,\xi_2)L_{n_2}(\xi_3), \quad 0 \leq n_1 \leq p^{s_i,1},\ 0 \leq n_2 \leq p^{s_i,2}, \tag{2.118}$$

where two-dimensional edge functions $\gamma^{e_i}_{n_1,t}$ have been defined in (2.99), (2.100) and (2.101). The compatibility of face functions (2.118) with face functions (2.105) of the master brick $\mathcal{K}_B^{\mathrm{div}}$ is an immediate effect of the fact that nonzero normal components of 2D edge functions $\gamma_{n_1,t}^{e_i}$ are the Legendre polynomials $L_0, L_1, \ldots$.

Face functions for triangular faces s_4 and s_5 will reside only in the third vector component. Their construction is very similar as their local orientations are the same as that of the master triangle $\mathcal{K}_t^{\mathrm{div}}$. Let us consider the face s_5 first. To be compatible with the master tetrahedron $\mathcal{K}_T^{\mathrm{div}}$, first we need a Whitney triangular face function, whose normal component on the face s_5 would be equal to one. Such a function will always be present in the basis of $\boldsymbol{V}_P$, and we can define it as

$$\gamma_{0,P}^{s_5}(\xi_1,\xi_2,\xi_3) = \underbrace{\sum_{k=1}^{3} \varphi_t^{v_k}(\xi_1,\xi_2)}_{\equiv 1} l_1(\xi_3)\boldsymbol{n}_{5,T}, \tag{2.119}$$

where $\varphi_t^{v_k}$, $k = 1,\ldots,3$, are scalar vertex functions of the master triangle $\mathcal{K}_t^1$. Similarly we define for the face s_5 *linear triangular face functions*

$$\begin{aligned} \gamma_{1,P}^{s_5,1}(\xi_1,\xi_2,\xi_3) &= (\varphi_t^{v_2} - \varphi_t^{v_1})(\xi_1,\xi_2)l_1(\xi_3)\boldsymbol{n}_{5,P}, \\ \gamma_{1,P}^{s_5,2}(\xi_1,\xi_2,\xi_3) &= (\varphi_t^{v_1} - \varphi_t^{v_3})(\xi_1,\xi_2)l_1(\xi_3)\boldsymbol{n}_{5,P}, \end{aligned} \tag{2.120}$$

which are present in the basis of $\boldsymbol{V}_P$ if $p^{s_5} \geq 1$. They are compatible with linear face functions (2.111) of the master tetrahedron $\mathcal{K}_T^{\mathrm{div}}$. *Edge-based triangular face functions* related to the face s_5 can be written as

$$\gamma_{k,T}^{s_5,e_j} = \varphi_{k,t}^{e_j}(\xi_1,\xi_2)l_1(\xi_3)\boldsymbol{n}_{5,P}, \quad 2 \leq k \leq p^{s_5};\ j = 7,8,9. \tag{2.121}$$

Notice, again, that their normal components have the same form as those of the edge-based face functions (2.112) of the master tetrahedron $\mathcal{K}_T^{\mathrm{div}}$. The last group of face functions for the face s_5 are *genuine triangular face functions*,

$$\gamma_{n_1,n_2,T}^{s_5} = \varphi_{n_1,n_2,t}^{b,1}(\xi_1,\xi_2)l_1(\xi_3)\boldsymbol{n}_{5,P}, \tag{2.122}$$

$1 \leq n_1, n_2;\ n_1 + n_2 \leq p^{s_5} - 1$, whose nonzero normal components match those of genuine face functions (2.113) of the master tetrahedron $\mathcal{K}_T^{\mathrm{div}}$.

As for face s_4, we only use $l_0(\xi_3)$ instead of $l_1(\xi_3)$ to let the functions vanish on the opposite triangular face s_5 and exchange $\boldsymbol{n}_{5,P}$ for $\boldsymbol{n}_{4,P}$.

At this point we need only bubble functions to complete the basis of the space $\boldsymbol{V}_P$. Let us begin with functions that are only nonzero in their first two vector components. We have *horizontal bubble functions* of the form

$$\gamma_{n_1,n_2,n_3,P}^{b,1} = \gamma_t^b(\xi_1,\xi_2)L_{n_3}(\xi_3), \quad 0 \leq n_3 \leq p^{b,2}, \tag{2.123}$$

where γ_t^b stands for edge-based and genuine bubble functions (2.102) and (2.103) of the master triangle $\mathcal{K}_t^{\mathrm{div}}$ up to the order $p^{b,1}$ (recall that in this case $p^{b,1} \geq 1$). Their number is

$$\left(3(p^{b,1}-1)+(p^{b,1}-1)(p^{b,1}-2)\right)(p^{b,2}+1). \tag{2.124}$$

Vertical bubble functions whose third vector component is the only one that is nonzero are defined as

$$\gamma_{n_1,n_2,n_3,P}^{b,2} = \varphi_t(\xi_1,\xi_2) l_0(\xi_3) l_1(\xi_3) L_{n_3-2}(\xi_3)\boldsymbol{\xi}_3, \quad 2 \leq n_3 \leq p^{b,2}+1, \tag{2.125}$$

where φ_t stands for *all scalar shape functions* up to the order $p^{b,1}-1$, associated with the master triangle $\mathcal{K}_t^1$. Scalar shape functions are here understood in the above sense, i.e., involving one constant lowest-order function $\varphi_t^{v_1} + \varphi_t^{v_2} + \varphi_t^{v_3}$, two linear functions and standard higher-order edge and bubble functions.

Numbers of vector-valued shape functions in the hierarchic basis of the space $\boldsymbol{V}_P$ are summarized in Table 2.15.

TABLE 2.15: Vector-valued hierarchic shape functions of $\mathcal{K}_P^{\mathrm{div}}$.

Node type	**Polynomial order**	**Number of shape functions**	**Number of nodes**
Quad. face ($i = 1,2,3$)	always	$(p^{s_i,1}+1)(p^{s_i,2}+1)$	3
Tri. face ($i = 4,5$)	always	$(p^{s_i}+1)(p^{s_i}+2)/2$	2
Horizontal bubble	$2 \leq p^{b,1}$	see (2.124)	1
Vertical bubble	$1 \leq p^{b,2}$	$p^{b,1}(p^{b,1}+1)p^{b,2}/2$	1

PROPOSITION 2.16
Quadrilateral face functions (2.118), triangular face functions (2.119), (2.120), (2.121) and (2.122), and bubble functions (2.123) and (2.125) constitute a hierarchic basis of the space $\boldsymbol{V}_P$, defined in (2.117).

PROOF It is easy to see that all shape functions belong to the space $\boldsymbol{V}_P$, and their product structure easily reveals their linear independence. Finally we have to verify that their number is equal to the dimension of the space $\boldsymbol{V}_P$. In this case the computation is easy, looking separately at basis functions with a zero third vector component (2.119), (2.120), (2.121), (2.122) and (2.125), and at basis functions with zero first two vector components (2.118) and (2.123).
$\square$

2.5 L^2-conforming approximations

In the last section of this chapter we will briefly discuss the design of L^2-conforming finite elements of arbitrary order on the reference domains K_q, K_t, K_B, K_T and K_P.

2.5.1 De Rham diagram and finite elements in L^2

The space L^2 stands at the end of the De Rham diagram (2.1), (2.2) and (2.3), and therefore the hierarchy of shape functions for L^2-conforming approximations is simpler than in the spaces H^1, $\boldsymbol{H}(\mathrm{curl})$ and $\boldsymbol{H}(\mathrm{div})$. Since no conformity restrictions are imposed on vertices, edges and faces, all shape functions are *bubble functions.* Obviously there are *no minimum rules* for L^2-conforming approximations. The design of master elements is very simple this time.

2.5.2 Master elements for L^2-conforming approximations

The L^2-conforming case is not exceptional in the sense that a master finite element will be constructed as a triad $\mathcal{K}^{L^2} = (K, X, \Sigma^{L^2})$, where K stands for a reference domain, X is a finite dimensional space, and Σ^{L^2} represents a set of degrees of freedom, which will be uniquely identified by a choice of basis of the space X.

Quadrilateral master element $\mathcal{K}_q^{L^2}$

Consider using local directional polynomial orders of approximation $p^{b,1}, p^{b,2}$ in the element interior to allow for anisotropic p-refinement. The basis of the space

$$X_q = \mathcal{Q}_{p^{b,1},p^{b,2}}, \tag{2.126}$$

where $\mathcal{Q}_{p,q}$ was introduced in (2.12), consists of $(p^{b,1}+1)(p^{b,2}+1)$ bubble functions

$$\omega^b_{n_1,n_2,q} = L_{n_1}(\xi_1)L_{n_2}(\xi_2), \quad 0 \le n_1 \le p^{b,1},\ 0 \le n_2 \le p^{b,2}. \tag{2.127}$$

PROPOSITION 2.17

Shape functions (2.127) form a basis of the space X_q, defined in (2.126).

Triangular master element $\mathcal{K}_t^{L^2}$

Consider local polynomial order of approximation p^b in the element interior. The basis of the space

$$X_t = \mathcal{P}_{p^b}(K_t), \tag{2.128}$$

where $\mathcal{P}_p(K_t)$ was defined in (2.19), consists of $(p^b+1)(p^b+2)/2$ bubble functions

$$\omega^b_{n_1,n_2,t} = L_{n_1}(\lambda_{3,t}-\lambda_{2,t})L_{n_2}(\lambda_{2,t}-\lambda_{1,t}), \quad 0 \le n_1, n_2;\ n_1+n_2 \le p^b. \tag{2.129}$$

PROPOSITION 2.18
Shape functions (2.129) form a basis of the space X_t, defined in (2.128).

Brick master element $\mathcal{K}_B^{L^2}$

Consider using local directional polynomial orders of approximation $p^{b,1}, p^{b,2}$, $p^{b,3}$ in the element interior to allow for anisotropic p-refinement. The basis of the space

$$X_B = \mathcal{Q}_{p^{b,1},p^{b,2},p^{b,3}}, \tag{2.130}$$

consists of $(p^{b,1}+1)(p^{b,2}+1)(p^{b,3}+1)$ bubble functions

$$\omega^b_{n_1,n_2,n_3,B} = L_{n_1}(\xi_1)L_{n_2}(\xi_2)L_{n_3}(\xi_3), \tag{2.131}$$

$0 \le n_1 \le p^{b,1},\ 0 \le n_2 \le p^{b,2},\ 0 \le n_3 \le p^{b,3}$.

PROPOSITION 2.19
Shape functions (2.131) form a basis of the space X_B, defined in (2.130).

Tetrahedral master element $\mathcal{K}_T^{L^2}$

Consider local polynomial order of approximation p^b in the element interior. The basis of the space

$$X_T = \mathcal{P}_{p^b}(K_T), \tag{2.132}$$

where

$$\mathcal{P}_p(K_T) = \mathrm{span}\left\{\xi_1^i\xi_2^j\xi_3^k;\ i,j,k = 0,\ldots,p;\ i+j+k \le p\right\},$$

consists of $(p^b+1)(p^b+2)(p^b+3)/6$ bubble functions

$$\omega^b_{n_1,n_2,n_3,T} = L_{n_1}(\lambda_{3,T}-\lambda_{2,T})L_{n_2}(\lambda_{2,T}-\lambda_{1,T})L_{n_3}(\lambda_{4,T}-\lambda_{2,T}), \tag{2.133}$$

$0 \le n_1, n_2, n_3;\ n_1 + n_2 + n_3 \le p^b$.

PROPOSITION 2.20
Shape functions (2.133) form a basis of the space X_T, defined in (2.132).

Prismatic master element $\mathcal{K}_P^{L^2}$

Consider local directional polynomial orders of approximation $p^{b,1}, p^{b,2}$ in the element interior ($p^{b,1}$ in *horizontal direction* $\xi_1\xi_2$, and p^{b_2} in the *vertical direction* ξ_3 as before). Anisotropic p-refinement of this element is allowed only in the vertical direction. The basis of the space

$$X_P = \mathcal{R}_{p^{b,1},p^{b,2}}(K_P), \tag{2.134}$$

(meaning of this symbol is the same as before) consists of $(p^{b,1}+1)(p^{b,1}+2)(p^{b,2}+1)/2$ bubble functions

$$\omega^b_{n_1,n_2,n_3,P} = L_{n_1}(\lambda_{3,t} - \lambda_{2,t})L_{n_2}(\lambda_{2,t} - \lambda_{1,t})L_{n_3}(\xi_3), \tag{2.135}$$

$0 \le n_1, n_2;\ n_1 + n_2 \le p^{b,1};\ 0 \le n_3 \le p^{b,2}$.

PROPOSITION 2.21
Shape functions (2.135) form a basis of the space X_P, defined in (2.134).

As a simple exercise, the reader may try to find ancestors of the aforementioned finite element spaces in the De Rham diagram.

Chapter 3

Higher-order finite element discretization

With a database of scalar and vector-valued hierarchic master elements of variable order on all reference domains in hand, we can proceed to the discussion of the higher-order finite element technology in two and three spatial dimensions. The reader should be prepared that although the methodology stays basically the same as in 1D, many of its particular aspects become more technical. Therefore we encourage her/him to be truly familiar with the one-dimensional model example from Section 1.3 before reading this chapter.

We will begin with the *projection-based interpolation* technique that extends the standard nodal interpolation to hierarchic higher-order elements. Section 3.3 is devoted to the construction of reference maps for all reference domains. In Section 3.5 we will design hierarchic higher-order elements in the physical domain by means of master element shape functions and the reference maps (or, in other words, we transfer the variational formulation from physical mesh elements to the reference domain). Presentation of the assembling algorithm accomplishes the discretization on regular meshes. An approach to the treatment of constrained approximation (discretization on 1-irregular meshes) will be presented in Section 3.6, and selected issues related to computer implementation of hierarchic higher-order finite element methods and automatic *hp*-adaptivity will be discussed in Section 3.7.

3.1 Projection-based interpolation on reference domains

Projection-based interpolation on hierarchic elements is a nontrivial technique that forms an essential part of higher-order finite element methods. Recall from Section 1.1 that in contrast to nodal higher-order elements, the degrees of freedom $L_1, L_2, \ldots, L_{N_P}$ for hierarchic elements *are not defined outside of the local polynomial space* $P(K)$. This means that Definition 1.7 cannot be used to design local interpolation operators for hierarchic elements. Hence one needs to combine the standard nodal (Lagrange) *interpolation* with *projection* on higher-order polynomial subspaces.

Given a sufficiently regular function $u \in V(\Omega_h)$, it is our aim to find an appropriate piecewise-polynomial interpolant $u_{h,p} \in V_{h,p}(\Omega_h)$, $V_{h,p} \subset V$. For every element $K \in \mathcal{T}_{h,p}$ with affine reference map $\boldsymbol{x}_K : \hat{K} \to K$ this is equivalent to the interpolation of the function $u|_K \circ \boldsymbol{x}_K$ *on the reference domain*. Therefore we will stay on the reference domain for a while. Extension to physical mesh elements, which in the case of nonaffine maps consists of an additional adjustment of metric on the reference domain, will be discussed in Section 3.4.

Properties of projection-based interpolation operators

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

1. *Locality.* The projection-based interpolant Πu of a function u is constructed elementwise. Therefore we request that *within an element,* Π *uses function values of u from this element only.*

2. *Global conformity.* For every function $u \in V^0$, $\overline{V^0} = V$, the projection-based interpolant $u_{h,p} = \Pi u$ still lies in the space V. Recall global conformity requirements from Paragraph 1.1.4:

   (a) *continuity* across element interfaces for $V = H^1$,

   (b) continuity of *tangential component* across element interfaces for $V = \boldsymbol{H}(\mathrm{curl})$, and

   (c) continuity of *normal component* across element interfaces for $V = \boldsymbol{H}(\mathrm{div})$.

3. *Optimality.* The interpolant $u_{h,p} \in V_{h,p}$ must have the minimum distance from the interpolated function $u \in V$ in an appropriate norm. The choice of a suitable norm in the spaces H^1, $\boldsymbol{H}(\mathrm{curl})$ and $\boldsymbol{H}(\mathrm{div})$ is a nontrivial mathematical question that also will be addressed.

3.1.1 H^1-conforming elements

Consider the one-dimensional master element $\mathcal{K}_a^1$ of a polynomial order $p^b \geq 1$, i.e., equipped with a polynomial space of the form (2.5),

$$W_a = \{w;\ w \in \mathcal{P}_{p^b}(K_a)\}.$$

Consider further a function $u \in H^1(K_a)$. Recall that hierarchic shape functions in one spatial dimension comprise vertex functions $\varphi_a^{v_1}$, $\varphi_a^{v_2}$ defined in (2.6), and bubble functions $\varphi_{k,a}^b$, $k = 2, 3, \ldots, p^b$, defined in (2.8). The projection-based interpolant $u_{h,p} = \Pi_a^1 u \in W_a$ is constructed as a sum of a vertex and bubble interpolants,

$$u_{h,p} = u^v_{h,p} + u^b_{h,p}.$$

Vertex interpolant $u^v_{h,p} \in W_a$ is a linear function that matches u at both endpoints,

$$u^v_{h,p}(\pm 1) = u(\pm 1).$$

Thus it can be expressed as a linear combination of vertex functions $\varphi^{v_1}_a$, $\varphi^{v_2}_a$, as illustrated in Figure 3.1.

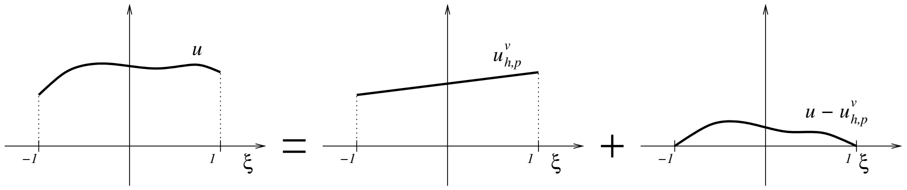

FIGURE 3.1: Decomposition of u into a vertex interpolant $u^v_{h,p}$ and a residual $u - u^v_{h,p}$ which vanishes at the endpoints.

Bubble interpolant $u^b_{h,p} \in W_a$ is obtained by projecting the residual $u - u^v_{h,p}$ on the space $\mathcal{P}_{p^b,0}(K_a)$ (of polynomials of the order at most p^b that vanish at interval endpoints) in the H^1-seminorm,

$$|u - u^v_{h,p} - u^b_{h,p}|_{H^1} \to \min . \tag{3.1}$$

Since the space $\mathcal{P}_{p^b,0}(K_a)$ is generated by the bubble functions $\varphi^b_{k,a}$, $k = 2, 3, \ldots, p^b$, the bubble interpolant $u^b_{h,p}$ can be expressed as

$$u^b_{h,p} = \sum_{m=2}^{p^b} \alpha^b_m \varphi^b_{m,a}.$$

Hence the discrete minimization problem (3.1) is equivalent to a system of $p^b - 1$ linear algebraic equations

$$\int_{K_a} (u - u^v_{h,p} - u^b_{h,p})'(\varphi^b_{k,a})' = 0, \quad k = 2, 3, \ldots, p^b,$$

for the unknown coefficients α^b_m.

REMARK 3.1 (Uniqueness of the interpolant) The vertex interpolant $u^v_{h,p}$ does not have to be linear. As long as it is chosen to be a polynomial of order lower than or equal to p^b, the function $u^b_{h,p}$, and consequently the final interpolant $u_{h,p}$, are uniquely defined. $\square$

Master triangle $\mathcal{K}^1_t$

Let the master element $\mathcal{K}^1_t$ be equipped with local polynomial orders p^{e_j}, $j = 1, 2, 3$ on its edges and with an order p^b in the interior. The minimum rule for H^1-conforming discretizations requires that $p^{e_j} \le p^b$ for all $j = 1, \ldots, 3$. The master element polynomial space has the form

$$W_t = \{w;\ w \in \mathcal{P}_{p^b}(K_t);\ w|_{e_j} \in \mathcal{P}_{p^{e_j}}(e_j),\ j = 1, 2, 3\}.$$

Consider a sufficiently regular function $u : K_t \to \mathbb{R}$ (theory requires that $u \in H^{1+\epsilon}(K_t)$, $\epsilon > 0$, which is usually satisfied in practical computations). Recall that hierarchic shape functions on the master triangle $\mathcal{K}^1_t$ comprise vertex functions $\varphi^{v_1}_t, \ldots, \varphi^{v_3}_t$ defined in (2.20), edge functions $\varphi^{e_1}_t, \ldots, \varphi^{e_3}_t$ from (2.21), and bubble functions $\varphi^b_{n_1,n_2,t}$, $1 \le n_1, n_2$, $n_1 + n_2 \le p^b - 1$, defined in (2.23).

The projection-based interpolant $u_{h,p} = \Pi^1_t u \in W_t$ is constructed as a sum of a vertex, edge and bubble interpolants,

$$u_{h,p} = u^v_{h,p} + u^e_{h,p} + u^b_{h,p}. \tag{3.2}$$

Vertex interpolant $u^v_{h,p} \in W_t$ is a linear function matching u at vertices,

$$u^v_{h,p}(v_j) = u(v_j),\ j = 1, 2, 3.$$

Similarly as in the one-dimensional case, one can write $u^v_{h,p}$ as a linear combination of vertex shape functions – see Figure 3.2.

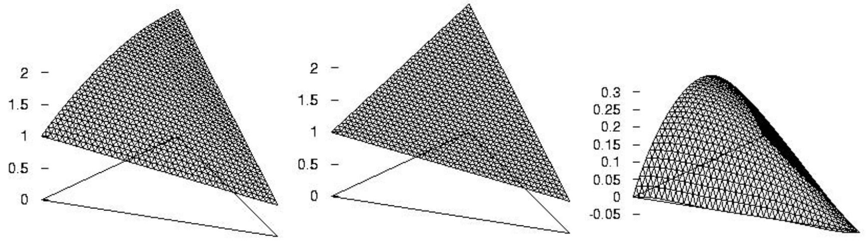

FIGURE 3.2: Decomposition of u (left) into a vertex interpolant $u^v_{h,p}$ (middle) and residual $u - u^v_{h,p}$ (right), which vanishes at the vertices. (Notice different scaling.)

Edge interpolant $u^e_{h,p} \in W_t$ is constructed as a sum of contributions from the element edges,

$$u^e_{h,p} = \sum_{j=1}^{3} u^{e_j}_{h,p}.$$

By the locality and conformity arguments, the value of $u^e_{h,p}$ along an edge e_j must depend on the values of the function u on the edge only – the only information shared by the corresponding pair of neighboring elements. The approximation theory suggests that the distance between the residual $u - u^v_{h,p}$ and the edge interpolant $u^e_{h,p}$ is minimized on edges e_j, $j = 1, 2, 3$, in the norm $H^{\frac{1}{2}}_{00}(e_j)$.

This is a nontrivial norm. To understand it, choose one of the element edges e_j and consider the space

$$H^{\frac{1}{2}}_{00}(e_j) = \left\{ \tilde{w}|_{e_j};\ \tilde{w} \in H^{\frac{1}{2}}(\partial K_t),\ \tilde{w} \equiv 0 \text{ on } \partial K_t \setminus e_j \right\},$$

where $H^{\frac{1}{2}}(\partial K_t)$ is the space of traces of functions from $H^1(K_t)$ to the boundary ∂K_t. For a function $\tilde{w}$ from this space define

$$\|\tilde{w}\|_{H^{\frac{1}{2}}_{00}(e_j)} = \|w\|_{H^1(K_t)} = \|\nabla w\|_{L^2(K_t)},$$

where $w \in H^1(K_t)$ is the *minimum energy extension* of $\tilde{w}$ into the element interior, i.e., $\triangle w = 0$ in K_t, $w \equiv \tilde{w}$ on e_j and $w \equiv 0$ on remaining edges. Indeed it is difficult to evaluate this norm exactly, and in practice one approximates it with a weighted H^1_0 norm [64],

$$\|\tilde{w}\|^2_{H^{\frac{1}{2}}_{00}(e_j)} \approx \|\tilde{w}\|^2_{e_j} = \int_{e_j} \left(\frac{\mathrm{d}\tilde{w}}{\mathrm{d}s}\right)^2 \left(\frac{\mathrm{d}s}{\mathrm{d}\zeta}\right)^{-1} \mathrm{d}s = \int_{-1}^{1} \left(\frac{\mathrm{d}\tilde{w}}{\mathrm{d}\zeta}\right)^2 \mathrm{d}\zeta, \tag{3.3}$$

where $\boldsymbol{x} = \boldsymbol{x}(\zeta)$, $\zeta \in (-1, 1)$ is the parametrization of the edge e_j, and

$$\frac{\mathrm{d}s}{\mathrm{d}\zeta} = \sqrt{\sum_{i=1}^{d} \left(\frac{\mathrm{d}x_i}{\mathrm{d}\zeta}\right)^2}.$$

Notice that this weighted H^1_0 norm scales with the length of the edge e_j in the same way as the $H^{\frac{1}{2}}_{00}$ norm. The difference between these two norms was studied in [64] with the conclusion that it is, at least for orders $p < 10$, insignificant. In the following we will still refer to the norm as to $H^{\frac{1}{2}}_{00}$ despite its appropriate nature upon implementation.

We proceed one edge at a time, solving three discrete minimization problems

$$\|\tilde{u} - \tilde{u}^v_{h,p} - \tilde{u}^{e_j}_{h,p}\|_{H^{\frac{1}{2}}_{00}(e_j)} \to \min, \quad j = 1, \dots, 3 \tag{3.4}$$

(here a tilde over a function denotes its trace). The resulting functions $u^{e_j}_{h,p}$ are obtained as (any sufficiently regular) extensions of functions $\tilde{u}^{e_j}_{h,p}$ to element interior, vanishing always along the two remaining edges. In practice one uses the shape functions $\varphi^{e_j}_{k,t}$, $k = 2, \dots, p^{e_j}$, for this purpose.

On each edge e_j the trace $\tilde{u}^{e_j}_{h,p}$ can be written as a linear combination of traces of the edge functions,

$$\tilde{u}^{e_j}_{h,p} = \sum_{m=2}^{p^{e_j}} \alpha^{e_j}_m \tilde{\varphi}^{e_j}_{m,t},$$

and the minimization problem (3.4) transforms into a system of $p^{e_j} - 1$ linear equations

$$(\tilde{u} - \tilde{u}^v_{h,p} - \tilde{u}^{e_j}_{h,p}, \tilde{\varphi}^{e_j}_{k,t})_{H^{\frac{1}{2}}_{00}} = 0, \quad k = 2, 3, \dots, p^{e_j},$$

for the unknown coefficients $\alpha^{e_j}_m$. The situation is depicted in Figure 3.3.

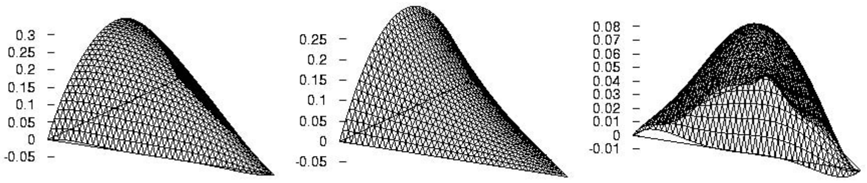

FIGURE 3.3: Decomposition of $u - u^v_{h,p}$ (left) into edge interpolant $u^e_{h,p}$ (middle) and residual $u - u^v_{h,p} - u^e_{h,p}$ (right). The residual vanishes at the vertices, but generally it does not completely vanish on edges. (Again notice different scaling.)

Bubble interpolant $u^b_{h,p} \in W_t$ is obtained by projecting the residual $u - u^v_{h,p} - u^e_{h,p}$ on the polynomial space $\mathcal{P}_{p^b,0}$, generated by the bubble functions $\varphi^b_{n_1,n_2,t}$, $1 \le n_1, n_2$, $n_1 + n_2 \le p^b - 1$ in H^1-seminorm. Since supports of the bubble shape functions lie in element interior, this operation is obviously local. Defining

$$u^b_{h,p} = \sum_{n_1=1}^{p^b-2} \sum_{n_2=1}^{p^b-n_1-1} \alpha^b_{n_1,n_2} \varphi^b_{n_1,n_2,t},$$

the corresponding discrete minimization problem,

$$|u - u^v_{h,p} - u^e_{h,p} - u^b_{h,p}|_{H^1} \to \min,$$

translates into a system of $(p^b - 2)(p^b - 1)/2$ linear algebraic equations

$$\int_{K_t} \nabla(u - u^v_{h,p} - u^e_{h,p} - u^b_{h,p}) \cdot \nabla\varphi^b_{n_1,n_2,t}\, \mathrm{d}\boldsymbol{x} = 0,$$

for unknown coefficients $\alpha^b_{n_1,n_2}$. This is illustrated in Figure 3.4.

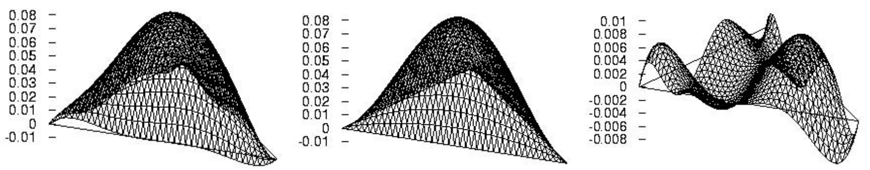

FIGURE 3.4: Decomposition of $u-u^v_{h,p}-u^e_{h,p}$ (left) into bubble interpolant $u^b_{h,p}$ (middle) and residual $u - u^v_{h,p} - u^e_{h,p} - u^b_{h,p}$ (right).

REMARK 3.2 (Uniqueness of the interpolant) Notice that the final interpolant $u_{h,p}$ is uniquely defined despite many possibilities of choice of vertex interpolant $u^v_{h,p}$ and extensions $u^{e_j}_{h,p}$ as long as they lie in the polynomial space W_t. Nonuniqueness of extensions $u^{e_j}_{h,p}$ is compensated by the bubble interpolant $u^b_{h,p}$. $\square$

REMARK 3.3 (Master quadrilateral $\mathcal{K}^1_q$) The interpolant $u_{h,p} = \Pi^1_q u$ is constructed as a sum $u^v_{h,p} + u^e_{h,p} + u^b_{h,p}$ of a vertex, edge and bubble interpolants.

First one constructs the standard bilinear interpolant $u^v_{h,p} \in W_q$ that matches the function u at vertices $v_1, \ldots, v_4$, exploiting the four vertex functions. The edge interpolant $u^e_{h,p} \in W_q$ of the difference $u - u^v_{h,p}$ is constructed one edge at a time, computing the projection of the trace $\tilde{u} - \tilde{u}^v_{h,p}$ in the $H^{\frac{1}{2}}_{00}$ norm of the edge, and then extending it into the element interior. In this case one has to solve four systems of linear algebraic equations. In the last step one computes the bubble interpolant $u^b_{h,p} \in W_q$ by projecting the difference $u - u^v_{h,p} - u^e_{h,p}$ on the polynomial space generated by the bubble functions $\varphi^b_{n_1,n_2,q}$, $2 \le n_1 \le p^{b,1}$, $2 \le n_1 \le p^{b,2}$ in the H^1 seminorm. The functions $\tilde{u}^{e_j}_{h,p}$ and the final interpolant $u_{h,p}$ are uniquely defined. $\square$

PROPOSITION 3.1 ([61])
Operators $\Pi^1_t : H^{1+\epsilon}(K_t) \to W_t$ and $\Pi^1_q : H^{1+\epsilon}(K_q) \to W_q$, $\epsilon > 0$, are well defined and bounded, with the norm independent of orders p^b, p^{e_j}.

PROOF The original result related to a reference triangular domain

$$T = \left\{ (x_1, x_2);\ x_2 > 0,\, x_2 < \sqrt{3}\left(x_1 + \frac{1}{2}\right),\, x_2 < -\sqrt{3}\left(x_1 - \frac{1}{2}\right) \right\},$$

based on the polynomial extension theorem [20, 22] and the Poincaré inequality, extends naturally to the master elements $\mathcal{K}^1_t, \mathcal{K}^1_q$. $\square$

THEOREM 3.1 (H^1-conforming interpolation error estimate [61])
Consider the master triangle $\mathcal{K}^1_t$. There exists a constant C, dependent upon ϵ but independent of the polynomial orders p^b and p^{e_j}, $j = 1, \ldots, 3$, such that

$$\|u - \Pi^1_t u\|_{H^1(K_t)} \leq C \inf_{W_t} \|u - v\|_{H^{1+\epsilon}(K_t)} \leq C(\min_j p^{e_j})^{-(r-\epsilon)} \|u\|_{H^{1+r}(K_t)},$$

for every $r > 1$ and $0 < \epsilon < r$.

PROOF The proof is based on the best approximation result for polynomial spaces [22]. $\square$

Master tetrahedron $\mathcal{K}^1_T$

Let the master tetrahedron $\mathcal{K}^1_T$ be equipped with local orders of approximation p^{e_j} on edges, p^{s_k} on faces and p^b in the interior, satisfying the minimum rule for H^1-conforming discretizations. The master element polynomial space has the form

$$W_T = \{w;\ w \in \mathcal{P}_{p^b}(K_T);\ w|_{s_i} \in \mathcal{P}_{p^{s_i}}(s_i),\ w|_{e_j} \in \mathcal{P}_{p^{e_j}}(e_j)\}.$$

Consider a sufficiently regular function u defined in K_T (the theory requires that $u \in H^{3/2+\epsilon}$, $\epsilon > 0$, see [68]). Recall that hierarchic shape functions on the master tetrahedron $\mathcal{K}^1_T$ comprise vertex functions $\varphi^{v_1}_T, \ldots, \varphi^{v_4}_t$ defined in (2.37), edge functions $\varphi^{e_1}_T, \ldots, \varphi^{e_6}_T$ from (2.38), face functions $\varphi^s_{n_1,n_2,T}$, $n_1 + n_2 \leq p^{s_j} - 1$, $1 \leq n_1, n_2$ from (2.39) and bubble functions $\varphi^b_{n_1,n_2,n_3,T}$, $1 \leq n_1, n_2, n_3$, $n_1 + n_2 + n_3 \leq p^b - 1$, defined in (2.40).

The projection-based interpolant $u_{h,p} \in W_T$ is constructed as a sum of a vertex, edge, face and bubble interpolants,

$$u_{h,p} = u^v_{h,p} + u^e_{h,p} + u^s_{h,p} + u^b_{h,p}.$$

Vertex interpolant $u^v_{h,p} \in W_T$ is a linear function that matches u at all vertices,

$$u^v_{h,p}(v_j) = u(v_j),\ j = 1, \ldots, 4.$$

It is easily obtained as a linear combination of vertex shape functions $\varphi^{v_1}_T$, ..., $\varphi^{v_4}_T$.

Edge interpolant $u^e_{h,p} \in W_T$ is constructed as a sum of contributions from the element edges,

$$u^e_{h,p} = \sum_{j=1}^{6} u^{e_j}_{h,p}.$$

Locality and conformity arguments imply that the value of $u^e_{h,p}$ along an edge e_j must depend on the values of function u on the edge only. The approximation theory says that the right norm for the edge interpolant is the L^2 norm (the trace of functions from the space $H^1(K_T)$ to faces lies in the space $H^{\frac{1}{2}}(\partial K_T)$, and the trace of the latter ones to edges loses the additional portion of regularity).

Thus, on each edge e_j one projects the trace of the difference $u - u^v_{h,p}$ on the one-dimensional polynomial space $\mathcal{P}_{p^{e_j},0}(e_j)$ on the edge e_j (generated by traces of edge functions $\varphi^{e_j}_{k,T}$, $k = 2, \ldots, p^{e_j}$) in the L^2 norm. We proceed one edge at a time, solving six discrete minimization problems

$$\|\tilde{u} - \tilde{u}^v_{h,p} - \tilde{u}^{e_j}_{h,p}\|_{L^2(e_j)} \to \min,\ j = 1, \ldots, 6. \tag{3.5}$$

Again a tilde over a function denotes its trace to the edge. The resulting functions $u^{e_j}_{h,p}$ are obtained as (any sufficiently regular) extensions of $\tilde{u}^{e_j}_{h,p}$ to element interior, vanishing along the remaining edges. In practice it is convenient to construct the extension using the shape functions $\varphi^{e_j}_{k,T}$, $k = 2, \ldots, p^{e_j}$. When putting

$$\tilde{u}^{e_j}_{h,p} = \sum_{m=2}^{p^{e_j}} \alpha^{e_j}_m \tilde{\varphi}^{e_j}_{m,T},$$

for each edge e_j, $j = 1, \ldots, 6$ the problem (3.5) is equivalent to a system of $p^{e_j} - 1$ linear equations

$$(\tilde{u} - \tilde{u}^v_{h,p} - \tilde{u}^{e_j}_{h,p}, \tilde{\varphi}^{e_j}_{m,T})_{L^2(e_j)} = 0$$

for the unknown coefficients $\alpha^{e_j}_m$.

Face interpolant $u^s_{h,p} \in W_T$ is constructed as a sum of contributions from the element faces,

$$u^s_{h,p} = \sum_{i=1}^{4} u^{s_i}_{h,p}.$$

By locality and conformity requirements, the value of $u^s_{h,p}$ on a face s_i must depend on the values of function u on s_i only. Here the projection is done in the $H^{\frac{1}{2}}$ norm. This norm is evaluated approximately (in the same way as the $H^{\frac{1}{2}}_{00}$ norm that was discussed in the triangular case). This time one solves four minimization problems

$$\|\tilde{u} - \tilde{u}^v_{h,p} - \tilde{u}^e_{h,p} - \tilde{u}^{s_i}_{h,p}\|_{H^{\frac{1}{2}}(s_i)} \to \min,\ i = 1, \ldots, 4 \tag{3.6}$$

(now the tilde stands for traces to faces). Functions $u^{s_i}_{h,p}$ are obtained as (any sufficiently regular) extensions of $\tilde{u}^{s_i}_{h,p}$ to element interior that vanishes on all remaining faces. In practice one uses the shape functions $\varphi^{s_i}_{n_1,n_2,T}$, $1 \le n_1, n_2$, $n_1 + n_2 \le p^{s_i} - 1$ to extend the traces $\tilde{u}^{s_i}_{h,p}$ to element interior.

Expressing

$$\tilde{u}^{s_i}_{h,p} = \sum_{n_1=1}^{p^{s_i}-2} \sum_{n_2=1}^{p^{s_i}-n_1-1} \alpha^{s_i}_{n_1,n_2} \tilde{\varphi}^{s_i}_{n_1,n_2,T},$$

for each face s_i, $i = 1, \ldots, 4$, the problem (3.6) yields a system of $(p^{s_i} - 2)(p^{s_i} - 1)/2$ linear equations

$$(\tilde{u} - \tilde{u}^v_{h,p} - \tilde{u}^e_{h,p} - \tilde{u}^{s_i}_{h,p}, \tilde{\varphi}^{s_i}_{n_1,n_2,T})_{H^{\frac{1}{2}}(s_i)} = 0$$

for the unknown coefficients $\alpha^{s_i}_{n_1,n_2}$.

Bubble interpolant $u^b_{h,p} \in W_T$ is obtained by projecting the residual $u - u^v_{h,p} - u^e_{h,p} - u^s_{h,p}$ on the polynomial space $\mathcal{P}_{p^b,0}(K_T)$ (generated by the bubble functions $\varphi^b_{n_1,n_2,n_3,T}$, $1 \le n_1, n_2, n_3$, $n_1 + n_2 + n_3 \le p^b - 1$) in the H^1 seminorm.

With the substitution

$$u^b_{h,p} = \sum_{n_1=1}^{p^b-3} \sum_{n_2=1}^{(p^b-n_1-2)} \sum_{n_3=1}^{(p^b-n_1-n_2-1)} \alpha^b_{n_1,n_2,n_3} \varphi^b_{n_1,n_2,n_3,T},$$

the corresponding discrete minimization problem,

$$|(u - u^v_{h,p} - u^e_{h,p} - u^s_{h,p}) - u^b_{h,p}|_{H^1(K_T)} \to \min,$$

yields a final system of $(p^b - 3)(p^b - 2)(p^b - 1)/6$ linear algebraic equations,

$$\int_{K_T} \nabla(u - u^v_{h,p} - u^e_{h,p} - u^s_{h,p} - u^b_{h,p}) \cdot \nabla\varphi^b_{n_1,n_2,n_3,T}\, \mathrm{d}\boldsymbol{x} = 0,$$

for the unknown coefficients $\alpha^b_{n_1,n_2,n_3}$.

REMARK 3.4 (Master elements $\mathcal{K}^1_B$ and $\mathcal{K}^1_P$) The interpolants $\Pi^1_B u$ and $\Pi^1_P u$ on master elements $\mathcal{K}^1_B$ and $\mathcal{K}^1_P$ are constructed exactly in the same way. They comprise a vertex, edge, face and bubble interpolants $u^v_{h,p}$, $u^e_{h,p}$, $u^s_{h,p}$ and $u^b_{h,p}$. The projection is done in L^2 norm on edges, $H^{\frac{1}{2}}$ on faces and in H^1 seminorm in element interior. $\square$

Let us mention a few theoretical results [68] for the tetrahedral projection-based interpolation operator Π^1_T, based on a few conjectures, suitable polynomial extension theorems and on the Poincaré inequality.

PROPOSITION 3.2 ([68])
Operator $\Pi^1_T : H^{3/2+\epsilon}(K_T) \to W_T$, $\epsilon > 0$, is well defined and bounded, with the norm independent of orders p^b, p^{s_i} and p^{e_j}.

Conjecture 3.1 (Polynomial extension map [68]) *There exists a polynomial extension map,*

$$A : \mathcal{P}_p(\partial K_T) \to \mathcal{P}_p(K_T), \quad (Au)|_{\partial K_T} = u \quad \forall u \in \mathcal{P}_p(\partial K_T),$$

such that

$$||Au||_{H^1,K_T} \le C||u||_{H^{\frac{1}{2}},\partial K_T},$$

with constant C independent of p.

REMARK 3.5 In Conjecture 3.1, $\mathcal{P}_p(\partial K_T)$ stands for the space of continuous functions defined on the boundary of ∂K_T, whose restrictions to element faces reduce to polynomials of order p. The conjecture postulates a 3D equivalent of the 2D result established in [61]. $\square$

Conjecture 3.2 (Polynomial extension map [68]) *There exists a polynomial extension map,*

$$A : \mathcal{P}_p(\partial s) \to \mathcal{P}_p(s), \quad (Au)|_{\partial s} = u \quad \forall u \in \mathcal{P}_p(\partial s),$$

such that

$$||Au||_{H^{\frac{1}{2}+\epsilon},s} \le C||u||_{H^\epsilon,\partial s},$$

with constant C independent of p.

REMARK 3.6 In Conjecture 3.2, $\mathcal{P}_p(\partial s)$ stands for the space of continuous functions defined on face boundary ∂s, whose restriction to the face edges reduce to polynomials of order p. $\square$

LEMMA 3.1 ([68])
Let $I = (-1,1)$. For a given $u \in H^1(I)$, let u^p denote the L^2-projection of function u on space $\mathcal{P}_{p,0}$ of polynomials of order p, vanishing at the endpoints. Then, for every $\epsilon > 0$, there exists a constant $C = C(\epsilon)$, dependent upon ϵ, but independent of polynomial order p and function u, such that

$$||u - u^p||_{L^2,I} \leq \frac{C}{p^{1-\epsilon}}||u||_{H^1,I}.$$

THEOREM 3.2 (H^1 interpolation error estimate [68])
Given the above Conjectures 3.1, 3.2 on the polynomial extension, we have

$$||u - \Pi^1_T u||_{H^1,K_T} \leq C(\epsilon)p^{r-\epsilon}||u||_{H^r,K_T},$$

where $C(\epsilon) > 0$ and p is the minimum order of approximation for the element interior, the element faces and the element edges.

3.1.2 $\boldsymbol{H}(\text{curl})$-conforming elements

Let us now turn our attention to projection-based interpolation operators for $\boldsymbol{H}(\text{curl})$-conforming approximations. We will use the symbols $\mathcal{P}$ and $\mathbf{P}$ for spaces of scalar and vector-valued polynomials, respectively. The procedure will be presented for the reference triangle K_t in 2D and for the reference tetrahedron K_T in three spatial dimensions. The generalization to other element types is straightforward.

Master triangle $\mathcal{K}_t^{\text{curl}}$

Let the triangular master element $\mathcal{K}_t^{\text{curl}}$ be equipped with local orders of approximation p^b in the interior and p^{e_j}, $j = 1,2,3$, on edges. The finite element space $\boldsymbol{Q}_t$ has the form (2.55),

$$\begin{aligned}\boldsymbol{Q}_t = \{&\boldsymbol{E};\ \boldsymbol{E} \in \mathbf{P}^{p^b}(K_t);\\ &\boldsymbol{E}\cdot\boldsymbol{t}_j|_{e_j} \in \mathcal{P}_{p^{e_j}}(e_j),\ p^{e_j} \leq p^b,\ j = 1,2,3\}.\end{aligned}$$

Here $\boldsymbol{t}_j$ stands for the unitary tangential vector to the oriented edge e_j.

Consider a sufficiently regular function $\boldsymbol{E}$ defined in K_t (theory requires that $\boldsymbol{E} \in \boldsymbol{H}^\epsilon \cap \boldsymbol{H}(\text{curl})$, $\epsilon > 0$). Recall that $\boldsymbol{H}(\text{curl})$-hierarchic shape functions on the master triangle $\mathcal{K}_t^{\text{curl}}$ comprise edge functions $\psi^{e_j}_{n,t}$, $0 \leq n \leq p^{e_j}$, given by (2.57), (2.58) and (2.59), and (edge-based and genuine) bubble functions (2.60), (2.61).

Projection-based interpolant $\boldsymbol{E}_{h,p} = \Pi_t^{\text{curl}}\boldsymbol{E} \in \boldsymbol{Q}_t$ is constructed as a sum of Whitney, higher-order edge and bubble interpolants,

$$\boldsymbol{E}_{h,p} = \boldsymbol{E}^w_{h,p} + \boldsymbol{E}^e_{h,p} + \boldsymbol{E}^b_{h,p}.$$

The *Whitney interpolant* $\boldsymbol{E}^w_{h,p} \in \boldsymbol{Q}_t$ is defined as

$$\boldsymbol{E}^w_{h,p} = \sum_{j=1}^{3} \left(\int_{e_j} (\boldsymbol{E} \cdot \boldsymbol{t}_j) \, \mathrm{d}s \right) \psi^{e_j}_{0,t},$$

where $\psi^{e_j}_{0,t}$ are the Whitney shape functions (2.57).

Edge interpolant $\boldsymbol{E}^e_{h,p} \in \boldsymbol{Q}_t$ is constructed as a sum of edge contributions $\boldsymbol{E}^{e_j}_{h,p}$, $j = 1, \ldots, 3$. The construction of the Whitney interpolant yields that the trace of tangential component $(\boldsymbol{E} - \boldsymbol{E}^w_{h,p}) \cdot \boldsymbol{t}_j$ has zero average over each edge e_j. This means that for each edge e_j one can introduce a scalar function θ_{e_j}, defined on e_j, which vanishes at its endpoints and satisfies

$$\frac{\partial \theta_{e_j}}{\partial s} = (\boldsymbol{E} - \boldsymbol{E}^w_{h,p}) \cdot \boldsymbol{t}_j.$$

Next one constructs projection $\theta^{p+1}_{e_j}$ of the function θ_{e_j} on the polynomial space $\mathcal{P}_{p^{e_j}+1,0}(e_j)$ in the $H^{\frac{1}{2}}_{00}$ norm. This yields a discrete minimization problem,

$$||\theta^{p+1}_{e_j} - \theta_{e_j}||_{H^{\frac{1}{2}}_{00}(e_j)} \to \min,$$

which translates into a system of p^{e_j} linear equations

$$\left(\theta^{p+1}_{e_j} - \theta_{e_j}, \tilde{\varphi}^{e_j}_{k,t} \right)_{H^{\frac{1}{2}}_{00}(e_j)} = 0, \; k = 2, \ldots, p^{e_j} + 1,$$

(one can use traces $\tilde{\varphi}^{e_j}_{k,t}$ of scalar edge shape functions to generate the polynomial space) for the unknown coefficients α_m,

$$\theta^{p+1}_{e_j} = \sum_{m=2}^{p^{e_j}+1} \alpha_m \tilde{\varphi}^{e_j}_{m,t}.$$

See Paragraph 3.1.1, triangular case, for the approximate evaluation of the norm $H^{\frac{1}{2}}_{00}$. One takes any polynomial extension $\theta^{p+1}_{e_j,ext} \in \mathcal{P}_{p^b+1,p^{e_j}+1}(K_t)$, $\theta^{p+1}_{e_j,ext}|_{e_j} \equiv \theta^{p+1}_{e_j}$ of the projection $\theta^{p+1}_{e_j}$, which vanishes along the two remaining edges. The vector-valued edge interpolant $\boldsymbol{E}^{e_j}_{h,p}$ is finally constructed as a gradient of this extension,

$$\boldsymbol{E}^{e_j}_{h,p} = \nabla \theta^{p+1}_{e_j,ext} \in \boldsymbol{Q}_t.$$

Bubble interpolant $\boldsymbol{E}^b_{h,p} \in \boldsymbol{Q}_t$ is sought in the space $\boldsymbol{P}_{p^b,0}(K_t)$ of vector-valued polynomials of order lower than or equal to p^b in K_t, with traces of tangential component vanishing on the edges e_j, $j = 1, \dots, 3$. This leads to the discrete minimization problem

$$\|\mathrm{curl}(\boldsymbol{E}^b_{h,p} - (\boldsymbol{E} - \boldsymbol{E}^w_{h,p} - \boldsymbol{E}^e_{h,p}))\|_{L^2} = \|\mathrm{curl}(\boldsymbol{E}^b_{h,p} - (\boldsymbol{E} - \boldsymbol{E}^w_{h,p}))\|_{L^2} \to \min$$

(recall that $\boldsymbol{E}^e_{h,p}$ is a gradient, and therefore $\mathrm{curl}(\boldsymbol{E}^e_{h,p}) = 0$), with the Helmholtz decomposition constraint,

$$(\boldsymbol{E}^b_{h,p} - (\boldsymbol{E} - \boldsymbol{E}^w_{h,p} - \boldsymbol{E}^e_{h,p}), \nabla \varphi^b_{n_1,n_2,t}) = 0,$$

$1 \le n_1, n_2,\ n_1 + n_2 \le p^b$. Here $\varphi^b_{n_1,n_2,t}$ are scalar bubble functions spanning the space $\mathcal{P}_{p^b+1,0}(K_t)$ of polynomials of order lower than or equal to $p^b + 1$ that vanish on the boundary ∂K_t.

PROPOSITION 3.3 ([61])
Operator $\Pi^{\mathrm{curl}}_t : \boldsymbol{H}^\epsilon \cap \boldsymbol{H}(\mathrm{curl}) \to \boldsymbol{Q}_t$ *is well defined and bounded, with the norm independent of orders* p^b, p^{e_j}.

THEOREM 3.3 ($\boldsymbol{H}(\mathrm{curl})$ ***interpolation error estimate [61])***
Consider the master triangle $\mathcal{K}^{\mathrm{curl}}_t$. *There exists a constant* $C > 0$, *dependent upon* ϵ *but independent of* p^b, p^{e_j}, $j = 1, \dots, 3$, *such that*

$$\|\boldsymbol{E} - \Pi^{\mathrm{curl}}_t \boldsymbol{E}\|_{H(\mathrm{curl})} \le C \inf_{\boldsymbol{F} \in \boldsymbol{Q}_t} \left(\|\boldsymbol{E} - \boldsymbol{F}\|^2_{H^\epsilon} + \|\mathrm{curl}(\boldsymbol{E} - \boldsymbol{F})\|^2_{L^2}\right)^{\frac{1}{2}}$$

$$\le C(\min_{j=1,\dots,3} p^{e_j})^{-(r-\epsilon)} \left(\|\boldsymbol{E}\|^2_{H^r} + \|\mathrm{curl}\boldsymbol{E}\|^2_{H^r}\right)^{\frac{1}{2}},$$

for every $0 < r < 1$ *and* $0 < \epsilon < r$.

Master tetrahedron $\mathcal{K}^{\mathrm{curl}}_T$

Local projection-based interpolation operators for $\boldsymbol{H}(\mathrm{curl})$-conforming elements in 3D were first introduced in [68]. Consider the master tetrahedron $\mathcal{K}^{\mathrm{curl}}_T$ from Paragraph 2.2.3, with local orders of approximation p^b in its interior, p^{s_i}, $i = 1, \dots, 4$ on faces, and p^{e_j}, $j = 1, \dots, 6$ on edges. The polynomial space $\boldsymbol{Q}_T$ has the form (2.72),

$$\begin{aligned}\boldsymbol{Q}_T = \{&\boldsymbol{E} \in (\mathcal{P}_{p^b})^3(K_T),\ \boldsymbol{E}_t|_{s_i} \in (\mathcal{P}_{p^{s_i}})^2(s_i),\\ &\boldsymbol{E} \cdot \boldsymbol{t}|_{e_j} \in \mathcal{P}_{p^{e_j}}(e_j),\ i = 1, \dots, 4,\ j = 1, \dots, 6\}.\end{aligned}$$

The theory requires that the projected function $\boldsymbol{E}$ satisfies regularity assumptions $\boldsymbol{E} \in \boldsymbol{H}^{\frac{1}{2}+\epsilon}, \boldsymbol{\nabla} \times \boldsymbol{E} \in \boldsymbol{H}^\epsilon$, $\epsilon > 0$.

The projection-based interpolant $\boldsymbol{E}_{h,p} = \Pi_T^{\text{curl}}\boldsymbol{E} \in \boldsymbol{Q}_T$ is constructed as a sum of Whitney, higher-order edge, face and bubble interpolants,

$$\boldsymbol{E}_{h,p} = \boldsymbol{E}_{h,p}^w + \boldsymbol{E}_{h,p}^e + \boldsymbol{E}_{h,p}^s + \boldsymbol{E}_{h,p}^b.$$

Let s_i be a face of the reference domain K_T. For $\boldsymbol{E} \in \boldsymbol{H}^\epsilon(s_i)$, $\text{curl}_{s_i}\boldsymbol{E} \in H^{-\frac{1}{2}+\epsilon}$, the restriction of the tangential component $\boldsymbol{E}_t$ to face boundary ∂s belongs to the space $H^{-\frac{1}{2}+\epsilon}$. Here

$$\text{curl}_{s_i}\boldsymbol{E} = \boldsymbol{n}_i \cdot (\boldsymbol{\nabla} \times \boldsymbol{E})$$

is the surface curl-operator. Hence the edge averages

$$\int_{e_j} \boldsymbol{E} \cdot \boldsymbol{t}_j \text{de},$$

$j = 1, \ldots, 6$, are well defined, and one can construct the *Whitney interpolant* $\boldsymbol{E}_{h,p}^w \in \boldsymbol{Q}_T$ as

$$\boldsymbol{E}_{h,p}^w = \sum_{j=1}^{6} \left(\int_{e_j} (\boldsymbol{E} \cdot \boldsymbol{t}_j) \, \text{d}s \right) \psi_{0,T}^{e_j},$$

where $\psi_{0,T}^{e_j}$ are the Whitney shape functions $\psi_{0,T}^{e_j}$ (2.75).

Similarly as in two dimensions, for each edge e_j one introduces a scalar potential $\theta_{e_j} \in H^{\frac{1}{2}+\epsilon}(e_j)$, vanishing at its endpoints, such that

$$\frac{\partial \theta_{e_j}}{\partial s} = (\boldsymbol{E} - \boldsymbol{E}_{h,p}^w) \cdot \boldsymbol{t}_j.$$

One projects the potential in H^ϵ-norm on the polynomial spaces $\mathcal{P}_{p^{e_j}+1,0}(e_j)$, solving discrete minimization problems

$$||\theta_{e_j}^{p+1} - \theta_{e_j}||_{H^\epsilon(e_j)} \rightarrow \min,$$

for the projections $\theta_{e_j}^{p+1}$ on edges. The final *edge interpolant* $\boldsymbol{E}_{h,p}^e \in \boldsymbol{Q}_T$ is constructed as a sum

$$\boldsymbol{E}_{h,p}^e = \sum_{j=1}^{6} \boldsymbol{E}_{h,p}^{e_j},$$

where the edge contributions $\boldsymbol{E}_{h,p}^{e_j}$ are obtained as gradients of an extension $\theta_{e_j,ext}^{p+1}$ of the scalar potential $\theta_{e_j}^{p+1}$, such that $\theta_{e_j,ext}^{p+1}|_{e_k} \equiv 0$ for all $k \neq j$, into the element interior,

$$\boldsymbol{E}_{h,p}^{e_j} = \nabla\theta_{e_j,ext}^{p+1} \in \boldsymbol{Q}_T.$$

Face interpolant: on each face s_i one solves the discrete minimization problem

$$\begin{aligned} \|\mathrm{curl}_{s_i}(\boldsymbol{E}^{s_i}_{h,p} - (\boldsymbol{E} - \boldsymbol{E}^{w}_{h,p} - \boldsymbol{E}^{e}_{h,p}))\|_{\mathrm{curl},-\frac{1}{2}+\epsilon,s_i} &= \\ \|\mathrm{curl}_{s_i}(\boldsymbol{E}^{s_i}_{h,p} - (\boldsymbol{E} - \boldsymbol{E}^{w}_{h,p}))\|_{\mathrm{curl},-\frac{1}{2}+\epsilon,s_i} &\to \min \end{aligned} \tag{3.7}$$

(recall that $\boldsymbol{E}^{e}_{h,p}$ is a gradient), where the face interpolant $\boldsymbol{E}^{s_i}_{h,p} \in \boldsymbol{P}_{p^{s_i},0}(s_i)$. Here $\boldsymbol{P}_{p^{s_i},0}(s_i)$ stands for the space of vector-valued polynomials, defined on the face s_i, whose tangential components vanish on the boundary ∂s_i. Expression (3.7) is minimized with the Helmholtz decomposition constraint,

$$(\boldsymbol{E}^{s_i}_{h,p} - (\boldsymbol{E} - \boldsymbol{E}^{w}_{h,p} - \boldsymbol{E}^{e}_{h,p}), \nabla\varphi^{s_i}_{n_1,n_2,T})_{\mathrm{curl},-\frac{1}{2}+\epsilon,s_i} = 0,$$

$1 \le n_1, n_2,\ n_1 + n_2 \le p^{s_i}$. Here $\varphi^{s_i}_{n_1,n_2,T}$ are scalar face functions, whose traces span the space $\mathcal{P}_{p^{s_i}+1,0}(s_i)$ of polynomials of order lower than or equal to $p^{s_i} + 1$ on the face s_i, vanishing on the boundary ∂s_i. The right norm for the minimization (see [68]) is

$$\|\boldsymbol{E}\|^2_{\mathrm{curl},-\frac{1}{2}+\epsilon,s_i} = \|\boldsymbol{E}\|^2_{-\frac{1}{2}+\epsilon,s_i} + \|\boldsymbol{n}\cdot(\boldsymbol{\nabla}\times\boldsymbol{E})\|^2_{\frac{1}{2}+\epsilon,s_i}.$$

Face interpolant $\boldsymbol{E}^{s}_{h,p} \in \boldsymbol{Q}_T$ is obtained in a standard way using polynomial extensions of $E^{s_i}_{h,p}$ into the element interior with tangential component vanishing on all other faces.

Finally, the *bubble interpolant* $\boldsymbol{E}^{b}_{h,p}$ is obtained as a solution of constrained minimization problem

$$\begin{aligned} \|\mathrm{curl}(\boldsymbol{E}^{b}_{h,p} - (\boldsymbol{E} - \boldsymbol{E}^{w}_{h,p} - \boldsymbol{E}^{e}_{h,p} - \boldsymbol{E}^{s}_{h,p}))\|_{L^2,K_T} &= \\ \|\mathrm{curl}(\boldsymbol{E}^{s_i}_{h,p} - (\boldsymbol{E} - \boldsymbol{E}^{w}_{h,p} - \boldsymbol{E}^{s}_{h,p}))\|_{L^2,K_T} &\to \min . \end{aligned} \tag{3.8}$$

The bubble interpolant $\boldsymbol{E}^{b}_{h,p}$ now lies in the space $\boldsymbol{Q}_{T,0}$. In the same way as before, one has to solve simultaneously the constraint

$$(\boldsymbol{E}^{b}_{h,p} - (\boldsymbol{E} - \boldsymbol{E}^{w}_{h,p} - \boldsymbol{E}^{e}_{h,p} - \boldsymbol{E}^{s}_{h,p}), \nabla\varphi^{b}_{n_1,n_2,n_3,T})_{L^2,K_T} = 0,$$

$1 \le n_1, n_2, n_3,\ n_1+n_2+n_3 \le p^b$. Here $\varphi^{b}_{n_1,n_2,n_3,T}$ are scalar bubble functions, which span the space $\mathcal{P}_{p^b+1,0}(K_T)$ of polynomials of order lower than or equal to $p^b + 1$, vanishing on the boundary ∂K_T. Herewith, the projection-based interpolation operator Π^{curl}_T is defined. Let us mention an error estimate for this operator from [68].

Conjecture 3.3 (2D discrete Friedrichs inequality [68]) *Let s_i be a face of the reference tetrahedron K_T. There exists constant $C > 0$, independent of p, such that*

$$\|\boldsymbol{E}\|_{\mathrm{curl},-\frac{1}{2}+\epsilon,\partial s_i} \le C\|\boldsymbol{\nabla}\times\boldsymbol{E}\|_{\mathrm{curl},-\frac{1}{2}+\epsilon,\partial s_i}$$

for all polynomials $\boldsymbol{E}$ such that

$$\boldsymbol{E} \in \boldsymbol{P}_{p^{s_i},0}(s_i) \text{ and } (\boldsymbol{E}, \boldsymbol{\nabla}\varphi)_{\mathrm{curl},-\frac{1}{2}+\epsilon,\partial s_i} = 0, \quad \text{for all } \varphi \in \mathcal{P}_{p^{s_i}+1,0}.$$

THEOREM 3.4 (3D discrete Friedrichs inequality [68])
There exists constant $C > 0$, independent of p, such that

$$\|\boldsymbol{E}\|_{L^2,K_T} \leq C\|\boldsymbol{\nabla} \times \boldsymbol{E}\|_{L^2,K_T}$$

for all polynomials $\boldsymbol{E}$ such that

$$\boldsymbol{E} \in \boldsymbol{P}_{p^b,0}(K_T)$$

and

$$(\boldsymbol{E}, \boldsymbol{\nabla}\varphi) = 0 \quad \text{for all } \varphi \in \mathcal{P}_{p^b+1,0}(K_T),$$

where $\mathcal{P}_{p^b+1,0}(K_T)$ is the space of scalar polynomials of order at most $p^b + 1$ which entirely vanish on the element boundary.

THEOREM 3.5 ($\boldsymbol{H}$(curl) interpolation error estimate [68])
Given the above Conjectures 3.1, 3.2 on the polynomial extension, and Conjecture 3.3 on discrete Friedrichs inequality in $H^{-\frac{1}{2}+\epsilon}$ norm on a triangle, we have

$$||\boldsymbol{E} - \Pi^{\mathrm{curl}}_{K_T}\boldsymbol{E}||_{\mathrm{curl},0,K_T} \leq C(\epsilon)p^{-(r-\epsilon)}||\boldsymbol{E}||_{\mathrm{curl},r,K_T},$$

where $C(\epsilon) > 0$.

Operators Π^{curl}_B, Π^{curl}_P for the master elements $\mathcal{K}^{\mathrm{curl}}_B$ and $\mathcal{K}^{\mathrm{curl}}_P$ are constructed analogously, taking into account relevant directional order of approximation associated with the element interior and quadrilateral faces.

3.1.3 $\boldsymbol{H}$(div)-conforming elements

As mentioned before, the space $\boldsymbol{H}$(div) is in reality not much different from the space $\boldsymbol{H}$(curl) in 2D. This applies also to the projection-based interpolation operators – they will be defined exactly in the same way as in the $\boldsymbol{H}$(curl)-conforming case in Paragraph 3.1.2, only that the tangential and normal directions on edges will be switched.

Master triangle $\mathcal{K}^{\mathrm{div}}_t$

We confine ourselves to the master triangle $\mathcal{K}^{\mathrm{div}}_t$ in 2D. Consider local orders of approximation p^b in its interior, and p^{e_j}, $j = 1, 2, 3$, on edges. Now the finite element space $\boldsymbol{V}_t$ has the form (2.97),

$$\boldsymbol{V}_t = \{\boldsymbol{v};\ \boldsymbol{v} \in \mathbf{P}^{p^b}(K_t);\ \boldsymbol{v}\cdot\boldsymbol{n}_j|_{e_j} \in \mathcal{P}_{p^{e_j}}(e_j),\ p^{e_j} \le p^b,\ j = 1,2,3\},$$

where $\boldsymbol{n}_j$ stands for unitary outer normal vector to the edge e_j.

Recall that $\boldsymbol{H}(\mathrm{div})$-hierarchic shape functions comprise edge functions $\gamma_{n,t}^{e_j}$, $0 \le n \le p^{e_j}$, given by (2.99), (2.100) and (2.101), and (edge-based and genuine) bubble functions (2.102), (2.103).

Projection-based interpolant $\boldsymbol{v}_{h,p} = \Pi_t^{\mathrm{div}}\boldsymbol{v} \in \boldsymbol{V}_t$ of a sufficiently regular function $\boldsymbol{v}$ is constructed as a sum of Whitney, higher-order edge and bubble interpolants,

$$\boldsymbol{v}_{h,p} = \boldsymbol{v}_{h,p}^{w} + \boldsymbol{v}_{h,p}^{e} + \boldsymbol{v}_{h,p}^{b}.$$

Whitney interpolant $\boldsymbol{v}_{h,p}^{w} \in \boldsymbol{V}_t$ is defined as

$$\boldsymbol{v}_{h,p}^{w} = \sum_{j=1}^{3}\left(\int_{e_j}(\boldsymbol{v}\cdot\boldsymbol{n}_j)\,\mathrm{d}s\right)\gamma_{0,t}^{e_j}.$$

Edge interpolant $\boldsymbol{v}_{h,p}^{e} \in \boldsymbol{V}_t$ is constructed one edge at a time,

$$\boldsymbol{v}_{h,p}^{e} = \sum_{j=1}^{3}\boldsymbol{v}_{h,p}^{e_j}.$$

The trace of normal component $(\boldsymbol{v}-\boldsymbol{v}_{h,p}^{w})\cdot\boldsymbol{n}_j$ has zero average over each edge e_j, which means that for each edge e_j one can introduce a scalar function θ_{e_j}, defined on e_j, which vanishes at its endpoints and satisfies

$$\frac{\partial\theta_{e_j}}{\partial s} = (\boldsymbol{v}-\boldsymbol{v}_{h,p}^{w})\cdot\boldsymbol{n}_j.$$

Next one constructs projection $\theta_{e_j}^{p+1}$ of the function θ_{e_j} on the polynomial space $\mathcal{P}_{p^{e_j}+1}(e_j)$, and its extension $\theta_{e_j,ext}^{p+1}$ into element interior, exactly in the same way as in Paragraph 3.1.2.

This time at the end one takes **curl** of the extension,

$$\boldsymbol{v}_{h,p}^{e_j} = \mathbf{curl}(\theta_{e_j,ext}^{p+1}) \in \boldsymbol{V}_t$$

(recall from Remark 2.1 that $\mathbf{curl}(a) = (-\partial a/\partial x_2, \partial a/\partial x_1)$).

Bubble interpolant $\boldsymbol{v}_{h,p}^{b} \in \boldsymbol{V}_t$ is sought in the space $\boldsymbol{P}_{p^b,0}(K_t)$ of vector-valued polynomials of order lower than or equal to p^b in K_t, with traces of normal component vanishing on the edges e_j, $j = 1,\ldots,3$. One solves the discrete minimization problem,

$$\|\mathrm{div}(\boldsymbol{v}^b_{h,p} - (\boldsymbol{v} - \boldsymbol{v}^w_{h,p} - \boldsymbol{v}^e_{h,p}))\|_{L^2} = \|\mathrm{div}(\boldsymbol{v}^b_{h,p} - (\boldsymbol{v} - \boldsymbol{v}^v_{h,p}))\|_{L^2} \to \min$$

(recall that $\boldsymbol{v}^e_{h,p}$ is a **curl**, and therefore $\mathrm{div}(\boldsymbol{v}^e_{h,p}) = 0$), with the constraint,

$$(\boldsymbol{v}^b_{h,p} - (\boldsymbol{v} - \boldsymbol{v}^v_{h,p} - \boldsymbol{v}^e_{h,p}), \mathbf{curl}(\varphi^b_{n_1,n_2,t})) = 0,$$

$1 \le n_1, n_2,\ n_1 + n_2 \le p^b$. Here $\varphi^b_{n_1,n_2,t}$ are scalar bubble functions spanning the space $\mathcal{P}_{p^b+1}(K_t)$.

Master tetrahedron $\mathcal{K}^{\mathrm{div}}_T$

Finally let us construct the projection-based interpolation operator Π^{div}_T, first introduced in [68]. We will project a function $\boldsymbol{v} \in \boldsymbol{H}^r(K_T)$, $r > 0$, $\boldsymbol{\nabla} \cdot \boldsymbol{v} \in L^2(K_T)$, whose trace of the normal component $\boldsymbol{v} \cdot \boldsymbol{n}|_{\partial K_T}$ belongs to the scalar space $H^{-\frac{1}{2}+\epsilon}(\partial K_T)$. Consider local order of approximation p^b in element interior and local orders $p^{s_1}, \ldots, p^{s_4}$ on faces.

Let us begin with the face interpolant $\boldsymbol{v}^s \in \boldsymbol{V}_T$. Consider a face $s_i \subset \partial K_T$. Trace of the normal component of the face interpolant $v^{s_i}_n \in \mathcal{P}_{p^{s_i}}(s_i)$ is constructed using $H^{-\frac{1}{2}+\epsilon}$ projection on the face s_i, i.e., solving the discrete minimization problem

$$\|\boldsymbol{n} \cdot (\boldsymbol{v} - \boldsymbol{v}^{s_i})|_{s_i}\|_{-\frac{1}{2}+\epsilon, s_i} \to \min,$$

where $\boldsymbol{v}^{s_i}$ is sought as a linear combination of genuine and edge-based face functions corresponding to the local order of approximation p^{s_i} on the face s_i of the master element $\mathcal{K}^{\mathrm{div}}_T$. The *face interpolant* $\boldsymbol{v}^s$ is obtained as a sum of contributions over all element faces.

Bubble interpolant $\boldsymbol{v}^b \in \boldsymbol{V}_T$ is constructed by solving the discrete minimization problem

$$\|\boldsymbol{\nabla} \cdot (\boldsymbol{v} - \boldsymbol{v}^b - \boldsymbol{v}^s)\|_{L^2, K_T} \to \min,$$

with the constraint

$$(\boldsymbol{v} - (\boldsymbol{v}^b + \boldsymbol{v}^s), \boldsymbol{\nabla} \times \psi) = 0, \quad \text{for all } \psi \in \boldsymbol{P}_{p^b+1,0,0},$$

where $\boldsymbol{P}_{p^b+1,0,0}$ is the space of vector-valued polynomials of the order at most p^b, whose tangential component vanishes entirely on the boundary ∂K_T.

THEOREM 3.6 ("Friedrichs inequality" for the $\boldsymbol{\nabla}\cdot$ operator [68])
There exists constant $C > 0$, independent of p, such that

$$\|\boldsymbol{v}\|_{L^2, K_T} \le C \|\boldsymbol{\nabla} \cdot \boldsymbol{v}\|_{L^2, K_T}$$

for all polynomials $\boldsymbol{v}$ *such that*

$$\boldsymbol{v} \in \boldsymbol{P}_{p^b,0}(K_T)$$

and

$$(\boldsymbol{v}, \boldsymbol{\nabla} \times \psi) = 0 \quad \textit{for all } \psi \in \boldsymbol{P}_{p^b+1,0,0}(K_T),$$

where $\boldsymbol{P}_{p^b,0}(K_T)$ *is the space of vector-valued polynomials of order at most* p^b *whose* normal *component vanishes on the whole element boundary, and* $\mathcal{P}_{p^b+1,0,0}(K_T)$ *is the space of vector-valued polynomials of order at most* p^b+1 *whose* tangential *component vanishes entirely on the element boundary.*

Finally let us postulate another conjecture on polynomial extension, and mention an error estimate for the operator Π_T^{div}.

Conjecture 3.4 (Polynomial extension [68]) *There exists a polynomial extension map,*

$$A : \mathcal{P}_p(\partial K_T) \to \boldsymbol{P}_p(K_T), \quad \boldsymbol{n} \cdot (Av)|_{\partial K_T} = u \quad \textit{for all } u \in \mathcal{P}_p(\partial K_T),$$

such that

$$||Av||_{\mathrm{div},0,K_T} \leq C||v||_{\mathrm{div},0,K_T},$$

with constant C *independent of* p.

THEOREM 3.7 (*$\boldsymbol{H}(\mathrm{div})$ *interpolation error estimate [68])
Given the above Conjecture 3.4 and Theorem 3.6, we have

$$||\boldsymbol{v} - \Pi_{K_T}^{\mathrm{div}} \boldsymbol{v}||_{\mathrm{div},0,K_T} \leq C(\epsilon) p^{-(r-\epsilon)} ||\boldsymbol{v}||_{\mathrm{div},r,K_T},$$

where $C(\epsilon) > 0$.

Operators Π_B^{div}, Π_P^{div} for the master elements $\mathcal{K}_B^{\mathrm{curl}}$ and $\mathcal{K}_P^{\mathrm{curl}}$ are constructed analogously.

3.2 Transfinite interpolation revisited

After defining the projection-based interpolation operators on reference domains in the previous section, the next logical step in the presentation of the higher-order finite element technology would be to design the reference maps. However, before we do so, let us take a short excursion into the field of *transfinite interpolation techniques*. Bivariate and trivariate transfinite interpolation schemes are extensively used in finite element codes for the definition of parametrizations of two- and three-dimensional curved domains based on the known parametrization of their surfaces. However, not everyone seems to know that there is more to the transfinite interpolation than the notoriously known formulae.

Transfinite interpolation technique was first introduced by Steven A. Coons [55] in the 1960s, became very popular and was quickly extended into many directions (see, e.g., [97, 99, 98, 83, 84, 95] and references therein). In his original paper, Coons described a class of methods for constructing an interpolatory surface which coincides with arbitrary prescribed curves (and normal derivatives, if desired) on the boundary of the unit square. Since the *precision set* of this class of interpolation formulae (i.e., the set of points on which the interpolant exactly matches the interpolated function) is nondenumerable, these methods were later referred to as *transfinite*.

The "Coons surfaces" have been widely used in connection with problems of computer aided design and numerical control production of free-form surfaces such as ship hulls, airplane fuselages and automobile exteriors [83, 84, 95]. In addition to such geometric applications these approximation formulae provided the basis for development of new numerical schemes for the approximate integration of multivariate functions in [96]. The details of some of these methods, based upon polynomial blending, have been investigated in [27]. Other applications to multivariate data smoothing and to the approximate solution of integral equations, partial differential equations and variational problems are addressed in [93].

3.2.1 Projectors

The transfinite interpolation technique relies on sophisticated algebraic theory of multivariate approximation (see [94] and references therein). Although this theory lies beyond the scope of this book, we find it useful to introduce at least the notion of *projectors*, which give a general framework to transfinite interpolation formulae that we will utilize.

Consider a scalar continuous function f of two independent variables, defined (for example) in the reference domain K_q in the $\boldsymbol{\xi}$-plane. We seek approximations $\tilde{f} \approx f$, which interpolate f on certain (denumerable or nondenumerable) point sets contained in K_q. By a projector $\mathbb{P}$ we mean an idempotent ($\mathbb{P} \circ \mathbb{P} = \mathbb{P}$) linear operator from the linear space $\mathcal{L} = C(K_q)$ onto a closed subspace $\tilde{\mathcal{L}} \subset \mathcal{L}$. For example, let $\tilde{\mathcal{L}}$ be a space of continuous functions in K_q, such that their ξ_1-derivative exists in K_q and is constant. The appropriate projector $\mathbb{P} : \mathcal{L} \to \tilde{\mathcal{L}}$ is then defined as

$$\mathbb{P}(f) = \frac{1-\xi_1}{2} f(-1, \xi_2) + \frac{\xi_1 + 1}{2} f(1, \xi_2). \tag{3.9}$$

Intuitively speaking, the projector $\mathbb{P}$ keeps the function unchanged along the vertical edges e_1 and e_2 of the reference domain K_q, and interpolates its values between $\xi_1 = -1$ and $\xi_1 = 1$ linearly along each horizontal line $\xi_2 = \text{const}$ (see Figure 3.5). The reader is right when she/he recognizes the Lobatto shape functions l_0 and l_1 in (3.9).

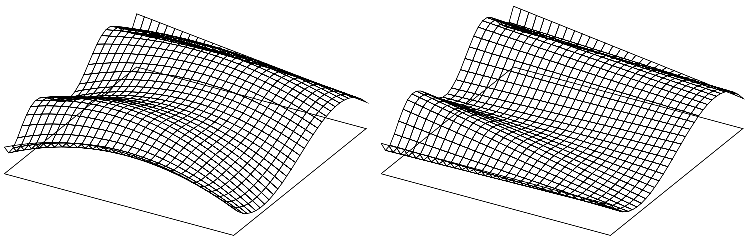

FIGURE 3.5: Example function f in K_q (left) and its projection $\mathbb{P}(f)$ using the projector (3.9) (right).

It is easy to see that $\mathbb{P}$ is both linear

$$\mathbb{P}(f+g) = \mathbb{P}(f) + \mathbb{P}(g)$$

and idempotent

$$(\mathbb{P}\circ\mathbb{P})(f) = \frac{1-\xi_1}{2}\mathbb{P}(f)(-1,\xi_2) + \frac{\xi_1+1}{2}\mathbb{P}(f)(1,\xi_2)$$

$$= \frac{1-\xi_1}{2}f(-1,\xi_2) + \frac{\xi_1+1}{2}f(1,\xi_2) = \mathbb{P}(f).$$

Projector (3.9) can easily be generalized to interpolate the function f exactly along $m+1$ vertical lines $\xi_1 = s_i$, $-1 = s_0 < s_1 \ldots < s_m = 1$:

$$\mathbb{P}_v(f) = \sum_{i=0}^{m} f(s_i,\xi_2)\theta_i^v(\xi_1), \tag{3.10}$$

where

$$\theta_i^v(\xi_1) = \frac{\prod_{j\neq i}(\xi_1 - s_j)}{\prod_{j\neq i}(s_i - s_j)},$$

$i = 0, 1, \ldots, m$, are the fundamental (cardinal) functions for *Lagrange polynomial interpolation* [58]. In the context of transfinite interpolation, the functions θ_i^v are called the *blending functions* [97]. Since for both operators (3.9) and (3.10) $\mathbb{P}(f)$ and $\mathbb{P}_v(f)$ coincides with f at a nondenumerable number of points, they are simple examples of transfinite interpolation schemes.

3.2.2 Bipolynomial Lagrange interpolation

Probably the best-known class of formulae for bivariate interpolation-approximation are the (tensor product) *bipolynomial Lagrange interpolation formulae.* Consider a formula analogous to (3.10) for horizontal lines $\xi_2 = t_j$, $-1 = t_0 < t_1 \ldots < t_n = 1$:

$$\mathbb{P}_h(f) = \sum_{j=0}^{n} f(\xi_1, t_j)\theta_j^h(\xi_2), \tag{3.11}$$

where

$$\theta_j^h(\xi_2) = \frac{\prod_{j\neq i}(\xi_2 - t_i)}{\prod_{j\neq i}(t_j - t_i)},$$

$j = 0, 1, \ldots, n$. This class of formulae is obtained as a product of the above projectors $\mathbb{P}_v$ and $\mathbb{P}_h$:

$$(\mathbb{P}_h \circ \mathbb{P}_v)(f) = \sum_{i=0}^{m}\sum_{j=0}^{n} f(s_i, t_j)\theta_i^v(\xi_1)\theta_j^h(\xi_2). \tag{3.12}$$

The product operator $\mathbb{P}_v \circ \mathbb{P}_h$ is itself a projector, and $(\mathbb{P}_v \circ \mathbb{P}_h)(f)$ interpolates the function f exactly at $(m+1)(n+1)$ points (s_i, t_j), $i = 0, 1, \ldots, m$, $j = 0, 1, \ldots, n$. As the precision set of the operator $\mathbb{P}_v \circ \mathbb{P}_h$ consists of these $(m+1)(n+1)$ points only, this is not a transfinite interpolation.

3.2.3 Transfinite bivariate Lagrange interpolation

There is, however, a second and stronger way to compound the projectors $\mathbb{P}_v$ and $\mathbb{P}_h$, resulting in a transfinite interpolation operator the precision set of which contains the *whole lines* $\xi_1 = s_i$, $\xi_2 = t_j$, $i = 0, 1, \ldots, m$, $j = 0, 1, \ldots, n$. The Boolean sum [94]

$$\mathbb{P}_v \oplus \mathbb{P}_h = \mathbb{P}_v + \mathbb{P}_h - \mathbb{P}_v \circ \mathbb{P}_h \tag{3.13}$$

serves as a basis for the following result [99]:

THEOREM 3.8
Let the operators $\mathbb{P}_v$ and $\mathbb{P}_h$ be defined as above. Then $(\mathbb{P}_v \oplus \mathbb{P}_h)(f)$ interpolates the function f exactly along the lines $\xi_1 = s_i$, $\xi_2 = t_j$, $i = 0, 1, \ldots, m$, $j = 0, 1, \ldots, n$.

PROOF Use the expressions (3.10), (3.11) and (3.13) to verify that

$$(\mathbb{P}_v \oplus \mathbb{P}_h)(f)(s_i, \xi_2) = f(s_i, \xi_2), \quad 0 \le i \le m,$$

$$(\mathbb{P}_v \oplus \mathbb{P}_h)(f)(\xi_1, t_j) = f(\xi_1, t_j), \quad 0 \le j \le n.$$

The result follows immediately. $\square$

Various extensions and generalizations of Theorem 3.8 are immediate. For example, the theorem remains valid if the projector $\mathbb{P}_v$ is taken to be the *cubic spline interpolation projector* in the variable ξ_1 and $\mathbb{P}_h$ is taken to be a *trigonometric polynomial interpolation projector* [94]. All that is really essential is that the functions $\theta_i^v(\xi_1)$ and $\theta_j^h(\xi_2)$ satisfy the cardinality conditions

$$\theta_i^v(s_k) = \delta_{ik}, \quad 0 \le i, k \le m,$$

$$\theta_j^h(t_k) = \delta_{jk}, \quad 0 \le j, k \le n.$$

After presenting the basic ideas of the transfinite interpolation, let us now turn our attention to its application to the finite element technology.

3.3 Construction of reference maps

Now we will define suitable parametrizations for edges and faces of (generally arbitrarily curved) elements $K \in \mathcal{T}_{h,p}$, and apply the transfinite interpolation technique from the previous section in order to design reference maps $\boldsymbol{X}_K(\boldsymbol{\xi}) : \hat{K} \to K$ (where $\hat{K}$ is an appropriate reference domain). For this purpose we will extend the projectors $\mathbb{P}_v$ and $\mathbb{P}_h$ from the previous section naturally to vector-valued functions.

Moreover, as we will see in Example 3.2, the reference maps $\boldsymbol{X}_K(\boldsymbol{\xi})$ are nonpolynomial (when the edges or faces are parametrized by nonpolynomial functions). As long as they are smooth and one-to-one, this is not a problem and in principle one can use them in the finite element code anyway. However, usually one prefers to construct their *isoparametric* approximations $\boldsymbol{x}_K(\boldsymbol{\xi}) \approx \boldsymbol{X}_K(\boldsymbol{\xi})$, which are polynomial maps defined in terms of master element shape functions. Isoparametric maps can be easily stored and handled in the computer code. This will be the next logical step in Paragraph 3.3.6.

3.3.1 Mapping (curved) quad elements onto K_q

Consider a quadrilateral $K \in \mathcal{T}_{h,p}$, with its edges $\tilde{e}_1, \ldots, \tilde{e}_4$ parametrized by continuous curves $\boldsymbol{X}^{e_1}(\zeta), \ldots, \boldsymbol{X}^{e_4}(\zeta) \subset \mathbb{R}^2$, $\zeta \in [-1, 1]$. We also will discuss situations when such parametrizations are not available.

In order to fit into the transfinite interpolation context from Section 3.2, we define a vector-valued function $\boldsymbol{X}_K$ on the boundary of the reference domain K_q such that

$$\begin{aligned}\boldsymbol{X}_K(-1,\xi_2) &\equiv \boldsymbol{X}^{e_1}(\xi_2),\\ \boldsymbol{X}_K(1,\xi_2) &\equiv \boldsymbol{X}^{e_2}(\xi_2),\\ \boldsymbol{X}_K(\xi_1,-1) &\equiv \boldsymbol{X}^{e_3}(\xi_1),\\ \boldsymbol{X}_K(\xi_1,1) &\equiv \boldsymbol{X}^{e_4}(\xi_1).\end{aligned} \tag{3.14}$$

It is essential to preserve the continuity of ∂K_q and the orientation of edges $e_1,\dots,e_4 \subset \partial K_q$ (as illustrated in Figure 2.1). In other words, the function $\boldsymbol{X}_K$ must satisfy

$$\boldsymbol{X}_K(-1,-1) = \boldsymbol{x}_1,\ \boldsymbol{X}_K(1,-1) = \boldsymbol{x}_2,\ \boldsymbol{X}_K(1,1) = \boldsymbol{x}_3,\ \boldsymbol{X}_K(-1,1) = \boldsymbol{x}_4,$$

where $\boldsymbol{x}_1,\dots,\boldsymbol{x}_4$ are vertices of the physical element K obeying the same ordering as vertices $v_1,\dots,v_4$ of the reference domain K_q.

Now the transfinite interpolation comes into the picture. Consider projectors $\mathbb{P}_v$ and $\mathbb{P}_h$ from (3.10), (3.11) with $m = n = 1$, and thus $s_0 = -1, s_1 = 1, t_0 = -1$ and $t_1 = 1$. Assume *any* continuous extension of the function $\boldsymbol{X}_K(\boldsymbol{\xi})$ from the boundary ∂K_q into element interior, for simplicity denoted by the same symbol $\boldsymbol{X}_K(\boldsymbol{\xi})$. Applying the Boolean sum $\mathbb{P}_v \oplus \mathbb{P}_h$ from Theorem 3.8 to $\boldsymbol{X}_K(\boldsymbol{\xi})$ satisfying (3.14) on ∂K_q, we obtain the simplest, but very useful, vector-valued *bilinearly blended* map

$$\begin{aligned}\boldsymbol{X}_K(\boldsymbol{\xi}) &= \begin{pmatrix} X_{K,1}(\boldsymbol{\xi}) \\ X_{K,2}(\boldsymbol{\xi}) \end{pmatrix}\\ &= \frac{1-\xi_1}{2} f(-1,\xi_2) + \frac{\xi_1+1}{2} f(1,\xi_2)\\ &\quad + \frac{1-\xi_2}{2} f(\xi_1,-1) + \frac{\xi_2+1}{2} f(\xi_1,1) - \frac{(1-\xi_1)}{2}\frac{(1-\xi_2)}{2} f(-1,-1)\\ &\quad - \frac{(1-\xi_1)}{2}\frac{(\xi_2+1)}{2} f(-1,1) - \frac{(1-\xi_2)}{2}\frac{(\xi_1+1)}{2} f(1,-1)\\ &\quad - \frac{(\xi_2+1)}{2}\frac{(\xi_1+1)}{2} f(1,1).\end{aligned} \tag{3.15}$$

REMARK 3.7 (A drawback of transfinite interpolation) By construction $\boldsymbol{X}_K(\boldsymbol{\xi})$ maps ∂K_q onto ∂K. If we could establish that also the Jacobian $\det(D\boldsymbol{X}_K/\partial\boldsymbol{\xi})$ is nonzero in K_q, then we could conclude that $\boldsymbol{X}_K$ is bijective. However, a serious drawback of transfinite interpolation schemes is *that this is generally not true.* There may be two or more points in the reference domain which map onto the same point in K. This deficiency was pointed out by Zienkiewicz [209]. We refer to [98] for a heuristic approach to cure this problem, which is guided by geometric intuition and analysis, and accomplished by visual inspection. □

Example 3.1 (When edge parametrizations are not available)
The parametrizations $\boldsymbol{X}^{e_1}, \ldots, \boldsymbol{X}^{e_4} : [-1,1] \to \mathbb{R}^2$ are in the optimal case provided explicitly as a part of the output of a mesh generator. If this is not the case, one has to use other information to define the parametrizations. For example, let us consider a physical mesh edge $e = \boldsymbol{x}_A\boldsymbol{x}_B$. If the edge is straight, we put $\boldsymbol{X}^e(-1) = \boldsymbol{x}_A$, $\boldsymbol{X}^e(1) = \boldsymbol{x}_B$, and the parametrization has a simple form

$$\boldsymbol{X}^e(\zeta) = \frac{1-\zeta}{2}\boldsymbol{x}_A + \frac{1+\zeta}{2}\boldsymbol{x}_B, \quad \zeta \in [-1,1].$$

With additional information about (for example) the midpoint $\boldsymbol{x}_C = \boldsymbol{X}^e(0)$ of the edge e, one can construct a quadratic parametrization of the form

$$\boldsymbol{X}^e(\zeta) = \boldsymbol{a}\zeta^2 + \boldsymbol{b}\zeta + \boldsymbol{c},\ \zeta \in [-1,1],$$

where $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ are vector-valued coefficients, as depicted in Figure 3.6.

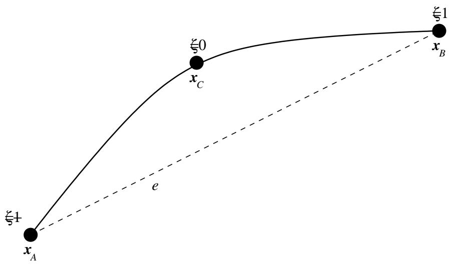

FIGURE 3.6: Construction of a quadratic parametrization based on an additional point $\boldsymbol{x}_C = \boldsymbol{X}^e(0)$.

This leads to two systems of three linear algebraic equations for the first and second component of the unknown coefficients, respectively. The quadratic case has a straightforward generalization to pth-order curves based on $p-1$ additional points. □

Example 3.2
To fix ideas, consider a simple example: let the parametrization of edges of a deformed quadrilateral be given by

$$\begin{aligned}
\boldsymbol{X}_K(-1,\xi_2) \equiv \boldsymbol{X}^{e_1}(\xi_2) &= \left(0, a + b\frac{(\xi_2+1)}{2}\right), \qquad (3.16)\\
\boldsymbol{X}_K(1,\xi_2) \equiv \boldsymbol{X}^{e_2}(\xi_2) &= \left(a + b\frac{(\xi_2+1)}{2}, 0\right),\\
\boldsymbol{X}_K(\xi_1,-1) \equiv \boldsymbol{X}^{e_3}(\xi_1) &= \left(a\cos\frac{(\xi_1+1)\pi}{2}, a\sin\frac{(\xi_1+1)\pi}{2}\right),\\
\boldsymbol{X}_K(\xi_1,1) \equiv \boldsymbol{X}^{e_4}(\xi_1) &= \left((a+b)\cos\frac{(\xi_1+1)\pi}{2}, (a+b)\sin\frac{(\xi_1+1)\pi}{2}\right),
\end{aligned}$$

where $a, b > 0$ are two real parameters, as illustrated in Figure 3.7.

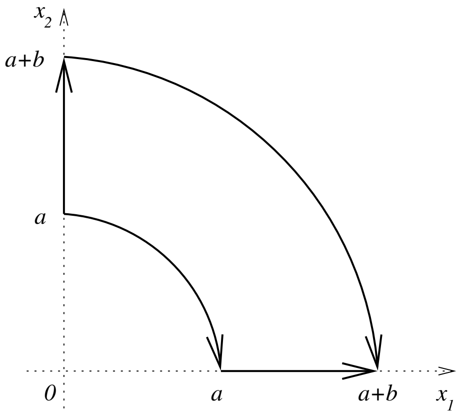

FIGURE 3.7: Sample deformed quadrilateral K.

The bilinearly blended map (3.15) in this very simple case reduces to

$$\boldsymbol{X}_K(\xi_1,\xi_2) = \left(a + b\frac{(\xi_2+1)}{2}\right)\begin{pmatrix}\cos\frac{(\xi_1+1)\pi}{4}\\ \sin\frac{(\xi_1+1)\pi}{4}\end{pmatrix}.$$

It is easy to confirm that $\boldsymbol{X}_K(\boldsymbol{\xi})$ is univalent and that the Jacobian $\det(D\boldsymbol{X}_K/\partial\boldsymbol{\xi})$ is nonzero in K_q. It is easy to see that lines $\xi_1 = \text{const.}$ and $\xi_2 = \text{const.}$ transform to radial lines and circular arcs, joining corresponding points on opposite boundaries of the region K, respectively. $\square$

3.3.2 Mapping (curved) triangular elements onto K_t

Assume that edges of a triangle $K \in \mathcal{T}_{h,p}$ are parametrized by continuous curves $\boldsymbol{X}^{e_j}(\zeta), \zeta \in [-1,1]$, $j = 1, \ldots, 3$, such that

$$\begin{aligned}\boldsymbol{X}^{e_1}(1) = \boldsymbol{X}^{e_2}(-1) = \boldsymbol{x}_2,\\ \boldsymbol{X}^{e_2}(1) = \boldsymbol{X}^{e_3}(-1) = \boldsymbol{x}_3,\\ \boldsymbol{X}^{e_3}(1) = \boldsymbol{X}^{e_1}(-1) = \boldsymbol{x}_1,\end{aligned}$$

where $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_3$ are vertices of the physical element K, ordered counterclockwise in the same way as the vertices $v_1, \ldots, v_3$ of the reference domain K_t (depicted in Figure 2.13).

Transfinite interpolation schemes can be formulated in terms of projectors, which are in this case constructed in the form of triple Boolean sums analogous to (3.13) – see, e.g., [9, 28]. Without going into algebraic details, a simple but very useful transfinite interpolation scheme has the form

$$\boldsymbol{X}_K(\boldsymbol{\xi}) = \boldsymbol{X}_K^v(\boldsymbol{\xi}) + \boldsymbol{X}_K^e(\boldsymbol{\xi}), \quad \boldsymbol{\xi} \in K_t, \tag{3.17}$$

where the affine part

$$\boldsymbol{X}_K^v(\boldsymbol{\xi}) = \sum_{i=1}^{3} \boldsymbol{x}_i \varphi_t^{v_j}(\boldsymbol{\xi})$$

can be expressed by means of the physical mesh vertices $\boldsymbol{x}_i$ and scalar vertex functions (2.20). The higher-order part of the parametrizations $\boldsymbol{X}^{e_j}$ translates into the two-dimensional map by virtue of the second term

$$\boldsymbol{X}_K^e(\boldsymbol{\xi}) = \sum_{j=1}^{3} \boldsymbol{X}_{int}^{e_j}(\lambda_B(\boldsymbol{\xi}) - \lambda_A(\boldsymbol{\xi}))\lambda_A(\boldsymbol{\xi})\lambda_B(\boldsymbol{\xi}), \tag{3.18}$$

which vanishes if all edges $\tilde{e}_j$ of the physical mesh element happen to be straight. Here for each (oriented) reference edge $e_j = v_A v_B$ the affine coordinates λ_A, λ_B are such that $\lambda_A(v_A) = \lambda_B(v_B) = 1$, and the function

$$\boldsymbol{X}_{int}^{e_j}(\zeta) = \frac{\boldsymbol{X}_0^{e_j}(\zeta)}{\left(\frac{1-\zeta}{2}\right)\left(\frac{\zeta+1}{2}\right)}, \quad \zeta \neq \pm 1, \tag{3.19}$$

is defined by eliminating roots ± 1 simultaneously from both vector components of the bubble part $\boldsymbol{X}_0^{e_j}$ of the parametrizations $\boldsymbol{X}^{e_j}$,

$$\boldsymbol{X}_0^{e_j}(\zeta) = \boldsymbol{X}^{e_j}(\zeta) - \boldsymbol{X}^{e_j}(-1)\frac{1-\zeta}{2} - \boldsymbol{X}^{e_j}(1)\frac{\zeta+1}{2}. \tag{3.20}$$

Geometrically, (3.20) corresponds to subtracting the straight part from the (curved) edge $\tilde{e}_j$. Notice that we never need to physically divide by zero, since we define $\boldsymbol{X}_K^e(\boldsymbol{\xi}) = 0$ at vertices in (3.18) instead of using (3.19). Recall that a scalar version of the same trick – elimination of roots ± 1 by division by the product $(1+\zeta)(1-\zeta)/4$ – was used for the construction of master triangle edge functions (2.21).

The reader may find it useful to verify by himself that

$$\boldsymbol{X}_K(\boldsymbol{\xi})|_{e_j} = \boldsymbol{X}^{e_j}(\lambda_B(\boldsymbol{\xi}) - \lambda_A(\boldsymbol{\xi}))|_{e_j}$$

for each edge e_j, $j = 1, \dots, 3$, using the fact that $\zeta = (\lambda_B(\boldsymbol{\xi}) - \lambda_A(\boldsymbol{\xi}))|_{e_j} \in [-1, 1]$ parametrizes the edge e_j.

3.3.3 Mapping (curved) brick elements onto K_B

Let the edges $\tilde{e}_j$, $j = 1, \dots, 12$, of a brick $K \in \mathcal{T}_{h,p}$ be parametrized by continuous curves $\boldsymbol{X}^{e_j}(\zeta) \subset \mathbb{R}^3, \zeta \in [-1, 1]$. As usual the parametrization of edges has to be compatible with the orientation of edges of the reference domain K_B (depicted in Figure 2.26). In other words,

$$\begin{aligned} \boldsymbol{X}^{e_1}(1) = \boldsymbol{X}^{e_2}(-1) &= \boldsymbol{X}^{e_6}(-1) = \boldsymbol{x}_2, \\ \boldsymbol{X}^{e_2}(1) = \boldsymbol{X}^{e_3}(1) &= \boldsymbol{X}^{e_7}(-1) = \boldsymbol{x}_3, \\ \boldsymbol{X}^{e_3}(-1) = \boldsymbol{X}^{e_4}(1) &= \boldsymbol{X}^{e_8}(-1) = \boldsymbol{x}_4, \\ \boldsymbol{X}^{e_4}(-1) = \boldsymbol{X}^{e_1}(-1) &= \boldsymbol{X}^{e_5}(-1) = \boldsymbol{x}_1 \end{aligned} \tag{3.21}$$

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

New in 3D are parametrizations $\boldsymbol{X}^{s_i}(\zeta_1, \zeta_2), \boldsymbol{\zeta} \in [-1, 1]^2$ for the faces $\tilde{s}_i \subset \partial K$, $i = 1, \dots, 6$. Recall that local coordinate axes ζ_1, ζ_2 attached to each face (see Paragraph 2.2.4) are oriented accordingly to the coordinate axes ξ_1, ξ_2, ξ_3, following their lexicographic order. An essential new issue in 3D is the compatibility of face parametrizations $\boldsymbol{X}^{s_i}(\zeta_1, \zeta_2)$ with parametrizations of edges. For example, for the face s_1 this translates into the compatibility conditions

$$\begin{aligned} \boldsymbol{X}^{s_1}(\zeta, -1) &= \boldsymbol{X}^{e_4}(\zeta), \quad \zeta \in [-1, 1], \\ \boldsymbol{X}^{s_1}(\zeta, 1) &= \boldsymbol{X}^{e_{12}}(\zeta), \quad \zeta \in [-1, 1], \\ \boldsymbol{X}^{s_1}(-1, \zeta) &= \boldsymbol{X}^{e_5}(\zeta), \quad \zeta \in [-1, 1], \\ \boldsymbol{X}^{s_1}(1, \zeta) &= \boldsymbol{X}^{e_8}(\zeta), \quad \zeta \in [-1, 1]. \end{aligned} \tag{3.22}$$

Notice that compatibility conditions (3.22) together with conditions (3.21) yield compatibility of parametrizations of faces with vertices:

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

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

$$\boldsymbol{X}_K(\boldsymbol{\xi}) = \boldsymbol{X}_K^v(\boldsymbol{\xi}) + \boldsymbol{X}_K^e(\boldsymbol{\xi}) + \boldsymbol{X}_K^s(\boldsymbol{\xi}), \quad \boldsymbol{\xi} \in K_B. \tag{3.23}$$

The vertex part $\boldsymbol{X}_K^v(\boldsymbol{\xi})$ is defined by combining the physical mesh vertex coordinates $\boldsymbol{x}_i$ and scalar vertex functions (2.28),

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

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

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

is bilinearly blended,

$$\boldsymbol{X}_K^{e_j}(\boldsymbol{\xi}) = \boldsymbol{X}_0^{e_j}(\lambda_B(\boldsymbol{\xi}) - \lambda_A(\boldsymbol{\xi}))\lambda_C(\boldsymbol{\xi})\lambda_D(\boldsymbol{\xi}), \tag{3.24}$$

and used for the definition of the edge part

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

of the transfinite interpolant $\boldsymbol{X}_K(\boldsymbol{\xi})$. For each edge e_j the affine coordinates in (3.24) are chosen so that λ_A, λ_B vanish on faces perpendicular to e_j and are ordered so that $\lambda_A(v_A) = \lambda_B(v_B) = 1$. The affine coordinates λ_C, λ_D vanish on the faces $s_C, s_D \subset \partial K_B$, which do not share any vertex with the edge e_j, respectively.

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

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

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

$$\boldsymbol{X}_K^{s_i}(\boldsymbol{\xi}) = \boldsymbol{X}_0^{s_i}(\lambda_B(\boldsymbol{\xi}) - \lambda_A(\boldsymbol{\xi}), \lambda_D(\boldsymbol{\xi}) - \lambda_C(\boldsymbol{\xi}))\lambda_E(\boldsymbol{\xi}), \tag{3.25}$$

and contribute to the face part

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

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

The affine coordinates in (3.25) are chosen taking into account the local coordinate system on the face s_i: λ_A, λ_B correspond to faces perpendicular to the local axis ζ_1 and $\lambda_A(e_A) = \lambda_B(e_B) = 1$ where the edges e_A, e_B correspond to $\zeta_1 = -1$ and $\zeta_1 = 1$ on the face s_i, respectively. Similarly λ_C, λ_D are chosen for the second local exial direction ζ_2. The affine coordinate λ_E vanishes on the element-opposite face s_E.

3.3.4 Mapping (curved) tetrahedral elements onto K_T

Let the edges $\tilde{e}_j$, $j = 1, \ldots, 6$ of a tetrahedron $K \in \mathcal{T}_{h,p}$ be parametrized by continuous curves $\boldsymbol{X}^{e_j}(\zeta) \subset \mathbb{R}^3, \zeta \in [-1, 1]$, and let the parametrizations be compatible with the orientation of the edges $e_j \subset \partial K_T$ (as depicted in Figure 2.30). This translates into compatibility conditions

$$\begin{aligned} \boldsymbol{X}^{e_1}(1) = \boldsymbol{X}^{e_2}(-1) &= \boldsymbol{X}^{e_5}(-1) = \boldsymbol{x}_2, \\ \boldsymbol{X}^{e_2}(1) = \boldsymbol{X}^{e_3}(-1) &= \boldsymbol{X}^{e_6}(-1) = \boldsymbol{x}_3, \\ \boldsymbol{X}^{e_3}(1) = \boldsymbol{X}^{e_1}(-1) &= \boldsymbol{X}^{e_4}(-1) = \boldsymbol{x}_1, \\ \boldsymbol{X}^{e_4}(1) = \boldsymbol{X}^{e_5}(1) &= \boldsymbol{X}^{e_6}(1) = \boldsymbol{x}_4, \end{aligned} \tag{3.26}$$

where $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_4$ are vertices of the physical element K, ordered compatibly with the vertices $v_1, \ldots, v_4 \subset \partial K_T$.

For simplicity assume that the faces $\tilde{s}_i \subset \partial K_T$, $i = 1, \ldots, 4$, are parametrized by continuous surfaces $\boldsymbol{X}^{s_i}(\boldsymbol{\zeta}), \boldsymbol{\zeta} \in \overline{K_t}$ (i.e., $\zeta_1 \in [-1, 1]$, $\zeta_2 \in [-1, -\zeta_1]$). Recall that each face is assigned a unique orientation (defined in Paragraph 2.2.5), given by a selection of its vertices v_A, v_B, v_C such that v_A has the lowest local index and the vector product $(v_B - v_A) \times (v_C - v_A)$ points outside of K_T. Compatibility of face and edge parametrizations $\boldsymbol{X}^{s_i}(\boldsymbol{\zeta})$ and $\boldsymbol{X}^{e_j}(\zeta)$ is requested in the same way as in the previous case (parametrization of a face, restricted to the boundary of K_t, must match parametrization of the corresponding edge), i.e.,

$$\begin{aligned} \boldsymbol{X}^{s_1}(\zeta, -1) = \boldsymbol{X}^{e_1}(\zeta), &\quad \zeta \in [-1, 1], \\ \boldsymbol{X}^{s_1}(-\zeta, \zeta) = \boldsymbol{X}^{e_5}(\zeta), &\quad \zeta \in [-1, 1], \\ \boldsymbol{X}^{s_1}(-1, \zeta) = \boldsymbol{X}^{e_4}(\zeta), &\quad \zeta \in [-1, 1], \end{aligned} \tag{3.27}$$

for the face s_1 and so on.

The parametrization of the boundary ∂K will be extended into the element interior using again the transfinite interpolation technique, with the transfinite interpolant $\boldsymbol{X}_K(\boldsymbol{\xi})$, $\boldsymbol{\xi} \in K_T$, composed from vertex, edge and face contributions $\boldsymbol{X}_K^v(\boldsymbol{\xi})$, $\boldsymbol{X}_K^e(\boldsymbol{\xi})$ and $\boldsymbol{X}_K^s(\boldsymbol{\xi})$. The vertex part $\boldsymbol{X}_K^v(\boldsymbol{\xi})$,

$$\boldsymbol{X}_K^v(\boldsymbol{\xi}) = \sum_{i=1}^{8} \boldsymbol{x}_i \varphi_T^{v_j}(\boldsymbol{\xi}),$$

is again obtained simply as a combination of coordinates of the physical element vertices $\boldsymbol{x}_i$ and the scalar vertex functions (2.37). The edge part $\boldsymbol{X}_K^e(\boldsymbol{\xi})$

(vanishing if all edges $\tilde{e}_j$ of the physical mesh element are straight) is constructed edgewise as

$$\boldsymbol{X}_K^e(\boldsymbol{\xi}) = \sum_{j=1}^{6} \boldsymbol{X}_{int}^{e_j}(\lambda_B(\boldsymbol{\xi}) - \lambda_A(\boldsymbol{\xi}))\lambda_A(\boldsymbol{\xi})\lambda_B(\boldsymbol{\xi}). \tag{3.28}$$

Here for each (oriented) reference edge $e_j = v_A v_B$ the affine coordinates λ_A, λ_B are such that $\lambda_A(v_A) = \lambda_B(v_B) = 1$, and the function

$$\boldsymbol{X}_{int}^{e_j}(\zeta) = \frac{\boldsymbol{X}_0^{e_j}(\zeta)}{\left(\frac{1-\zeta}{2}\right)\left(\frac{\zeta+1}{2}\right)}, \quad \zeta \neq \pm 1, \tag{3.29}$$

is defined by eliminating roots ± 1 simultaneously from all three vector components of the bubble part $\boldsymbol{X}_0^{e_j}$ of the parametrization of the edge e_j,

$$\boldsymbol{X}_0^{e_j}(\zeta) = \boldsymbol{X}^{e_j}(\zeta) - \boldsymbol{X}^{e_j}(-1)\frac{1-\zeta}{2} - \boldsymbol{X}^{e_j}(1)\frac{\zeta+1}{2}. \tag{3.30}$$

Again, we do not use (3.29) at $\zeta = \pm 1$, since (3.28) is zero at vertices. Next we compute for each face s_i the bubble part $\boldsymbol{X}_0^{s_i}(\boldsymbol{\zeta})$ of its parametrization $\boldsymbol{X}^{s_i}(\boldsymbol{\zeta})$, vanishing on its boundary:

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

In order to vanish on all remaining faces s_A, s_B and s_C, the face contributions $\boldsymbol{X}_K^{s_i}(\boldsymbol{\xi})$ to the transfinite interpolant have to contain the product of affine coordinates λ_A, λ_B and λ_C, corresponding to these faces. Therefore we again first need to divide the function $\boldsymbol{X}_0^{s_i}(\boldsymbol{\zeta})$ by the trace of this product to the face s_i,

$$\boldsymbol{X}_{int}^{s_i}(\boldsymbol{\zeta}) = \frac{\boldsymbol{X}_0^{s_i}(\boldsymbol{\zeta})}{\lambda_A\lambda_B\lambda_C|_{s_i}(\boldsymbol{\zeta})}, \quad \boldsymbol{\zeta} \notin \partial s_i,$$

(this relation is not used on edges, where $\boldsymbol{X}_K^{s_i}(\boldsymbol{\xi}) \equiv 0$) and then multiply it by the same product extended to the whole element interior,

$$\boldsymbol{X}_K^{s_i}(\boldsymbol{\xi}) = \boldsymbol{X}_{int}^{s_i}(\lambda_B(\boldsymbol{\xi}) - \lambda_A(\boldsymbol{\xi}), \lambda_C(\boldsymbol{\xi}) - \lambda_A(\boldsymbol{\xi}))\lambda_A(\boldsymbol{\xi})\lambda_B(\boldsymbol{\xi})\lambda_C(\boldsymbol{\xi}).$$

The face part of the transfinite interpolant $\boldsymbol{X}_K(\boldsymbol{\xi})$ is finally defined by summing up the face contributions,

$$\boldsymbol{X}_K^s(\boldsymbol{\xi}) = \sum_{i=1}^{4} \boldsymbol{X}_K^{s_i}(\boldsymbol{\xi}).$$

The transfinite interpolant $\boldsymbol{X}_K(\boldsymbol{\xi})$ is defined by summing up the vertex, edge and face contributions,

$$\boldsymbol{X}_K(\boldsymbol{\xi}) = \boldsymbol{X}_K^v(\boldsymbol{\xi}) + \boldsymbol{X}_K^e(\boldsymbol{\xi}) + \boldsymbol{X}_K^s(\boldsymbol{\xi}), \quad \boldsymbol{\xi} \in K_T. \tag{3.31}$$

3.3.5 Mapping (curved) prismatic elements onto K_P

Let the edges $\tilde{e}_j$, $j = 1, \dots, 9$, of a prismatic element $K \in \mathcal{T}_{h,p}$ be parametrized by continuous curves $\boldsymbol{X}^{e_j}(\zeta) \subset \mathbb{R}^3, \zeta \in [-1,1]$, whose orientations are compatible with orientations of the edges $e_j \subset K_P$ (depicted in Figure 2.34). Compatibility conditions analogous to (3.26) must be satisfied.

In analogy to the previous cases we assume that the quadrilateral faces $\tilde{s}_i \subset \partial K$, $i = 1, \dots, 3$, are parametrized by continuous surfaces $\boldsymbol{X}^{s_i}(\boldsymbol{\zeta}), \boldsymbol{\zeta} \in \overline{K}_q$, and that the triangular faces s_4, s_5 are parametrized by continuous surfaces $\boldsymbol{X}^{s_i}(\boldsymbol{\zeta}), \boldsymbol{\zeta} \in \overline{K}_t$. Recall the orientation of faces from Paragraph 2.2.6 – each quadrilateral face is assigned a local coordinate system whose horizontal and vertical axes are parallel to its horizontal and vertical edges, respectively. Triangular faces are assigned local orientations in the same way as in the tetrahedral case. Again, compatibility of face and edge parametrizations $\boldsymbol{X}^{s_i}(\boldsymbol{\zeta})$ and $\boldsymbol{X}^{e_j}(\zeta)$ in the usual sense is requested.

The transfinite interpolant $\boldsymbol{X}_K(\boldsymbol{\xi})$, $\boldsymbol{\xi} \in K_P$, will comprise vertex, edge and face contributions $\boldsymbol{X}_K^v(\boldsymbol{\xi})$, $\boldsymbol{X}_K^e(\boldsymbol{\xi})$ and $\boldsymbol{X}_K^s(\boldsymbol{\xi})$. The vertex part $\boldsymbol{X}_K^v(\boldsymbol{\xi})$ is defined as a combination of coordinates of the physical element vertices $\boldsymbol{x}_i$ and the scalar vertex functions (2.45),

$$\boldsymbol{X}_K^v(\boldsymbol{\xi}) = \sum_{i=1}^{6} \boldsymbol{x}_i \varphi_P^{v_j}(\boldsymbol{\xi}). \tag{3.32}$$

The edge part $\boldsymbol{X}_K^e(\boldsymbol{\xi})$ of the transfinite interpolant $\boldsymbol{X}_K(\boldsymbol{\xi})$ is constructed one edge at a time,

$$\boldsymbol{X}_K^e(\boldsymbol{\xi}) = \sum_{j=1}^{9} \boldsymbol{X}_K^{e_j}(\boldsymbol{\xi}).$$

For each edge $e_1, \dots, e_3$ and $e_7, \dots, e_9$ we construct the bubble part $\boldsymbol{X}_0^{e_j}(\zeta)$ of its parametrization by subtracting the trace of the vertex interpolant (3.32) in the standard way, and eliminating the roots ± 1 in the same way as in the tetrahedral case. The result is finally blended using a product of three affine coordinates λ_A, λ_B and λ_C such that the first two of them vanish on quadrilateral faces s_A, s_B, sharing with the edge e_j a single vertex, and λ_C vanishes on the other triangular face. Contributions of edges $e_4, \dots, e_6$ are easier to obtain – one only calculates the bubble part of the parametrizations of these edges, and blends it by a single affine coordinate λ_A that vanishes on the element-opposite face s_A.

The face part $\boldsymbol{X}_K^s(\boldsymbol{\xi})$ of the transfinite interpolant is constructed one face at a time,

$$\boldsymbol{X}_K^s(\boldsymbol{\xi}) = \sum_{i=1}^{5} \boldsymbol{X}_K^{s_i}(\boldsymbol{\xi}).$$

For each quadrilateral and triangular face s_i we need to compute the bubble part of its parametrization $\boldsymbol{X}_0^{s_i}(\boldsymbol{\zeta})$ by subtracting the traces of the vertex and edge interpolants $(\boldsymbol{X}_K^v + \boldsymbol{X}_K^e)|_{s_i}$. Contributions $\boldsymbol{X}_0^{s_i}(\boldsymbol{\zeta})$ of triangular faces s_4, s_5 are blended linearly using the affine coordinate vanishing on the element-opposite triangular face.

The situation for the quadrilateral faces $s_1, \dots, s_3$ is similar to the tetrahedral case. For each of these faces by λ_A, λ_B denote affine coordinates which vanish on the remaining two quadrilateral faces, respectively, and compute the bubble part of their parametrizations $\boldsymbol{X}_0^{s_i}(\boldsymbol{\zeta})$ as usual. On each quadrilateral face s_i we divide the function $\boldsymbol{X}_0^{s_i}(\boldsymbol{\zeta})$ for $\boldsymbol{\zeta} \notin \partial s_i$ by the product $\lambda_A \lambda_B|_{s_i}$, and blend the result bilinearly into the element interior, multiplying it by the same product $\lambda_A(\boldsymbol{\xi})\lambda_B(\boldsymbol{\xi})$ in the whole element interior. The final transfinite interpolant is defined as

$$\boldsymbol{X}_K(\boldsymbol{\xi}) = \boldsymbol{X}_K^v(\boldsymbol{\xi}) + \boldsymbol{X}_K^e(\boldsymbol{\xi}) + \boldsymbol{X}_K^s(\boldsymbol{\xi}), \quad \boldsymbol{\xi} \in K_P.$$

3.3.6 Isoparametric approximation of reference maps

The reference maps $\boldsymbol{X}_K(\boldsymbol{\xi}) : \hat{K} \to K$ are generally nonpolynomial (as illustrated in Example 3.2). In order to facilitate their computer implementation, people usually further approximate them by polynomial *isoparametric* maps $\boldsymbol{x}_K(\boldsymbol{\xi}) \approx \boldsymbol{X}_K(\boldsymbol{\xi})$ that for each element $K \in \mathcal{T}_{h,p}$ are defined as a linear combination of scalar (H^1-conforming) master element shape functions with vector-valued coefficients (*geometrical degrees of freedom*). Isoparametric maps are easy to store and to deal with – in particular their values and partial and inverse derivatives can be calculated efficiently.

The original notion of isoparametric elements, first introduced by Ergatoudis, Irons and Zienkiewicz in [77, 210], is based upon the use of the same set of shape functions for the definition of the reference maps and the approximate solution of the finite element problem. Despite the popularity of this method the reader should be aware of the fact that the mappings and the approximation problem have in general *no relation to each other.*

The right tool for the construction of isoparametric maps is the projection-based interpolation technique on the reference domains (H^1-conforming case), which was introduced in Section 3.1.

Reference quadrilateral K_q

Since the procedure is simple and analogous for all geometrical element types, let us describe it for quadrilateral elements only. Let $K \in \mathcal{T}_{h,p}$ be a quadrilateral element and $\mathcal{K}_q^1$ the corresponding master element with local directional orders of approximation $p^{b,1}, p^{b,2}$ in the element interior and local polynomial orders $p^{e_1}, \dots, p^{e_4}$ on edges. The isoparametric element reference map $\boldsymbol{x}_K(\boldsymbol{\xi}) \approx \boldsymbol{X}_K(\boldsymbol{\xi})$ is sought in the form

$$\boldsymbol{x}_K(\boldsymbol{\xi}) = \sum_{j=1}^{4} \boldsymbol{\alpha}_K^{v_j} \varphi_q^{v_j}(\boldsymbol{\xi}) + \sum_{j=1}^{4} \sum_{k=2}^{p^{e_j}} \boldsymbol{\alpha}_{K,k}^{e_j} \varphi_{k,q}^{e_j}(\boldsymbol{\xi}) + \sum_{n_1=2}^{p^{b,1}} \sum_{n_2=2}^{p^{b,2}} \boldsymbol{\alpha}_{K,n_1,n_2}^{b} \varphi_{n_1,n_2,q}^{b}(\boldsymbol{\xi}) \tag{3.33}$$

(one is free to consider higher polynomial orders than those currently used to approximate the solution). The symbols $\varphi_q^{v_j}, \varphi_{k,q}^{e_j}$ and $\varphi_{n_1,n_2,q}^{b}$ stand for scalar master element vertex, edge and bubble functions (2.13), (2.14) and (2.15) defined in Paragraph 2.2.2.

Since each component of the original map $\boldsymbol{X}_K(\boldsymbol{\xi})$ is nothing other than a continuous scalar function defined on the reference domain K_q, projection-based interpolation is applied to $\boldsymbol{X}_K(\boldsymbol{\xi})$ in the standard way, one component at a time.

REMARK 3.8 (Application of the projection-based interpolation) Recall that the technique works hierarchically: first one computes the geometrical degrees of freedom associated with the vertex functions $\varphi_q^{v_1}, \ldots, \varphi_q^{v_4}$ (those are nothing other than the coordinates of the vertices of the element K – see Paragraph 3.3.7 for details). If one is interested in higher-order maps, in the next step the vertex interpolant is subtracted from the projected function and edge interpolant is constructed. Finally, the design of a higher-order map is accomplished by subtracting both the vertex and edge interpolants from the processed component of $\boldsymbol{X}_K(\boldsymbol{\xi})$ and projecting the residual in the H^1-norm at the polynomial space generated by the master element bubble functions. $\square$

Three-dimensional isoparametric maps are constructed in the same way.

3.3.7 Simplest case – lowest-order reference maps

Due to the hierarchy of H^1-conforming shape functions, the simplest reference map that is based *on vertex functions only* maps all vertices of the mesh element K exactly. In other words, all higher-order shape functions (i.e., edge, face and bubble functions) vanish at the vertices of reference domains and therefore they do not contribute to the values of the map at vertices. Hence, when $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m$ are coordinates of vertices of a mesh element K, it is sufficient to define

$$\boldsymbol{\alpha}_K^{v_j} := \boldsymbol{x}_j, \quad j = 1, 2, \ldots, m \tag{3.34}$$

in (3.33), set the higher-order part of the map to zero, and the simplest working reference map is born. One only has to be careful to enumerate the vertices of K accordingly to the ordering of vertices of the reference domain $\hat{K}$ in order to avoid violation of geometry that would result in a zero Jacobian of the reference map somewhere inside of $\hat{K}$.

The details of this paragraph obviously apply to all types of reference domains K_a, K_q, K_t, K_B, K_T and K_P.

3.3.8 Inversion of reference maps

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

$$\boldsymbol{x}^* = \boldsymbol{x}_K(\boldsymbol{\xi}^*) \in K \in \mathcal{T}_{h,p}. \tag{3.35}$$

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

Affine case

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

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

which yields

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

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

Numerical inversion using the Newton-Raphson technique

The Newton-Raphson technique is a generalization of the standard Newton method to a system of nonlinear algebraic equations. The implicit equation (3.35) for the unknown geometrical point $\boldsymbol{\xi}^* \in \hat{K}$ can be reduced to the standard problem of finding a zero root $\boldsymbol{\xi}^*$, $\boldsymbol{F}_K(\boldsymbol{\xi}^*) = \boldsymbol{0}$, of a nonlinear vector-valued function $\boldsymbol{F}_K : \hat{K} \to \mathbb{R}^d$,

$$\boldsymbol{F}_K(\boldsymbol{\xi}) = \boldsymbol{x}_K(\boldsymbol{\xi}) - \boldsymbol{x}^*,$$

where d is the spatial dimension. The procedure is standard (see Figure 3.8): After choosing an initial guess $\boldsymbol{\xi}_0 \in \hat{K}$, which for higher-order maps can be the solution corresponding to the first-order part of the map only, we iterate

$$\boldsymbol{\xi}_{j+1} = \boldsymbol{\xi}_j - \left(\frac{D\boldsymbol{F}_K}{D\boldsymbol{\xi}}\right)^{-1}(\boldsymbol{\xi}_j)\boldsymbol{F}_K(\boldsymbol{\xi}_j).$$

Similarly as above, the symbol $D\boldsymbol{F}_K/D\boldsymbol{\xi}$ stands for the Jacobi matrix of the function $\boldsymbol{F}_K$, and we assume that the reference map $\boldsymbol{x}_K(\boldsymbol{\xi})$ is constructed in such a way that $D\boldsymbol{F}_K/D\boldsymbol{\xi}$ is not singular in $\hat{K}$. The procedure is repeated until sufficient precision is reached.

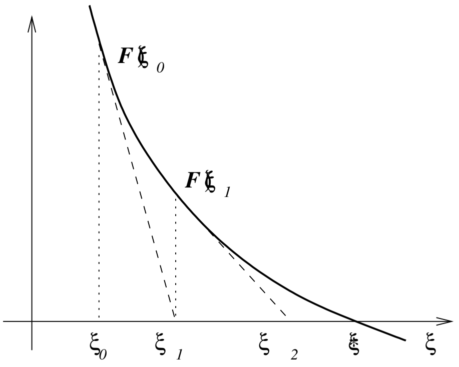

FIGURE 3.8: Schematic picture of the Newton-Raphson technique.

3.4 Projection-based interpolation on physical mesh elements

With the reference maps $\boldsymbol{x}_K$ in hand, we can return to the projection-based interpolation operators Π^1, Π^{curl} and Π^{div}, defined on master elements in Paragraphs 3.1.1, 3.1.2 and 3.1.3, and extend them to physical mesh elements. Recall that the basic property of projection-based interpolation is *locality*: given a sufficiently regular function u in the physical mesh, the interpolant Πu must be constructed elementwise, without any information from neighboring elements, and still it must conform to the global finite element space.

In general it is not enough to transform the projected function from the physical mesh element to the appropriate reference domain and apply procedures described in Paragraphs 3.1.1, 3.1.2 and 3.1.3. The only case where this is possible is when the Jacobi matrix $D\boldsymbol{x}_K$ is constant. In all other cases we have to adjust the norms incorporated into the definition of the projection-based interpolation operators on the reference domains in order to yield correct results for the physical mesh elements. For simplicity, let us demonstrate the procedure in the case of H^1-conforming approximations in two spatial dimensions (see, e.g., [60]).

Consider a sufficiently regular function u given in the physical mesh, a triangular mesh element K, for example, and a reference map $\boldsymbol{x}_K$ such that

$$K = \boldsymbol{x}_K(K_t)$$

(the reference triangular domain K_t was introduced in Paragraph 2.2.3). The projection-based interpolant consists, as in (3.2), of a vertex, edge and bubble part

$$u_{h,p} = u^v_{h,p} + u^e_{h,p} + u^b_{h,p} \in V_{h,p},$$

where $V_{h,p} \subset H^1$ is the corresponding global piecewise polynomial finite element space. The vertex interpolant does not require any extra treatment and is defined simply as

$$u^v_{h,p}|_K = \sum_{k=1}^{3} a^{v_k} \varphi_t^{v_k} \circ \boldsymbol{x}_K^{-1}, \tag{3.36}$$

where the coefficients a^{v_k} are chosen such that $u^v_{h,p}$ matches the original function u at vertices v_1, v_2, v_3 of the mesh element K.

We must be more careful with the edge interpolant. First let us introduce a parametrization $\boldsymbol{x}^{e_j}(\zeta)$, $\zeta \in [-1,1]$, for each edge e_j, $j = 1,\dots,3$ of the reference domain K_t, *consistent with the local orientation of the edges.* These are

$$\boldsymbol{x}^{e_1} = (\zeta, -1),\ \boldsymbol{x}^{e_2} = (-\zeta, \zeta),\ \boldsymbol{x}^{e_3} = (-1, -\zeta).$$

The H^1_0 product on a physical mesh edge $e \subset \partial K$ transforms after an appropriate change of variables and application of the chain rule to

$$(u,v)_{H^1_0,e} = \int_e \frac{\mathrm{d}u}{\mathrm{d}s}\frac{\mathrm{d}v}{\mathrm{d}s}\,\mathrm{d}s = \int_{-1}^{1} \frac{\mathrm{d}\hat{u}}{\mathrm{d}s}\frac{\mathrm{d}\hat{v}}{\mathrm{d}s}\left(\frac{\mathrm{d}s}{\mathrm{d}\zeta}\right)^{-1}\mathrm{d}\zeta$$

(recall the role of this weighted H^1_0 product for the approximation of the norm $H^{\frac{1}{2}}_{00}$ from (3.3)), where again

$$\frac{\mathrm{d}s}{\mathrm{d}\zeta} = \sqrt{\sum_{i=1}^{d}\left(\frac{\mathrm{d}x_i}{\mathrm{d}\zeta}\right)^2}.$$

We see that without the additional weight, unless $\mathrm{d}s/\mathrm{d}\zeta$ is constant (the edge is rectilinear), H^1_0 projection done on the reference domain edge would yield results different from the projection on the physical element edge $e \subset \partial K$. The difference becomes even more pronounced in the transformation rule for the H^1_0 product over the whole element:

$$(u,v)_{H^1_0,K} = \int_K \sum_{k=1}^{2} \frac{\partial u}{\partial x_k}\frac{\partial v}{\partial x_k}\,\mathrm{d}\boldsymbol{x} = \int_{K_t} \sum_{i=1}^{2}\sum_{j=1}^{2} g_{ij}(\boldsymbol{\xi})\frac{\partial \hat{u}}{\partial \xi_k}\frac{\partial \hat{v}}{\partial \xi_k}\mathrm{d}\xi_1\mathrm{d}\xi_2,$$

with a new metric g_{ij} given by

$$g_{ij} = \sum_{k=1}^{2} \frac{\partial \xi_i}{\partial x_k}\frac{\partial \xi_j}{\partial x_k}\det(D\boldsymbol{x}_K),$$

where $\det(D\boldsymbol{x}_K)$ is the Jacobian of the reference map $\boldsymbol{x}_K$. In other words, projection-based interpolation done on physical mesh elements is still performed on the appropriate reference domains, but with different edge and element metrics $\mathrm{d}\zeta/\mathrm{d}s$, g_{ij}.

Both edge and bubble interpolants in the physical mesh are obtained as a composition of the master element interpolants and the inverse reference map in the same way as in (3.36).

3.5 Technology of discretization in two and three dimensions

So far we have designed hierarchic shape functions on master elements and exploited the projection-based and transfinite interpolation techniques to construct polynomial reference maps. What remains to be done in this section is to turn the approximate variational formulation (1.22) into a system of algebraic equations (or ordinary differential equations in the case of time-dependent problems, as mentioned in Paragraph 1.1.7).

REMARK 3.9 To avoid confusion with basis functions that generate polynomial spaces on the reference domains (master element shape functions), we use *global* basis functions to mean the basis functions of the space $V_{h,p}$. ▯

The reader can identify many aspects of the one-dimensional methodology from Section 1.3 in the following outline:

3.5.1 Outline of the procedure

1. We begin with a step that was not present in the 1D scheme. Recall that in Chapter 2 it was necessary to equip the edges and faces of the reference domains with unique orientations in order to make the definitions of master element edge and face functions unique. In the same way one has to assign unique (global) orientations to edges and faces in the mesh $\mathcal{T}_{h,p}$ in order to ensure the uniqueness of basis functions of the space $V_{h,p}$. Let $\boldsymbol{x}_K$ be a smooth bijective reference map corresponding to an element $K \in \mathcal{T}_{h,p}$, and let $\hat{K}$ be the appropriate reference domain. Since the reference and global orientations have been chosen independently, indeed the orientations of the edges and faces of the (geometrically identical) domains K and $\boldsymbol{x}_K(\hat{K})$ *are generally mismatched.*

   This problem has to be resolved in order to ensure *global conformity* of edge and face basis functions. For each element $K \in \mathcal{T}_{h,p}$ one has to adjust the basis of the master element polynomial space in an algorithmically simple way, such that that *the space itself stays unchanged.* We will discuss the procedure in Paragraph 3.5.2.

2. Master elements were designed in Chapter 2 in such a way that they are compatible with the De Rham diagram on the reference domains. It is essential for good performance of finite elements schemes that the finite elements conserve this compatibility on the physical mesh level as well. Therefore the master element polynomial spaces need to be transformed into the physical mesh in a sophisticated way, by means

of transforms that are different for H^1-, $\boldsymbol{H}(\mathrm{curl})$-, $\boldsymbol{H}(\mathrm{div})$- and L^2-conforming elements. We will derive them in Paragraph 3.5.3.

3. With these transforms in hand, global basis functions will be built by "gluing together" various constallations of images of the orientation-adjusted master element shape functions in Paragraph 3.5.4.

4. In Paragraph 3.5.5 we will present *minimum rules* that uniquely identify local polynomial orders for all edges and faces in the finite element mesh, based on the distribution of the polynomial order in element interiors. In other words, at this point the total number of unknowns in the discrete problem will be known.

5. In Paragraph 3.5.6 *connectivity information* will be established in the same way as in Section 1.3. Links from master element shape functions to the appropriate global basis functions will be constructed, which allows the assembling algorithm to access the correct entries in the global stiffness matrix and in the global load vector from the reference domain.

6. Transformation of the variational formulation to the reference domain is a simple operation since one can exploit relations from Paragraph 3.5.3. We demonstrate this briefly on a model equation in Paragraph 3.5.7.

7. In Paragraph 3.5.8 the assembling algorithm will be presented. We proceed in one local and one global step. First one constructs local element stiffness matrices and load vectors by means of orientation-adjusted master element shape functions. Here the situation is more complicated that in 1D – the first significant difference is that already the lowest-order reference maps on some element types *are not affine*, and thus one generally cannot take full advantage of precomputed master element stiffness integrals. In the global step one exploits the connectivity information in order to distribute entries of the local matrices and vectors to appropriate positions in the global discrete system.

3.5.2 Orientation of master element edge and face functions

Let $K \in \mathcal{T}_{h,p}$ be a mesh element, $\hat{K}$ the appropriate reference domain and $\boldsymbol{x}_K : \hat{K} \to K$ a smooth bijective reference map. At the beginning of Paragraph 3.5.1 we explained that the orientation of the master element edge and face functions has to be adjusted in order to compensate for the difference between the unique orientation of edges and faces of the element K, and the unique orientation of edges and faces of the transformed reference domain $\boldsymbol{x}_K(\hat{K})$.

Local orientation of edges and faces of reference domains: Local orientation of edges for the reference domains K_q, K_t, K_B, K_T and K_P can be found in

Figures 2.1, 2.13, 2.26, 2.30 and 2.34, respectively. Local orientations of faces of 3D reference domains, which were also defined in Chapter 2, are depicted in Figure 3.9.

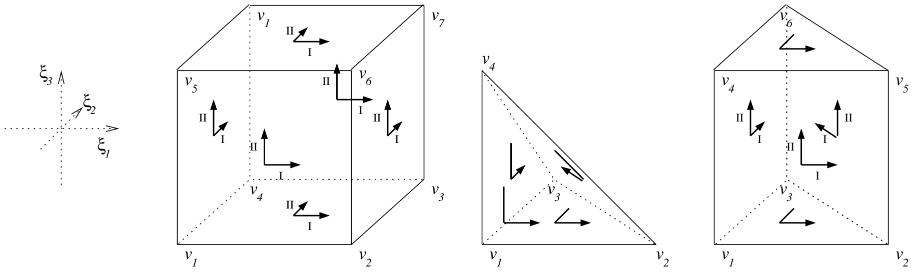

FIGURE 3.9: Local orientation of faces of the reference domains K_B, K_T and K_P.

Global orientation of edges and faces in the mesh: Each edge and face in the initial mesh needs to be assigned a unique orientation. These orientations do not have to be stored explicitly as they can easily be retrieved from a unique global enumeration of vertices. In adaptive algorithms, refined nodes may inherit the orientation of their parents (more details will be given in Paragraph 3.7.1).

Edges are oriented according to the enumeration of their vertices (for example) in increasing order, as illustrated in Figure 3.10.

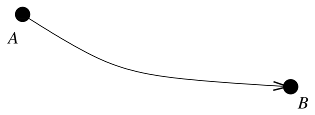

FIGURE 3.10: Global orientation of mesh edges based on a unique enumeration of vertices (here $index(A) < index(B)$).

For each quadrilateral face s we select its vertex A with the lowest index and two edges AB and AC such that $index(A) < index(B) < index(C)$. Triangular faces are oriented in the same way, except that they do not have the product form of the quadrilateral ones, as illustrated in Figure 3.11.

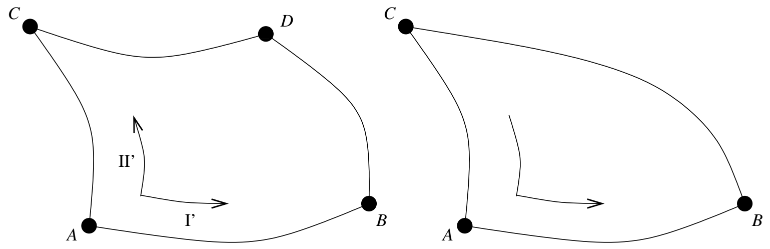

FIGURE 3.11: Global orientation of quadrilateral and triangular faces, $index(A) < index(B) < index(C)$; the vertex A has the lowest index among all vertices of the face.

Transformation of master element edge functions in 2D: Consider an oriented edge $e' = v_i'v_j' \subset \partial K$, $K \in \mathcal{T}_{h,p}$, reference domain $\hat{K}$ such that $K = \boldsymbol{x}_K(\hat{K})$ and oriented edge $e = v_iv_j \subset \partial\hat{K}$ such that $\boldsymbol{x}_K(e) = e'$ (up to the orientation). The edge e needs to be equipped with an orientation flag $o(e) = \pm 1$ that indicates whether its image $\boldsymbol{x}_K(e)$ has the same or opposite orientation with respect to the edge e'. In other words,

$$o(e) = \begin{cases} 1 \text{ if } \ \boldsymbol{x}_K(v_i) = v_i', \ \boldsymbol{x}_K(v_j) = v_j', \\ \\ -1 \text{ if } \ \boldsymbol{x}_K(v_i) = v_j', \ \boldsymbol{x}_K(v_j) = v_i'. \end{cases} \tag{3.37}$$

REMARK 3.10 (Storing edge orientation flags) Let us remark that the edge orientation flags must be stored elementwise for all edges, for adaptive algorithms at least on the coarse mesh level. One may come up with various compression formats – for instance, up to eight edges can be stored in a single 1B variable (type `char` in C++) etc. $\square$

If $o(e) = 1$ then all master element edge functions associated with e stay unchanged. Otherwise we have to transform them using a suitable bijective map $\boldsymbol{x}_e^o : \hat{K} \to \hat{K}$, which inverts the parametrization of the edge e. This is done very easily as follows: One may start with the reference triangle K_t and its identical mapping $\boldsymbol{I}_t : K_t \to K_t$, which can be expressed as a linear combination of vertex shape functions $\varphi_t^{v_k}$ and vertex coordinates v_k,

$$\boldsymbol{I}_t(\xi_1, \xi_2) = \sum_{k=1}^{3} \varphi_t^{v_k}(\xi_1, \xi_2) v_k. \tag{3.38}$$

We arrive at $\boldsymbol{x}_e^o$ by switching in (3.38) the vertex functions corresponding to the vertices v_i, v_j of the edge e. For example, the map $\boldsymbol{x}_{e_1}^o$, after exchanging $\varphi_t^{v_1}$ and $\varphi_t^{v_2}$ in (3.38), has the form

$$\boldsymbol{x}^o_{e_1}(\xi_1,\xi_2) = \varphi^{v_2}_t(\xi_1,\xi_2)v_1 + \varphi^{v_1}_t(\xi_1,\xi_2)v_2 + \varphi^{v_3}_t(\xi_1,\xi_2)v_3.$$

The definition of the vertex functions (2.20),

$$\varphi^{v_1}_t(\xi_1,\xi_2) = \lambda_{2,t}(\xi_1,\xi_2),\ \varphi^{v_2}_t(\xi_1,\xi_2) = \lambda_{3,t}(\xi_1,\xi_2),\ \varphi^{v_3}_t(\xi_1,\xi_2) = \lambda_{1,t}(\xi_1,\xi_2),$$

together with the definition of the scalar edge functions (2.21),

$$\begin{aligned}
\varphi^{e_1}_{k,t} &= \lambda_{2,t}\lambda_{3,t}\phi_{k-2}(\lambda_{3,t} - \lambda_{2,t}), \quad 2 \le k \le p^{e_1},\\
\varphi^{e_2}_{k,t} &= \lambda_{3,t}\lambda_{1,t}\phi_{k-2}(\lambda_{1,t} - \lambda_{3,t}), \quad 2 \le k \le p^{e_2},\\
\varphi^{e_3}_{k,t} &= \lambda_{1,t}\lambda_{2,t}\phi_{k-2}(\lambda_{2,t} - \lambda_{1,t}), \quad 2 \le k \le p^{e_3},
\end{aligned}$$

yield that the exchange of the two vertex functions *is equivalent to the change of sign of the argument of the kernel functions* ϕ_{k-2}.

REMARK 3.11 (General nature of kernel functions) Kernel functions analogous to (1.52) can be constructed for any set of orthogonal polynomials, and they can be defined in many other ways. They do not even have to be symmetric or antisymmetric with respect to the midpoint of the interval where they are defined. However, many aspects simplify when the kernel functions satisfy the condition (3.39). $\square$

If the kernel functions satisfy the condition

$$\phi_{k-2}(-y) = (-1)^k\phi_{k-2}(y), \quad k = 2, 3, \ldots, \tag{3.39}$$

then the adjustment of orientation of scalar edge functions reduces to a change of sign of odd-order functions,

$$\varphi^e_{k,t} \circ \boldsymbol{x}^o_e = \begin{cases} \varphi^e_{k,t},\ k = 2, 3, \ldots, \text{ if } o(e) = 1, \\ (-1)^k\varphi^e_{k,t},\ k = 2, 3, \ldots, \text{ if } o(e) = -1. \end{cases} \tag{3.40}$$

For the particular choice of Lobatto shape functions (1.52) the condition (3.39) is equivalent to

$$l_k(-y) = (-1)^k l_k(y), \quad k = 2, 3, \ldots, \tag{3.41}$$

and therefore (3.39) is satisfied. Using $o(e)$ directly as a sign factor, (3.40) can be written simply as

$$\varphi^e_{k,t} \circ \boldsymbol{x}^o_e = o^k(e)\varphi^e_{k,t}, k = 2, 3, \ldots \tag{3.42}$$

REMARK 3.12 ($\boldsymbol{H}$(curl)- and $\boldsymbol{H}$(div)-conforming approximations) The trace of the tangential component of the $\boldsymbol{H}$(curl)-conforming edge functions (2.57), (2.58), (2.59) as well as the trace of the normal component of the $\boldsymbol{H}$(div)-conforming edge functions (2.99), (2.100) and (2.101) coincide with the Legendre polynomials. Therefore the condition

$$L_k(-y) = (-1)^k L_k(y), \quad k = 0, 1, \dots \tag{3.43}$$

holds. $\square$

We proceed in the same way for all the remaining reference domains. Concerning the reference quadrilateral K_q, the identity $\boldsymbol{I}_q : K_q \to K_q$ reads

$$\boldsymbol{I}_q(\xi_1, \xi_2) = \sum_{k=1}^{4} \varphi_q^{v_k}(\xi_1, \xi_2) v_k. \tag{3.44}$$

When inverting the orientation of the edge $e = v_i v_j$, obviously one has to switch in (3.44) not only the vertex functions associated with the vertices v_i and v_j, but also the other two vertices associated with the element-opposite edge $\tilde{e}$, in order to obtain a bijective map. For the edges e_1 and e_2 this means that $\lambda_{3,q}$ becomes $\lambda_{4,q}$ and vice versa, and analogously $\lambda_{1,q}$ is switched with $\lambda_{2,q}$ if $e = e_3$ or $e = e_4$. It follows from the definition of edge functions (2.14) that under the condition (3.41) we arrive at the same simplification as above,

$$\varphi_{k,q}^e \circ \boldsymbol{x}_e^o = o^k(e) \varphi_{k,q}^e, k = 2, 3, \dots \tag{3.45}$$

Thus once again, if $o(e) = -1$, it is sufficient to multiply all odd-order edge functions associated with the edge e by a sign factor -1, unless one deals with some rare set of shape functions that are not compatible with the conditions (3.41) or (3.43). The situation is illustrated in Figure 3.12.

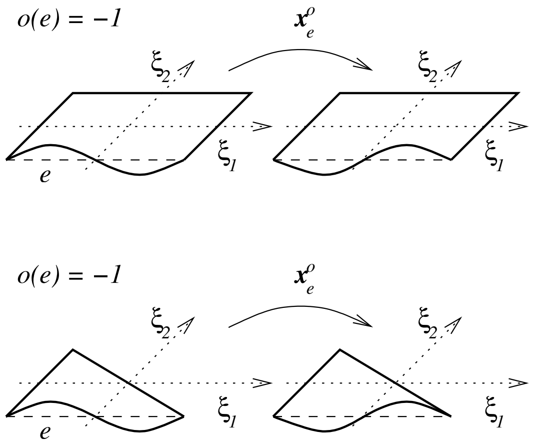

FIGURE 3.12: Local transformation $\boldsymbol{x}_e^o$ of edge functions for the reference quadrilateral K_q and reference triangle K_t if $o(e) = -1$.

Transformation of master element edge functions in 3D is done exactly in the same way as in 2D. For each mesh element K, $\boldsymbol{x}_K(\hat{K}) = K$, each edge $e = v_i v_j \subset \partial\hat{K}$ is equipped with an orientation flag $o(e) = \pm 1$, which is based on an algorithmic check of the condition (3.37). For all standard types of scalar shape functions compatible with (3.41), (3.43) we arrive at a result equivalent to (3.42) and (3.45) also for the reference domains K_B, K_T and K_P. Also in 3D the result naturally extends to the $\boldsymbol{H}$(curl)-conforming edge functions due to the fact that the trace of their tangential component coincides with the Legendre polynomials (Remark 3.12).

Transformation of quadrilateral master element face functions: Consider an oriented quadrilateral mesh face $s' = v_i'v_j'v_k'v_l' \subset \partial K$, $K \in \mathcal{T}_{h,p}$, reference domain $\hat{K}$ such that $K = \boldsymbol{x}_K(\hat{K})$ and oriented face $s = v_i v_j v_k v_l \subset \partial\hat{K}$ such that $\boldsymbol{x}_K(s) = s'$ (up to the orientation). The eight different situations that may occur are illustrated in Figure 3.13.

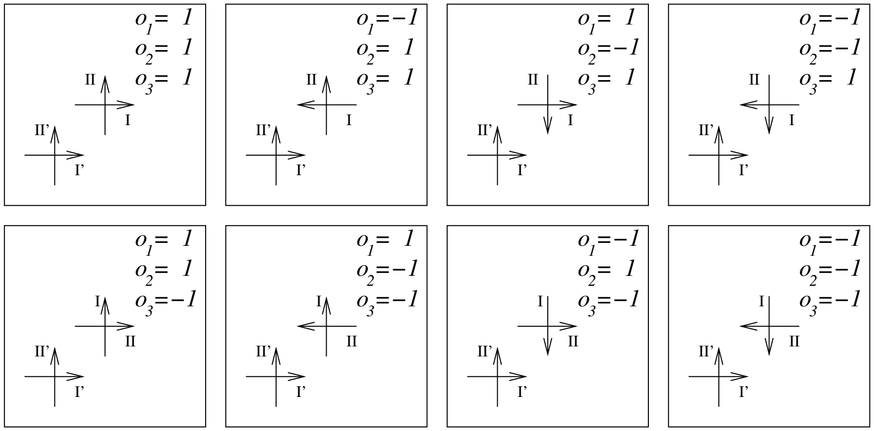

FIGURE 3.13: Eight possible combinations of global directions I$'$, II$'$ of the face s' and the transformed local directions I, II of the face $\boldsymbol{x}_K(s)$.

Hence, the face s needs to be equipped with three orientation flags $o_m = \pm 1, m = 1, \ldots, 3$. The flag $o_3(s)$ indicates whether the transformed local axes I, II match the global axes I$'$ and II$'$ in this order or not. If $o_3(s) = 1$, the flag $o_1(s)$ indicates if the transformed local direction I matches the global direction I$'$, and the flag $o_2(s)$ indicates if the transformed local direction II matches the global direction II$'$. At last, if $o_3(s) = -1$, the flag $o_1(s)$ indicates if the transformed local direction I matches the global direction II$'$, and the flag $o_2(s)$ indicates if the transformed local direction II matches the global direction I$'$.

If $o_1(s) = o_2(s) = o_3(s) = 1$ then all master element face functions associated with the face s stay unchanged. Otherwise, in the same way as with edge functions in 2D, we have to transform the face functions using a suitable bijective map $\boldsymbol{x}_s^o : \hat{K} \to \hat{K}$. Again we start from the identical mapping that, for instance for the reference brick K_B, reads

$$\boldsymbol{I}_B(\xi_1, \xi_2, \xi_3) = \sum_{k=1}^{8} \varphi_B^{v_k}(\xi_1, \xi_2, \xi_3) v_k. \tag{3.46}$$

First assume that $o_3(s) = 1$. If $o_1(s) = -1$, we obtain the transformation $\boldsymbol{x}_s^o$ by switching in (3.46) vertex functions in all four pairs of vertices corresponding to edges lying in the first local direction. The situation for the faces s_1 and s_2 is illustrated in Figure 3.14.

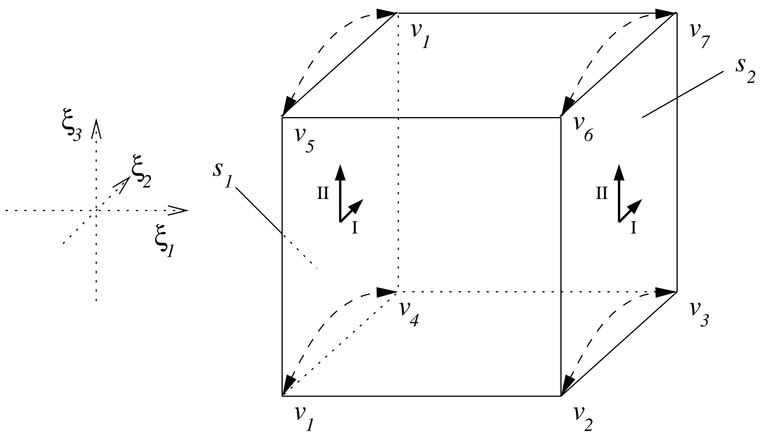

FIGURE 3.14: Construction of the local transformation $\boldsymbol{x}_s^o$ from (3.46) if $o_1(s_1) = -1$ and $o_2(s_1) = o_3(s_1) = 1$.

The same occurs with the flag $o_2(s)$ and the second local direction II.

Returning now to the definition (2.31) of scalar face functions for K_B, we realize that under the condition (3.41) the transformation simplifies to the same ± 1 sign factors as for the edge functions above,

$$\varphi_{n_1,n_2,B}^s \circ \boldsymbol{x}_s^o = o_1^{n_1} o_2^{n_2} \varphi_{n_1,n_2,B}^s, \quad n_1 = 2, 3, \ldots; \; n_2 = 2, 3, \ldots; \text{ if } o_3(s) = 1. \tag{3.47}$$

In the case that $o_3(s) = -1$ the situation is similar except that indices n_1 and n_2 in the definition of the face functions must be switched,

$$\varphi^s_{n_1,n_2,B} \circ \boldsymbol{x}^o_s = o_1^{n_1} o_2^{n_2} \varphi^s_{n_2,n_1,B}, \quad n_1 = 2, 3, \ldots; \; n_2 = 2, 3, \ldots; \text{ if } o_3(s) = -1. \tag{3.48}$$

The same conclusion holds, up to the compatibility with conditions (3.41) and (3.43), also for quadrilateral face functions for $\boldsymbol{H}$(curl)- and $\boldsymbol{H}$(div)-conforming approximations. The reader can now easily figure out the procedure for scalar and vector-valued quadrilateral face functions associated with the prismatic reference domain K_P – also in this case, multiplication by ± 1 sign factors and eventually switching of the indices n_1, n_2 is sufficient.

Transformation of triangular master element face functions: Now the situation becomes more interesting since triangular face functions are not invariant with respect to the enumeration of vertices. Recall that one of the vertices plays a special role in the definition of orientations for both the reference domain and physical mesh faces (for both we selected a vertex with the lowest index).

Consider an oriented triangular mesh face $s' = v'_i v'_j v'_k \subset \partial K$ $(index(v'_i) < index(v'_j) < index(v'_k))$, reference domain $\hat{K}$ such that $K = \boldsymbol{x}_K(\hat{K})$ and oriented face $s = v_A v_B v_C \subset \partial \hat{K}$ $(A < B < C)$ such that $\boldsymbol{x}_K(s) \equiv s'$ (up to the orientation). The six different situations that may occur are illustrated in Figure 3.15.

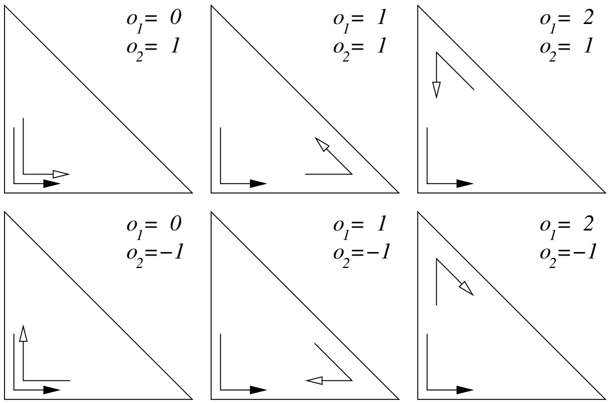

FIGURE 3.15: Six possible combinations of the global orientation of the triangular mesh face s' (black arrow) and the transformed local orientation of the face $\boldsymbol{x}_K(s)$.

It is natural to equip the face s with two orientation flags $o_1(s) \in \{0, 1, 2\}$ and $o_2(s) = \pm 1$ indicating the position of $\boldsymbol{x}_K(v_A)$ with respect to v'_A and compatibility of orientations of the faces s' and s, respectively. If $o_1(s) = 0$ and $o_1(s) = 1$ then all master element face functions associated with the face s stay unchanged. Otherwise, analogously as before, we have to transform them by means of a suitable bijective map $\boldsymbol{x}_s^o : \hat{K} \to \hat{K}$. This time let us use the reference tetrahedron K_T for demonstration purposes. The corresponding identical mapping $\boldsymbol{I}_T : K_T \to K_T$ can be written as

$$\boldsymbol{I}_T(\xi_1, \xi_2, \xi_3) = \sum_{k=1}^{4} \varphi_T^{v_k}(\xi_1, \xi_2, \xi_3) v_k. \tag{3.49}$$

We select three barycentric coordinates λ_A, λ_B and λ_C such that $\lambda_A(v_A) = \lambda_B(v_B) = \lambda_C(v_C) = 1$. If $o_2(s) = 1$ and $o_1(s) = 1$, the map $\boldsymbol{x}_s^o$ cyclically rotates the vertices v_A, v_B and v_C by one in the backward direction, i.e., to v_B, v_C and v_A. This is done by the corresponding rotation of vertex functions associated with these vertices in (3.49). If $o_2(s) = 1$ and $o_1(s) = 1$, the vertices are rotated backward by two, i.e., to v_C, v_A and v_B. Now let $o_2(s) = -1$. If $o_1(s) = 0$, the map $\boldsymbol{x}_s^o$ only switches the vertices v_B and v_C. If $o_1(s) = 1$, the vertices v_B and v_C are switched and moreover the vertices v_A, v_C and v_B are rotated by one backward, i.e., to v_C, v_B and v_A. Finally, if $o_1(s) = 2$, the orientation v_A, v_B, v_C is changed to v_B, v_A, v_C.

Recall the definition of scalar face functions for K_T (2.39),

$$\varphi_{n_1,n_2,T}^{s} = \lambda_A \lambda_B \lambda_C \phi_{n_1-1}(\lambda_B - \lambda_A)\phi_{n_2-1}(\lambda_A - \lambda_C), \tag{3.50}$$

$1 \le n_1, n_2;\ n_1 + n_2 \le p^s - 1$. What remains to be done is to compose these shape face functions with the mapping $\boldsymbol{x}_s^o$. This is done easily by rotating the barycentric coordinates λ_A, λ_B and λ_C in (3.50) in the way described above. For example, for $o_1(s) = 2$ and $o_2(s) = -1$, the new face functions have the form

$$\varphi_{n_1,n_2,T}^{s} \circ \boldsymbol{x}_s^o = \lambda_B \lambda_A \lambda_C \phi_{n_1-1}(\lambda_A - \lambda_B)\phi_{n_2-1}(\lambda_B - \lambda_C).$$

Analogously we proceed for triangular faces of the reference prism K_P and for vector-valued approximations.

3.5.3 Transformation of master element polynomial spaces

Before one can design basis functions of the space $V_{h,p}$, master element polynomial spaces have to be transformed elementwise to the physical mesh $\mathcal{T}_{h,p}$. This operation was simple in 1D since no orientation-related issues were relevant and since the reference maps were by definition *affine* (see Section 1.3). The transformation rule is well known for H^1-conforming approximations (see below); however, in the $\boldsymbol{H}(\mathrm{curl})$- and $\boldsymbol{H}(\mathrm{div})$-conforming case one must be careful to preserve the *commutativity of the De Rham diagram* (2.1), (2.2) and (2.3) between the reference domain and physical mesh level. It turns out that the master element polynomial spaces have to be transformed differently for H^1, $\boldsymbol{H}(\text{curl})$- and $\boldsymbol{H}(\text{div})$-conforming approximations (see, e.g., [57, 71, 179, 160, 67]).

Consider a reference domain $\hat{K}$, physical mesh element $K \in \mathcal{T}_{h,p}$, sufficiently smooth bijective reference map

$$\boldsymbol{x}_K(\boldsymbol{\xi}) : \hat{K} \to K$$

and the four appropriate types of master elements:

1. H^1 master element $\mathcal{K}^1 = (\hat{K}, \hat{W}, \Sigma^1)$,
2. $\boldsymbol{H}(\text{curl})$ master element $\mathcal{K}^{\text{curl}} = (\hat{K}, \hat{\boldsymbol{Q}}, \Sigma^{\text{curl}})$,
3. $\boldsymbol{H}(\text{div})$ master element $\mathcal{K}^{\text{div}} = (\hat{K}, \hat{\boldsymbol{V}}, \Sigma^{\text{div}})$ and
4. L^2 master element $\hat{\mathcal{K}}^{L^2} = (\hat{K}, \hat{X}, \Sigma^{L^2})$.

H^1-conforming elements

The situation is conventional in the H^1-conforming case, where the mapping $\boldsymbol{\Phi}^1_K$ from the master element space $\hat{W}(\hat{K})$ to the corresponding space $W(K)$ on the element K,

$$\begin{tikzcd} \hat{W}(\hat{K}) \arrow[d, "\boldsymbol{\Phi}^1_K"] \\ W(K), \end{tikzcd} \tag{3.51}$$

requires that the function value of the master element shape function $\hat{w}$ at each reference point $\boldsymbol{\xi} \in \hat{K}$ coincides with the value of the transformed shape function w at its image $\boldsymbol{x} = \boldsymbol{x}_K(\boldsymbol{\xi}) \in K$. Hence the transformation rule reads

$$w = \boldsymbol{\Phi}^1_K(\hat{w}) = \hat{w} \circ \boldsymbol{x}_K^{-1}, \tag{3.52}$$

or, in other words,

$$w(\boldsymbol{x}) = \hat{w}(\boldsymbol{\xi}) \text{ where } \boldsymbol{x} = \boldsymbol{x}_K(\boldsymbol{\xi}). \tag{3.53}$$

The polynomial space on the mesh element K has the form

$$W = \boldsymbol{\Phi}^1_K(\hat{W}). \tag{3.54}$$

In the following we will need the reference map to be at least C^2-smooth.

$\boldsymbol{H}$(curl)-conforming elements

The transform $\boldsymbol{\Phi}_K^{\mathrm{curl}}$ of the master element space $\hat{\boldsymbol{Q}}(\hat{K})$ has to be designed in such a way that the $H^1 - \boldsymbol{H}(\mathrm{curl})$ part of the De Rham diagram,

$$\begin{tikzcd}
\hat{W}(\hat{K}) \arrow[r, "\nabla_\xi"] \arrow[d, "\boldsymbol{\Phi}_K^1"] & \hat{\boldsymbol{Q}}(\hat{K}) \arrow[d, "\boldsymbol{\Phi}_K^{\mathrm{curl}}"] \\
W(K) \arrow[r, "\nabla_x"] & \boldsymbol{Q}(K),
\end{tikzcd} \tag{3.55}$$

commutes (the symbol ∇ was described in Remark 2.1). This means that starting with a scalar shape function $\hat{w} \in \hat{W}(\hat{K})$, one arrives at the same vector-valued function $\boldsymbol{E} \in \boldsymbol{Q}(K)$ either way. In other words,

$$\nabla_x(\hat{w} \circ \boldsymbol{x}_K^{-1}) = \boldsymbol{\Phi}_K^1(\nabla_\xi \hat{w}) \tag{3.56}$$

has to hold for all $\hat{w} \in \hat{W}(\hat{K})$.

It is not difficult to find $\boldsymbol{\Phi}_K^{\mathrm{curl}}$: use the chain rule to differentiate

$$\begin{aligned}
\nabla_x(\hat{w} \circ \boldsymbol{x}_K^{-1})(x) &= \left(\frac{D\boldsymbol{x}_K}{D\boldsymbol{\xi}}(\xi|_{\xi=\boldsymbol{x}_K^{-1}(x)})\right)^{-T} \nabla_\xi \hat{w}(\xi|_{\xi=\boldsymbol{x}_K^{-1}(x)}) \\
&= \left[\left(\frac{D\boldsymbol{x}_K}{D\boldsymbol{\xi}}\right)^{-T} \nabla_\xi \hat{w}\right] \circ \boldsymbol{x}_K^{-1}(x).
\end{aligned} \tag{3.57}$$

Hence, the $\boldsymbol{H}$(curl) transformation rule reads

$$\boldsymbol{E} = \boldsymbol{\Phi}_K^{\mathrm{curl}}(\hat{\boldsymbol{E}}) = \left[\left(\frac{D\boldsymbol{x}_K}{D\boldsymbol{\xi}}\right)^{-T} \hat{\boldsymbol{E}}\right] \circ \boldsymbol{x}_K^{-1}, \tag{3.58}$$

and the master element polynomial space $\hat{\boldsymbol{Q}}$ transforms to

$$\boldsymbol{Q} = \boldsymbol{\Phi}_K^{\mathrm{curl}}(\hat{\boldsymbol{Q}}). \tag{3.59}$$

This conclusion holds both in 2D and 3D (see also [71, 57]).

REMARK 3.13 (Transformation of gradients for H^1-conforming approximations) Relation (3.57) has another interesting interpretation: the gradient operator ∇_x in the physical mesh can be written by means of the gradient ∇_ξ on the reference domain as follows,

$$\nabla_x = \left(\frac{D\boldsymbol{x}_K}{D\boldsymbol{\xi}}\right)^{-T} \nabla_\xi. \tag{3.60}$$

We will exploit this fact later for the transformation of variational formulations in the space H^1 to the reference domain. $\square$

The following Remark 3.14 delivers an analogous message for $\boldsymbol{H}(\mathrm{curl})$-conforming approximations:

REMARK 3.14 (Transformation of the curl operator) It has been shown in [71] that the curl operator $\nabla_x \times$ in the physical mesh can be written by means of the curl operator $\nabla_\xi \times$ on the reference domain as follows,

$$\nabla_x \times \boldsymbol{E} = J_K^{-1}(\nabla_\xi \times \hat{\boldsymbol{E}}). \tag{3.61}$$

As usual we use the symbol

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

for the Jacobian of the reference map $\boldsymbol{x}_K$, which is assumed to be always positive. $\square$

$\boldsymbol{H}(\mathrm{div})$-conforming elements

The divergence section of the De Rham diagram is different in 2D and 3D (recall that it relates H^1 with $\boldsymbol{H}(\mathrm{div})$ in 2D and $\boldsymbol{H}(\mathrm{curl})$ with $\boldsymbol{H}(\mathrm{div})$ in 3D). Therefore let us begin with the 2D case. In the same way as above it is necessary that the diagram

$$\begin{tikzcd} \hat{W}(\hat{K}) \arrow[r, "\nabla_\xi \times"] \arrow[d, "\boldsymbol{\Phi}_K^1"] & \hat{\boldsymbol{V}}(\hat{K}) \arrow[d, "\boldsymbol{\Phi}_K^{\mathrm{div}}"] \\ W(K) \arrow[r, "\nabla_x \times"] & \boldsymbol{V}(K), \end{tikzcd} \tag{3.62}$$

commutes, i.e., that

$$\nabla_x \times (\hat{w} \circ \boldsymbol{x}_K^{-1}) = \boldsymbol{\Phi}_K^{\mathrm{curl}}(\nabla_\xi \times \hat{w}) \tag{3.63}$$

holds for all $\hat{w} \in \hat{W}(\hat{K})$. Recall that in 2D $\mathbf{curl}_\xi = \nabla_\xi \times = (-\partial/\partial\xi_2, \partial/\partial\xi_1)$.

Applying the operator $\nabla_\xi \times$ to the expression

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

we obtain

$$\begin{pmatrix} -\dfrac{\partial \hat{w}}{\partial \xi_2} \\ \dfrac{\partial \hat{w}}{\partial \xi_1} \end{pmatrix} = \begin{pmatrix} \dfrac{\partial x_{K,2}}{\partial \xi_2} & -\dfrac{\partial x_{K,1}}{\partial \xi_2} \\ -\dfrac{\partial x_{K,2}}{\partial \xi_1} & \dfrac{\partial x_{K,1}}{\partial \xi_1} \end{pmatrix} \begin{pmatrix} -\dfrac{\partial w}{\partial x_2} \\ \dfrac{\partial w}{\partial x_1} \end{pmatrix}.$$

Standard identity on inverse matrices (also see Remark 3.26) yields that

$$\begin{pmatrix} \frac{\partial x_{K,2}}{\partial \xi_2} & -\frac{\partial x_{K,1}}{\partial \xi_2} \\ -\frac{\partial x_{K,2}}{\partial \xi_1} & \frac{\partial x_{K,1}}{\partial \xi_1} \end{pmatrix}^{-1} = \left(\frac{D\boldsymbol{x}_K}{D\boldsymbol{\xi}}\right)^T J_K^{-1}.$$

Hence the $\boldsymbol{H}(\mathrm{div})$ transformation rule reads

$$\boldsymbol{v} = \boldsymbol{\Phi}_K^{\mathrm{div}}(\hat{\boldsymbol{v}}) = \left[\left(\frac{D\boldsymbol{x}_K}{D\boldsymbol{\xi}}\right)^T J_K^{-1}\,\hat{\boldsymbol{v}}\right] \circ \boldsymbol{x}_K^{-1}, \tag{3.64}$$

and the master element polynomial space $\hat{\boldsymbol{V}}$ transforms to

$$\boldsymbol{V} = \boldsymbol{\Phi}_K^{\mathrm{div}}(\hat{\boldsymbol{V}}). \tag{3.65}$$

REMARK 3.15 Notice that in analogy to the $\boldsymbol{H}(\mathrm{curl})$-conforming case, where the vector-valued shape functions were transformed as gradients, the $\boldsymbol{H}(\mathrm{div})$-conforming vector-valued shape functions transform as *curls.* $\square$

The result (3.64) holds in unchanged form also in 3D, where we look at the $\boldsymbol{H}(\mathrm{curl}) - \boldsymbol{H}(\mathrm{div})$ section of the De Rham diagram,

$$\begin{tikzcd} \hat{\boldsymbol{Q}}(\hat{K}) \arrow[r, "\boldsymbol{\nabla}_\xi \times"] \arrow[d, "\boldsymbol{\Phi}_K^{\mathrm{curl}}"] & \hat{\boldsymbol{V}}(\hat{K}) \arrow[d, "\boldsymbol{\Phi}_K^{\mathrm{div}}"] \\ \boldsymbol{Q}(K) \arrow[r, "\boldsymbol{\nabla}_x \times"] & \boldsymbol{V}(K) \end{tikzcd} \tag{3.66}$$

(also see [71, 57]).

L^2-conforming elements

Finally we come to the $\boldsymbol{H}(\mathrm{div}) - L^2$ part of the De Rham diagram,

$$\begin{tikzcd} \hat{\boldsymbol{V}}(\hat{K}) \arrow[r, "\boldsymbol{\nabla}_\xi \cdot"] \arrow[d, "\boldsymbol{\Phi}_K^{\mathrm{div}}"] & \hat{X}(\hat{K}) \arrow[d, "\boldsymbol{\Phi}_K^{L^2}"] \\ \boldsymbol{V}(K) \arrow[r, "\boldsymbol{\nabla}_x \cdot"] & X(K), \end{tikzcd} \tag{3.67}$$

which dictates that $\boldsymbol{H}(\mathrm{div})$-conforming master element shape functions have to be transformed *as the divergence.* Thus the transformation rule reads

$$\omega = \boldsymbol{\Phi}_K^{L^2}(\hat{\omega}) = [J_K^{-1}\hat{\omega}] \circ \boldsymbol{x}_K^{-1}. \tag{3.68}$$

Standard identity on determinants,

$$\frac{\partial J_K}{\partial \xi_j} = \sum_{i,k} \left(\frac{\mathrm{D}\boldsymbol{x}_K}{\mathrm{D}\boldsymbol{\xi}} \right)^{-1}_{ki} \frac{\partial}{\partial \xi_j} \left(\frac{\mathrm{D}\boldsymbol{x}_K}{\mathrm{D}\boldsymbol{\xi}} \right)_{ik},$$

is used for the derivation of (3.68). The master element polynomial space $\hat{X}(\hat{K})$, transformed to the physical mesh, becomes

$$X = \boldsymbol{\Phi}_K^{L^2}(\hat{X}). \tag{3.69}$$

3.5.4 Design of global basis functions

In Paragraph 3.5.2 the orientation of master element edge and face functions was elementwise adjusted in order to make them compatible with the unique orientation of edges and faces in the physical mesh $\mathcal{T}_{h,p}$. Recall that these adjustments did not alter the master element polynomial spaces. In Paragraph 3.5.3 one has described the transformation of master element polynomial spaces to physical mesh elements by means of reference maps and their derivatives. The basis functions of the space $V_{h,p}$ can easily be defined now.

REMARK 3.16 (Role of global basis functions in 2D and 3D) In Section 1.3 the global basis functions were used to define unique connectivity information arrays for all elements. In 2D and 3D their additional function is to determine the orientation information for all elements. The connectivity and orientation data structures are essential for the assembling algorithm. Once they are defined, one translates the approximate variational formulation (1.22) elementwise from the physical mesh to the reference domain, and discretizes it by means of master element shape functions. In the same way as in 1D, assembling of the resulting discrete system takes place exclusively on the reference domain. $\square$

The conformity requirements of the spaces H^1, $\boldsymbol{H}(\mathrm{curl})$, $\boldsymbol{H}(\mathrm{div})$ and L^2 dictate to split the global basis functions in the same fashion as we did for master element shape functions (Table 3.1).

In the following we will define the way globally conforming basis functions on element patches are constructed using the master element shape functions and orientation adjustments $\boldsymbol{x}_0^e$ and $\boldsymbol{x}_0^s$ defined in Paragraph 3.5.2. Some of the definitions are similar for the spaces H^1, $\boldsymbol{H}(\mathrm{curl})$ and $\boldsymbol{H}(\mathrm{div})$, but we prefer to introduce them separately for the sake of clarity.

DEFINITION 3.1 (Vertex basis function) *Let the space V from the variational formulation be a subspace of $H^1(\Omega)$ and let $V_{h,p}$ be a finite-dimensional*

TABLE 3.1: Hierarchic basis functions in various function spaces.

	H^1	$\boldsymbol{H}(\mathrm{curl})$	$\boldsymbol{H}(\mathrm{div})$	L^2
2D	vertex	edge	edge	bubble
	edge	bubble	bubble	
	bubble			
3D	vertex	edge	face	bubble
	edge	face	bubble	
	face	bubble		
	bubble			

approximation of V on Ω_h. Vertex function $w \in V_{h,p}$, *associated with a grid vertex $\boldsymbol{v}_k$, is a continuous function defined in Ω_h which equals to one at $\boldsymbol{v}_k$ and vanishes outside of a patch S_k formed by all elements that share the vertex $\boldsymbol{v}_k$. Restricted to each element $K_m \in S_k$,*

$$w|_{K_m} = \Phi^1_{K_m}(\hat{w}), \tag{3.70}$$

where $v = \boldsymbol{x}^{-1}_{K_m}(\boldsymbol{v}_k)$ is the corresponding vertex of the reference domain, $\hat{w}$ is a master element vertex function associated with the vertex v and the H^1 transformation $\Phi^1_{K_m}$ was defined in (3.52).

REMARK 3.17 (Vertex functions in hybrid meshes) Recall that linearity of vertex shape functions along the edges of reference domains K_t and K_q allows for the definition of vertex basis functions in hybrid meshes in 2D. Analogously, their linearity on edges together with linearity on triangular faces and bilinearity on quadrilateral faces of the reference domains K_B, K_T and K_P allow us to construct vertex basis functions in hybrid 3D meshes. $\square$

Vertex functions have in 2D the form of the traditional "hat functions" related to grid vertices, as shown in Figure 3.16. The 3D situation is illustrated in Figure 3.17.

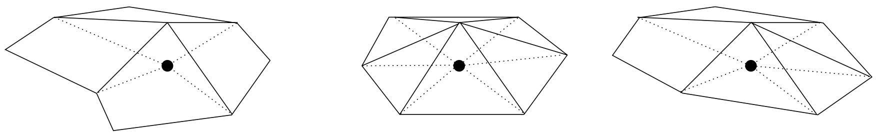

FIGURE 3.16: Vertex basis functions on quadrilateral, triangular and hybrid meshes in 2D.

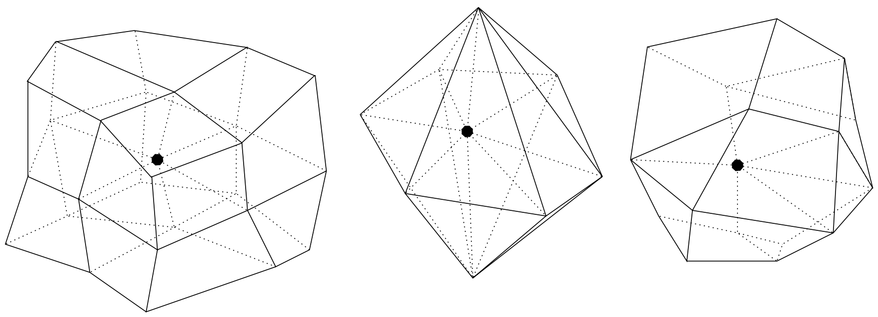

FIGURE 3.17: Element patches for vertex basis functions on hexahedral, tetrahedral and hybrid meshes in 3D (to keep the last figure clear, we consider that the vertex lies on the boundary).

DEFINITION 3.2 (H^1-conforming edge basis function) *Let the space V from the variational formulation be a subspace of $H^1(\Omega)$, and let $V_{h,p}$ be a finite-dimensional approximation of V on Ω_h.* Edge function $w \in V_{h,p}$, *associated with a mesh edge e_k, is a continuous function defined in Ω_h which vanishes outside of a patch S_k formed by all elements that share the edge e_k. Restricted to each element $K_m \in S_k$,*

$$w|_{K_m} = \Phi^1_{K_m}(\hat{w} \circ \boldsymbol{x}^e_0), \tag{3.71}$$

where $e = \boldsymbol{x}^{-1}_{K_m}(e_k)$ is the corresponding edge of the reference domain, $\hat{w}$ is a master element edge function associated with the edge e, $\boldsymbol{x}^e_0$ is the orientation adjustment defined in Paragraph 3.5.2 and the H^1 transformation $\boldsymbol{\Phi}^1_{K_m}$ was defined in (3.52).

DEFINITION 3.3 ($\boldsymbol{H}(\mathrm{curl})$-conforming edge basis function) *Let the space V from the variational formulation be a subspace of $\boldsymbol{H}(\mathrm{curl})(\Omega)$, and let $V_{h,p}$ be a finite-dimensional approximation of V on Ω_h.* Edge function $\boldsymbol{E} \in V_{h,p}$, *associated with a mesh edge e_k, is a vector-valued function defined in Ω_h which vanishes outside of a patch S_k formed by all elements that share the edge e_k. The tangential component of $\boldsymbol{E}$ has to be continuous on all edges of the patch S_k. Restricted to each element $K_m \in S_k$,*

$$\boldsymbol{E}|_{K_m} = \Phi^{\mathrm{curl}}_{K_m}(\hat{\boldsymbol{E}} \circ \boldsymbol{x}^e_0), \tag{3.72}$$

where $e = \boldsymbol{x}^{-1}_{K_m}(e_k)$ is the corresponding edge of the reference domain, $\hat{\boldsymbol{E}}$ is a master element edge function associated with the edge e, $\boldsymbol{x}^e_0$ is the orientation adjustment defined in Paragraph 3.5.2 and the $\boldsymbol{H}(\mathrm{curl})$ transformation $\boldsymbol{\Phi}^{\mathrm{curl}}_{K_m}$ was defined in (3.58).

DEFINITION 3.4 ($\boldsymbol{H}$(div)-conforming edge basis function) *This definition is relevant in 2D only. Let the space V from the variational formulation be a subspace of $\boldsymbol{H}(\mathrm{div})(\Omega)$, and let $V_{h,p}$ be a finite-dimensional approximation of V on Ω_h.* Edge function $\boldsymbol{v} \in V_{h,p}$, *associated with a mesh edge e_k, is a vector-valued function defined in Ω_h which vanishes outside of a patch S_k formed by all elements that share the edge e_k. The normal component of $\boldsymbol{v}$ has to be continuous on all edges of the patch S_k. Restricted to each element $K_m \in S_k$,*

$$\boldsymbol{v}|_{K_m} = \Phi^{\mathrm{div}}_{K_m}(\hat{\boldsymbol{v}} \circ \boldsymbol{x}_0^e), \tag{3.73}$$

where $e = \boldsymbol{x}_{K_m}^{-1}(e_k)$ is the corresponding edge of the reference domain, $\hat{\boldsymbol{v}}$ is a master element edge function associated with the edge e, $\boldsymbol{x}_0^e$ is the orientation adjustment defined in Paragraph 3.5.2 and the $\boldsymbol{H}(\mathrm{div})$ transformation $\boldsymbol{\Phi}^{\mathrm{div}}_{K_m}$ was defined in (3.64).

REMARK 3.18 (Edge functions in hybrid meshes) Recall from Chapter 2 that edge functions on reference domains K_t and K_q in two spatial dimensions as well as edge functions on reference domains K_B, K_T and K_P in 3D for H^1-, $\boldsymbol{H}$(curl)- and $\boldsymbol{H}$(div)-conforming discretizations were designed in such a way that globally conforming edge functions could be constructed in hybrid meshes. $\square$

Examples of element patches related to mesh edges are shown in Figures 3.18 and 3.19.

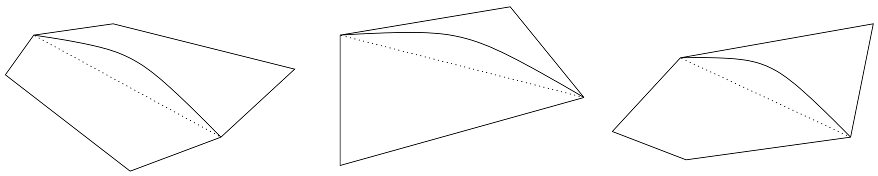

FIGURE 3.18: Edge functions on quadrilateral, triangular and hybrid meshes in 2D: the figure represents either their function values in the H^1-conforming case, or their tangential component (to the dashed edge) in the $\boldsymbol{H}$(curl)-conforming case, or their normal component (to the dashed edge) in the $\boldsymbol{H}$(div)-conforming case.

DEFINITION 3.5 (H^1-conforming face basis function) *Let the space V from the variational formulation be a subspace of $H^1(\Omega)$, $\Omega \subset \mathbb{R}^3$, and let $V_{h,p}$ be a finite-dimensional approximation of V on Ω_h.* Face function

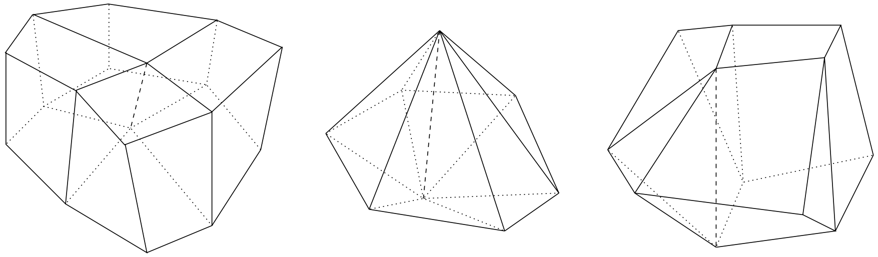

FIGURE 3.19: Element patches for edge functions on hexahedral, tetrahedral and hybrid meshes in 3D.

$w \in V_{h,p}$, associated with a mesh face s_k, is a continuous function defined in Ω_h which vanishes outside of a patch S_k formed by all elements that share the face s_k. Restricted to each element $K_m \in S_k$,

$$w|_{K_m} = \Phi^1_{K_m}(\hat{w} \circ \boldsymbol{x}_0^s), \tag{3.74}$$

where $s = \boldsymbol{x}_{K_m}^{-1}(s_k)$ is the corresponding face of the reference domain, $\hat{w}$ is a master element face function associated with the face s, $\boldsymbol{x}_0^s$ is the orientation adjustment defined in Paragraph 3.5.2 and the H^1 transformation $\boldsymbol{\Phi}^1_{K_m}$ was defined in (3.52).

DEFINITION 3.6 ($\boldsymbol{H}(\mathrm{curl})$-conforming face basis function) *Let the space V from the variational formulation be a subspace of $\boldsymbol{H}(\mathrm{curl})(\Omega)$, $\Omega \subset \mathbb{R}^3$, and let $V_{h,p}$ be a finite-dimensional approximation of V on Ω_h.* Face function *$\boldsymbol{E} \in V_{h,p}$, associated with a mesh face s_k, is a vector-valued function defined in Ω_h which vanishes outside of a patch S_k formed by all elements that share the face s_k. The tangential component of $\boldsymbol{E}$ has to be continuous on all edges and faces of the patch S_k. Restricted to each element $K_m \in S_k$,*

$$\boldsymbol{E}|_{K_m} = \Phi^{\mathrm{curl}}_{K_m}(\hat{\boldsymbol{E}} \circ \boldsymbol{x}_0^s), \tag{3.75}$$

where $s = \boldsymbol{x}_{K_m}^{-1}(s_k)$ is the corresponding edge of the reference domain, $\hat{\boldsymbol{E}}$ is a master element face function associated with the face s, $\boldsymbol{x}_0^s$ is the orientation adjustment defined in Paragraph 3.5.2 and the $\boldsymbol{H}(\mathrm{curl})$ transformation $\boldsymbol{\Phi}^{\mathrm{curl}}_{K_m}$ was defined in (3.58).

DEFINITION 3.7 ($\boldsymbol{H}(\mathrm{div})$-conforming face basis function) *Let the space V from the variational formulation be a subspace of $\boldsymbol{H}(\mathrm{div})(\Omega)$, $\Omega \subset \mathbb{R}^3$, and let $V_{h,p}$ be a finite-dimensional approximation of V on Ω_h.* Face function *$\boldsymbol{v} \in V_{h,p}$, associated with a mesh face s_k, is a vector-valued function defined in Ω_h which vanishes outside of a patch S_k formed by all elements that share* *the face s_k. The normal component of $\boldsymbol{v}$ has to be continuous on all faces of the patch S_k. Restricted to each element $K_m \in S_k$,*

$$\boldsymbol{v}|_{K_m} = \Phi^{\mathrm{div}}_{K_m}(\hat{\boldsymbol{v}} \circ \boldsymbol{x}^s_0), \tag{3.76}$$

where $s = \boldsymbol{x}^{-1}_{K_m}(s_k)$ is the corresponding edge of the reference domain, $\hat{\boldsymbol{v}}$ is a master element face function associated with the face s, $\boldsymbol{x}^s_0$ is the orientation adjustment defined in Paragraph 3.5.2 and the $\boldsymbol{H}(\mathrm{div})$ transformation $\boldsymbol{\Phi}^{\mathrm{div}}_{K_m}$ was defined in (3.64).

REMARK 3.19 (Face functions in hybrid meshes) Recall that (triangular) face functions on reference domains K_T and K_P as well as (quadrilateral) face functions on reference domains K_B and K_P for H^1-, $\boldsymbol{H}(\mathrm{curl})$- and $\boldsymbol{H}(\mathrm{div})$-conforming discretizations were in Chapter 2 designed in such a way that globally conforming face functions could be constructed in hybrid meshes. $\square$

DEFINITION 3.8 (H^1-conforming bubble basis functions) *Let the space V from the variational formulation be a subspace of $H^1(\Omega)$ and let $V_{h,p}$ be a finite-dimensional approximation of V on Ω_h.* Bubble function $w \in V_{h,p}$, *associated with an element K_m, is a continuous function defined in Ω_h that vanishes outside of the element K_m such that*

$$w|_{K_m} = \Phi^1_{K_m}(\hat{w}), \tag{3.77}$$

where $\hat{w}$ is a master element bubble function and the H^1 transformation $\boldsymbol{\Phi}^1_{K_m}$ was defined in (3.52).

DEFINITION 3.9 ($\boldsymbol{H}(\mathrm{curl})$-conforming bubble basis functions) *Let the space V from the variational formulation be a subspace of $\boldsymbol{H}(\mathrm{curl})(\Omega)$ and let $V_{h,p}$ be a finite-dimensional approximation of V on Ω_h.* Bubble function $\boldsymbol{E} \in V_{h,p}$, *associated with an element K_m, is a vector-valued function defined in Ω_h that vanishes outside of the element K_m. The tangential component of $\boldsymbol{E}$ has to vanish on all edges and faces of K_m and*

$$\boldsymbol{E}|_{K_m} = \Phi^{\mathrm{curl}}_{K_m}(\hat{\boldsymbol{E}}), \tag{3.78}$$

where $\hat{\boldsymbol{E}}$ is a master element bubble function and the $\boldsymbol{H}(\mathrm{curl})$ transformation $\boldsymbol{\Phi}^{\mathrm{curl}}_{K_m}$ was defined in (3.58).

DEFINITION 3.10 ($\boldsymbol{H}(\mathrm{div})$-conforming bubble basis functions) *Let the space V from the variational formulation be a subspace of $\boldsymbol{H}(\mathrm{div})(\Omega)$ and let $V_{h,p}$ be a finite-dimensional approximation of V on Ω_h.* Bubble function $\boldsymbol{v} \in V_{h,p}$, *associated with an element K_m, is a vector-valued function* *defined in Ω_h that vanishes outside of the element K_m. The normal component of $\boldsymbol{v}$ has to vanish on all mesh faces, and*

$$\boldsymbol{v}|_{K_m} = \Phi^{\mathrm{div}}_{K_m}(\hat{\boldsymbol{v}}), \tag{3.79}$$

where $\hat{\boldsymbol{v}}$ is a master element bubble function and the $\boldsymbol{H}(\mathrm{div})$ transformation $\boldsymbol{\Phi}^{\mathrm{div}}_{K_m}$ was defined in (3.64).

DEFINITION 3.11 (L^2-conforming bubble basis functions) *Let the space V from the variational formulation be a subspace of $L^2(\Omega)$ and let $V_{h,p}$ be a finite-dimensional approximation of V on Ω_h.* Bubble function $\omega \in V_{h,p}$, *associated with an element K_m, is a scalar function defined in Ω_h that vanishes outside of the element K_m, and*

$$\omega|_{K_m} = \Phi^{L^2}_{K_m}(\hat{\omega}), \tag{3.80}$$

where $\hat{\omega}$ is a master element bubble function and the L^2 transformation $\boldsymbol{\Phi}^{L^2}_{K_m}$ was defined in (3.68).

3.5.5 Minimum rules for higher-order FE discretizations

The minimum rules are specific for hierarchic finite elements and we encountered them already in Chapter 2. Recall from Remark 2.2 that the minimum rule for H^1-conforming approximations says that the (directional) order of approximation along each edge (face) in the finite element mesh is equal to the *minimum* of the (directional) polynomial orders associated with all adjacent elements, as illustrated in Figure 3.20.

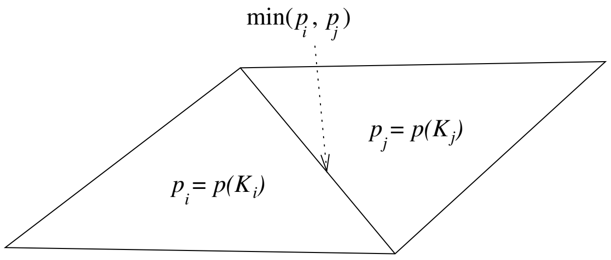

FIGURE 3.20: Minimum rule for two-dimensional approximations.

The function of the minimum rule is to split the global finite element space $V_{h,p}$ from the variational formulation (1.22) into a set of local polynomial spaces of order p_i on all finite elements $K_i \in \mathcal{T}_{h,p}$ – it guarantees that the polynomial order of any shape function associated with the element K_i does not exceed its order of approximation $p_i = p(K_i)$. This is the only way one can speak about the finite element approximation in terms of *function spaces*, independently of their concrete realization. In other words, with the minimum rule one can exchange the shape functions and as long as the polynomial orders associated with all elements stay the same, the approximation $u_{h,p}$ *will not change*. This would not be the case, for example, with a *maximum rule* (that would define the polynomial order for each edge as the maximum of the polynomial orders of all adjacent elements) – with any change of higher-order edge or face functions the approximate solution $u_{h,p}$ would also change.

The minimum rule works analogously in the spaces $\boldsymbol{H}(\mathrm{curl})$ and $\boldsymbol{H}(\mathrm{div})$, adapted to different conformity requirements – in $\boldsymbol{H}(\mathrm{curl})$ it only applies to *tangential components* of edge functions in 2D and to the tangential components of both edge and face functions in 3D. In $\boldsymbol{H}(\mathrm{div})$ it only restricts the *normal component* of edge functions in 2D and the normal component of face functions in 3D. No minimum rule applies to L^2-conforming approximations where no conformity restrictions on element interfaces are imposed.

3.5.6 Enumeration of functions and connectivity arrays

In two and three spatial dimensions the element connectivity arrays play the same role as in the 1D model problem (Section 1.3) – for each element $K \in \mathcal{T}_{h,p}$ they provide each master element shape function $\hat{\theta}$ with the (global) index of the corresponding basis function $\theta \in V_{h,p}$. The connectivity array corresponding to element K has the length $N_{dof,K}$ where $N_{dof,K}$ is the number of master element shape functions (determined by the local orders of approximation on the element K). These element arrays allow the assembling algorithm (Paragraph 3.5.8, Algorithm 3.1) to access appropriate entries in the global stiffness matrix and the appropriate components of the global load vector.

The element connectivity arrays generally do not have to be *stored explicitly* in the element data structure. Optimality of the treatment of connectivity information strongly depends upon the data structure for the representation of *hp*-meshes, and the level of sophistication of the data structure depends upon the expectations that we put into the finite element code. In general, the design of optimal data structures for *hp*-adaptivity is a difficult issue. We will mention a few of its aspects in Paragraph 3.7.1.

Unique enumeration of master element shape functions

This is the easy part of the enumeration scheme. In Chapter 2, the master element shape functions were split into *hierarchic nodes* according to their construction and properties (for instance, for H^1-conforming approximations in 3D there are a *vertex* node, *edge* node, *face* node and *interior* node). Unique enumeration of master element shape functions is achieved by enumerating uniquely these nodes and the shape functions in these nodes.

Unique enumeration of global basis functions

To preserve the compatibility with first-order approximations, it is convenient to enumerate vertex functions first, following a unique global enumeration of grid vertices. One leaves out vertices lying on a Dirichlet boundary.

After that, higher-order global basis functions are added. Unique enumeration of elements, together with unique enumeration of higher-order nodes on the master element and unique enumeration of shape functions in these nodes, determine a unique enumeration of basis functions of the space $V_{h,p}$. See Remark 3.28 for efficient enumeration of elements in adaptive algorithms. The algorithm for the enumeration of higher-order global basis functions consists of three loops: first (outer) loop over elements, second loop over master element higher-order nodes and third loop over shape functions in these nodes. If the shape function does not form part of a Dirichlet lift, and if the corresponding global basis function was not yet visited, its index is set and the value of a counter increased.

Storing versus reconstructing element connectivity arrays

The simplest working alternative, at least from the point of view of the assembling procedure, would be (in the same way as in Section 1.3) to store the element connectivity arrays elementwise. In this case the overlapping information may be quite memory-intensive if finite elements of higher polynomials orders are used. Advanced *hp*-adaptive finite element codes (see, e.g., [64, 63]) prefer to construct the element connectivity arrays in each element step of the assembling algorithm, before calculating the element stiffness matrix and element load vector.

3.5.7 Variational formulation on the reference domain

The reader knows from the 1D model problem in Section 1.3 that the assembling algorithm needs the approximate variational formulation (1.22) translated elementwise from the physical mesh to the reference domain. Let us demonstrate this operation briefly in the case of the Poisson equation $-\triangle u = f$:

Let $K_i \in \mathcal{T}_{h,p}$ be a mesh element, $\hat{K}$ the appropriate reference domain, and $\boldsymbol{x}_K : \hat{K} \to K_i$ a smooth bijective reference map. Without loss of generality we assume that the Jacobian of $\boldsymbol{x}_{K_i}$ is positive. The corresponding element integral from the approximate variational formulation has the form

$$\int_{K_i} \nabla u_{h,p} \cdot \nabla v_{h,p} \, \mathrm{d}\boldsymbol{x}. \tag{3.81}$$

Using the Substitution Theorem together with the relation (3.60) from Remark 3.13, (3.81) can be written as

$$\int_{\hat{K}} J_{K_i}(\boldsymbol{\xi}) \left[\left(\frac{\mathrm{D}\boldsymbol{x}_{K_i}}{\mathrm{D}\boldsymbol{\xi}} \right)^{-T} \nabla \tilde{u}_{h,p}^{(i)} \right] \cdot \left[\left(\frac{\mathrm{D}\boldsymbol{x}_{K_i}}{\mathrm{D}\boldsymbol{\xi}} \right)^{-T} \nabla \tilde{v}_{h,p}^{(i)} \right] \mathrm{d}\boldsymbol{\xi}, \tag{3.82}$$

where

$$\tilde{u}_{h,p}^{(i)}(\boldsymbol{\xi}) = (u_{h,p} \circ \boldsymbol{x}_{K_i})(\boldsymbol{\xi}), \quad \tilde{v}_{h,p}^{(i)}(\boldsymbol{\xi}) = (v_{h,p} \circ \boldsymbol{x}_{K_i})(\boldsymbol{\xi}).$$

In the same way as in the 1D model example, the transformed functions $\tilde{u}_{h,p}^{(i)}, \tilde{v}_{h,p}^{(i)}$ will be expressed by means of master element shape functions.

In the assembling algorithm we will need to calculate explicitly the partial derivatives of the inverse reference map $\boldsymbol{x}_{K_i}^{-1}$. Let us denote them by

$$\left(\frac{\mathrm{D}\boldsymbol{x}_{K_i}}{\mathrm{D}\boldsymbol{\xi}} \right)^{-1} = \left\{ \frac{\partial \xi_i}{\partial x_j} \right\}_{i,j=1}^{d}.$$

We can rewrite (3.82) further as

$$\sum_{m=1}^{d} \int_{\hat{K}} J_{K_i}(\boldsymbol{\xi}) \left(\sum_{s=1}^{d} \frac{\partial \tilde{u}^{(i)}}{\partial \xi_s} \frac{\partial \xi_s}{\partial x_m} \right) \left(\sum_{r=1}^{d} \frac{\partial \tilde{v}^{(i)}}{\partial \xi_r} \frac{\partial \xi_r}{\partial x_m} \right) \mathrm{d}\boldsymbol{\xi}. \tag{3.83}$$

At this point we decide whether (3.83) can be further simplified, i.e., whether

1. the reference map $\boldsymbol{x}_{K_i}$ is affine (i.e., all of its partial derivatives and the Jacobian J_{K_i} are constant),
2. the operator $L_{h,p}$ in the approximate variational formulation (1.22) does not depend explicitly on space. Obviously the Laplace operator in the model equation $-\Delta u = f$ does not, but this is not the general case – imagine, for instance, the stationary heat-transfer equation with spatially dependent material parameters.

If the answer to at least one of these questions is **no**, the assembling algorithm has to consider the element stiffness integrals in the general form (3.83). If the answer to both of them is **yes**, one further rewrites (3.83) as

$$\sum_{m=1}^{d} J_{K_i} \sum_{s=1}^{d} \frac{\partial \xi_s}{\partial x_m} \sum_{r=1}^{d} \frac{\partial \xi_r}{\partial x_m} \int_{\hat{K}} \frac{\partial \tilde{u}^{(i)}}{\partial \xi_s} \frac{\partial \tilde{v}^{(i)}}{\partial \xi_r} \mathrm{d}\boldsymbol{\xi}. \tag{3.84}$$

REMARK 3.20 (Choice of the assembling algorithms) The essential difference between (3.83) and (3.84) is that in the former case the assembling algorithm has to perform numerical quadrature on all mesh elements $K_i \in \mathcal{T}_{h,p}$, while in the latter case it only has to perform the numerical integration of stiffness terms corresponding to all relevant combinations of master element shape functions once on the reference domain. $\square$

REMARK 3.21 (Number of precomputed values for (3.84)) To illustrate, the number of precomputed master element stiffness integrals for (3.84) is $(dN_{dof,max})^2$, where d is the spatial dimension and $N_{dof,max}$ the number of shape functions corresponding to element with the highest polynomial order in the mesh. $\square$

Boundary integrals corresponding to Neumann boundary conditions are evaluated analogously. Concerning the right-hand side, one transforms

$$\int_{K_i} f(\boldsymbol{x}) v_{h,p}(x)\,\mathrm{d}\boldsymbol{x} = \int_{\hat{K}} \tilde{f}^{(i)}(\boldsymbol{\xi}) \tilde{v}_{h,p}^{(i)}(x)\,\mathrm{d}\boldsymbol{x}, \tag{3.85}$$

where

$$\tilde{f}^{(i)}(\boldsymbol{\xi}) = (f \circ \boldsymbol{x}_{K_i})(\boldsymbol{\xi}).$$

Since f generally depends on the spatial variable, numerical quadrature in (3.85) has to be performed on all mesh elements.

3.5.8 Local and global assembling procedures

The construction of the discrete problem is typically performed in two steps. First one evaluates local (element) matrices and load vectors on the reference domain using orientation-adjusted master element shape functions (Paragraph 3.5.2). In the second (global) step one uses the connectivity information (Paragraph 3.5.6) to distribute the entries of the local element stiffness matrices and load vectors into the global stiffness matrix and load vector. For the sake of simplicity, let us still stay with the Poisson equation $-\triangle u = f$.

Local step: calculation of element matrices and load vectors

We find it useful to begin with the more general version of the assembling algorithm that also will work for spatially dependent operators and general polynomial reference maps (Remark 3.20). Remark 3.24 describes how Algorithm 3.1 can be simplified when master element stiffness integrals can be precomputed (relation (3.84)). The algorithm is quite standard and can be found, e.g., in [60]. Hence, for each element $K \in \mathcal{T}_{h,p}$ we perform the following steps:

ALGORITHM 3.1 (Local assembling step)

1. *By $N_{dof,K}$ denote the number of shape functions associated with the element K. Include shape functions that represent a Dirichlet lift (Dirichlet boundary conditions will be incorporated in the global step). Initiate the element matrix $\boldsymbol{S}^K$ (type $N_{dof,K} \times N_{dof,K}$) and the element load vector $\boldsymbol{f}_K$ (length $N_{dof,K}$) with zeros.*

2. *Retrieve element geometry degrees of freedom $\boldsymbol{G}_K$ (elementwise stored real matrices of the type $d \times N_{dof,K}$ where d is the spatial dimension) that define the reference map $\boldsymbol{x}_K$. Recall that there are d real coefficients for each scalar shape function involved in the definition of $\boldsymbol{x}_K$.*

3. *Retrieve orientation information relevant for all edges and faces of the element K.*

4. *Evaluate the volume integrals in the variational formulation as follows: In a loop over all integration points $\boldsymbol{\xi}_i \in \hat{K}$ do*

   (a) *Evaluate the (orientation-adjusted) master element shape functions $\hat{\theta}_k(\boldsymbol{\xi}_i)$, $k = 1, \dots, N_{dof,K}$ and their derivatives $[\partial\hat{\theta}_k/\partial\xi_m](\boldsymbol{\xi}_i)$, $m = 1, \dots, d$, using the orientation information from Step 3.*

   (b) *Compute the physical coordinates of the integration point $\boldsymbol{x}_i = \boldsymbol{x}_K(\boldsymbol{\xi}_i)$ and the derivatives $[\partial\boldsymbol{x}_K/\partial\xi_s](\boldsymbol{\xi}_i)$, $s = 1, \dots, d$ as follows:*

      i. *Initiate $\boldsymbol{x}_i$ and $[\partial\boldsymbol{x}_K/\partial\xi_s](\boldsymbol{\xi}_i)$, $s = 1, \dots, d$ with zeros.*

      ii. *In a loop over all shape functions $\hat{\theta}_k$ do*

         A. $\boldsymbol{x}_i := \boldsymbol{x}_i + \boldsymbol{G}_{K,k}\hat{\theta}_k(\boldsymbol{\xi}_i)$,

         B. $[\partial\boldsymbol{x}_K/\partial\xi_s](\boldsymbol{\xi}_i) := [\partial\boldsymbol{x}_K/\partial\xi_s](\boldsymbol{\xi}_i) + \boldsymbol{G}_{K,k}[\partial\hat{\theta}_k/\partial\xi_s](\boldsymbol{\xi}_i)$, $s = 1, \dots, d$.

         End of loop through scalar master element shape functions.

   (c) *Evaluate the magnitude of Jacobian $J_K(\boldsymbol{\xi}_i) = |\det(D\boldsymbol{x}_K/D\boldsymbol{\xi})(\boldsymbol{\xi}_i)|$ as described in Remark 3.25 below.*

   (d) *Evaluate derivatives of the inverse map $\boldsymbol{x}_K^{-1}$ at the point $\boldsymbol{x}_i$ by inverting the Jacobi matrix $D\boldsymbol{x}_K/D\boldsymbol{\xi}$ as described in Remark 3.26 below. We denote the derivatives of the inverse map by $[\partial\xi_s/\partial x_m](\boldsymbol{x}_i)$.*

   (e) *Evaluate derivatives of the transformed shape functions θ_k at the point $\boldsymbol{x}_i = \boldsymbol{x}_K(\boldsymbol{\xi}_i)$ with respect to the physical coordinates. Recall that different transformation rules are used for H^1-, $\boldsymbol{H}(\mathrm{curl})$-, $\boldsymbol{H}(\mathrm{div})$- and L^2-conforming discretizations (see Paragraph 3.5.3). In the H^1-conforming case we execute the chain rule*

$$\frac{\partial\theta_k}{\partial x_m}(\boldsymbol{x}_i) = \sum_{s=1}^{d} \frac{\partial\hat{\theta}_k}{\partial\xi_s}(\boldsymbol{\xi}_i)\frac{\partial\xi_s}{\partial x_m}(\boldsymbol{x}_i). \tag{3.86}$$

   (f) *Compute the integration weight w_i at $\boldsymbol{\xi}_i$ as*

$$w_i := \hat{w}_i J_K(\boldsymbol{\xi}_i), \tag{3.87}$$

   where $\hat{w}_i$ is the original Gaussian weight at the point $\boldsymbol{\xi}_i$.

   (g) *Evaluate the right-hand side f at the point $\boldsymbol{x}_i$.*

    (h) *Fill the element stiffness matrix $\boldsymbol{S}^K$ and load vector $\boldsymbol{f}^K$ as follows: In an outer loop over all shape functions θ_k, $k = 1, \ldots, N_{dof,K}$ do*

       i. $\boldsymbol{f}^K k := \boldsymbol{f}^K_k + f(\boldsymbol{x}_i)\theta_k(\boldsymbol{x}_i)w_i$.

       ii. *In an inner loop over all shape functions θ_l, $l = 1, \ldots, N_{dof,K}$ do*

$$\boldsymbol{S}^K_{k,l} := \boldsymbol{S}^K_{k,l} + \sum_{m=1}^{d} \frac{\partial \theta_k}{\partial x_m} \frac{\partial \theta_l}{\partial x_m} w_i. \tag{3.88}$$

       (Notice that (3.88) exactly corresponds to (3.83).)
       End of the inner loop over shape functions.

    End of the outer loop over shape functions.

  End of loop through integration points.

5. *Evaluate the surface integrals in the variational formulation.*

REMARK 3.22 (Storage of element stiffness matrices and load vectors) Indeed it is not mandatory to store the element stiffness matrices and element load vectors for all elements $K_1, K_2, \ldots, K_M \in \mathcal{T}_{h,p}$ in the memory simultaneously. The way one does it depends on whether she/he wants to use them for other (e.g., preconditioning) purposes or not. Otherwise it is possible to distribute their entries into the global stiffness matrix each time as soon as an element is finished. $\square$

REMARK 3.23 (Evaluation of surface integrals) The boundary integrals are evaluated analogously, in a loop over all edges (or faces in 3D) lying on the Neumann boundary. We begin from the parametrization of the edge (face), and for all 1D (2D) Gaussian integration points execute a similar procedure as above. $\square$

REMARK 3.24 (Simplification of Algorithm 3.1 relevant for (3.84)) Since the load function $f = f(\boldsymbol{x})$, one still has to perform the quadrature on every mesh element K in order to construct the element load vector. Hence, calculations of element matrix and load vector have to be split.

As for the construction of the element stiffness matrix on K: one eliminates the numerical integration by leaving out the loop over integration points $\boldsymbol{\xi}_i$. The outer and inner loop over shape functions will stay. Derivatives and inverse derivatives of the reference map $\boldsymbol{x}_K$ are only evaluated once for each element K. For each pair of shape functions θ_k, θ_l one has to perform an operation corresponding to (3.84), i.e., to evaluate the integral

$$\boldsymbol{S}^K_{k,l} := \boldsymbol{S}^K_{k,l} + \sum_{m=1}^{d} J_K \sum_{s=1}^{d} \frac{\partial \xi_s}{\partial x_m} \sum_{r=1}^{d} \frac{\partial \xi_r}{\partial x_m} \underbrace{\int_{\hat{K}} \frac{\partial \hat{\theta}_k}{\partial \xi_s}(\boldsymbol{\xi}) \frac{\partial \hat{\theta}_l}{\partial \xi_r}(\boldsymbol{\xi})\, \mathrm{d}\boldsymbol{\xi}}_{\text{precomputed value } I_{klsr}} . \tag{3.89}$$

Hence we see that it is sufficient to store a single set of precomputed integrals I_{klsr}, $1 \le k, l \le N_{dof,K}$, $1 \le s, r \le d$. $\square$

REMARK 3.25 (Calculation of determinants) In 2D, the determinant $\det(\mathrm{D}\boldsymbol{x}_K/\mathrm{D}\boldsymbol{\xi})(\boldsymbol{\xi}_i)$ has the form

$$\det\left(\frac{\mathrm{D}\boldsymbol{x}_K}{\mathrm{D}\boldsymbol{\xi}}\right)(\boldsymbol{\xi}_i) = \det\begin{pmatrix} \frac{\partial x_{K,1}}{\partial \xi_1}(\boldsymbol{\xi}_i) & \frac{\partial x_{K,1}}{\partial \xi_2}(\boldsymbol{\xi}_i) \\ \frac{\partial x_{K,2}}{\partial \xi_1}(\boldsymbol{\xi}_i) & \frac{\partial x_{K,2}}{\partial \xi_2}(\boldsymbol{\xi}_i) \end{pmatrix} = \det\begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc. \tag{3.90}$$

The following *expansion rule* can be used for the computation of determinants of larger matrices of the type $n \times n$ (for instance $\det(\mathrm{D}\boldsymbol{x}_K/\mathrm{D}\boldsymbol{\xi})(\boldsymbol{\xi}_i)$ in 3D):

$$\det(\boldsymbol{A}) = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} \det\left(\boldsymbol{A}^{(ij)}\right). \tag{3.91}$$

Here $\boldsymbol{A}^{(ij)}$ stands for a matrix of the type $(n-1) \times (n-1)$ that is obtained by leaving out the ith row and the jth column out of the original matrix $\boldsymbol{A}$. See any basic textbook on linear algebra. $\square$

REMARK 3.26 (Inversion of matrices) It is convenient to invert the Jacobi matrix $[D\boldsymbol{x}_K/D\boldsymbol{\xi}](\boldsymbol{\xi}_i)$ using another basic rule of linear algebra: Let $\boldsymbol{A}$ be a regular $n \times n$ matrix. Then, the entries $\bar{a}_{ij}$ of the inverse matrix $\boldsymbol{A}^{-1}$ are defined by

$$\bar{a}_{ij} = (-1)^{i+j} \frac{\det\left(\boldsymbol{A}^{(ij)}\right)}{\det(\boldsymbol{A})}, \quad 1 \le i, j \le n. \tag{3.92}$$

The meaning of the symbol $\boldsymbol{A}^{(ij)}$ is the same as in Remark 3.25. $\square$

Global step: assembly of the resulting discrete system

The most important factor in this step is the connectivity information, which was discussed in Paragraph 3.5.6. In this step Dirichlet boundary conditions will be incorporated into the global discrete problem. The idea of the algorithm is identical to what it was in 1D (see Section 1.3).

One proceeds in a single loop over all elements (Remark 3.22 applies). For each element $K_k \in \mathcal{T}_{h,p}$ one establishes the element connectivity array $\boldsymbol{c}_i$ of length N_{dof,K_k}, where N_{dof,K_k} is the number of all relevant master element shape functions (including the ones used for the Dirichlet lift). The order of entries in this array corresponds to the unique enumeration of master element shape functions (also defined in Paragraph 3.5.6), and the entries are indices of global basis functions that are related to the master element shape functions. Again one may use, e.g., a negative index -1 in order to distinguish master element shape functions which do not correspond to any global basis function since they form part of a Dirichlet lift.

ALGORITHM 3.2 (Global assembling step)

```
⋮
(element loop:) for $k = 1, 2, \dots, M$ do {
  (1st loop over shape (test) functions:) for $i = 1, 2, \dots, N_{dof,K_k}$ do {
    (2nd loop over shape (basis) functions:) for $j = 1, 2, \dots, N_{dof,K_k}$ do {
      put $m_1 = c_{k,i}$ (if not $-1$, this is global index of a test function $v_{m_1} \in V_{h,p}$, i.e., row in the global stiffness matrix $\boldsymbol{S}$)
      put $m_2 = c_{k,j}$ (if not $-1$, this is global index of a basis function $v_{m_2} \in V_{h,p}$, i.e., column in the global stiffness matrix $\boldsymbol{S}$)
      if($m_1 \neq -1$ and $m_2 \neq -1$) then put $s_{m_1,m_2} = s_{m_1,m_2} + s_{i,j}^{(k)}$
      else { (beginning of treatment of Dirichlet bdy. conditions)
        if($m_1 \neq -1$ and $m_2 == -1$) then put $F_{m_1} = F_{m_1} - s_{i,j}^{(k)}$ (contribution to the right-hand side $\boldsymbol{f}$ from the Dirichlet lift $u^*_{h,p}$)
      } (end of treatment of Dirichlet bdy. conditions)
    } (end of 2nd loop over shape (basis) functions)
    if($m_1 \neq -1$) then put $F_{m_1} = F_{m_1} + F_i^{(k)}$ (regular contribution to the right-hand side $\boldsymbol{F}$)
  } (end of 1st loop over shape (test) functions)
} (end of element loop)
```

Above, $\boldsymbol{S}^{K_k} = \{s_{i,j}^{(k)}\}$ and $\boldsymbol{f}^{K_k} = (F_1^{(k)}, \dots, F_{N_{dof,K_k}}^{(k)})^T$ stand for the element matrix and the load vector corresponding to element K_k, respectively.

3.5.9 Static condensation of internal DOF

This procedure is a great tool for the parallelization of p and hp finite element codes for linear problems. It consists of three steps:

1. reduction of the size of the discrete problem by leaving out all bubble functions,

2. solution of the reduced linear system and

3. calculation of the remaining coefficients for the bubble functions by solving elementwise local linear problems.

Since in 2D and 3D the more complicated structure of shape functions works against the clarity of the presentation, let us explain the basic idea using a simple 1D example (see also, e.g., [172]).

The global degrees of freedom associated with basis functions that are nonzero within single mesh elements (bubble functions) are viewed as *internal*, while the remaining DOF (associated with basis functions that are nonzero within more than one mesh element) are by definition *external*. This splitting translates naturally into the structure of the physical element shape functions: Consider a one-dimensional element K_i, $1 \le i \le M$, and an appropriate smooth bijective reference map $\boldsymbol{x}_{K_i} : K_a \to K_i$, $K_a = [-1, 1]$. We may consider an additional condition

$$\frac{\partial \boldsymbol{x}_{K_i}}{\partial \xi} > 0 \quad \text{for all} \quad 1 \le i \le M \quad \text{and} \quad \xi \in K_a, \tag{3.93}$$

to make the maps unique. Let K_i be equipped with a polynomial order of approximation p_i, $i = 1, \ldots, M$. For each element K_i there are two external shape functions

$$\varphi^{(i)}_{ext,1} = \varphi^{v_1}_a \circ \boldsymbol{x}^{-1}_{K_i}, \ \varphi^{(i)}_{ext,2} = \varphi^{v_2}_a \circ \boldsymbol{x}^{-1}_{K_i},$$

and $p_i - 1$ internal shape functions

$$\varphi^{(i)}_{int,k} = \varphi^{b}_{k+1,a} \circ \boldsymbol{x}^{-1}_{K_i}, \ k = 1, \ldots, p_i - 1.$$

The functions $u_{h,p}$ and $v_{h,p}$ in the approximate variational formulation (1.22), which is in fact equivalent to a discrete problem, can be expressed on the element K_i as linear combinations of the external and internal shape functions with real (complex) coefficients,

$$u_{h,p}|_{K_i} = \sum_{k=1}^{2} u^{(i)}_{ext,k} \varphi^{(i)}_{ext,k} + \sum_{k=1}^{p_i - 1} u^{(i)}_{int,k} \varphi^{(i)}_{int,k} = [\boldsymbol{u}^{(i)}_{ext}]^T \boldsymbol{\varphi}^{(i)}_{ext} + [\boldsymbol{u}^{(i)}_{int}]^T \boldsymbol{\varphi}^{(i)}_{int}, \tag{3.94}$$

and

$$v_{h,p}|_{K_i} = [\boldsymbol{v}^{(i)}_{ext}]^T \boldsymbol{\varphi}^{(i)}_{ext} + [\boldsymbol{v}^{(i)}_{int}]^T \boldsymbol{\varphi}^{(i)}_{int}. \tag{3.95}$$

It is worth noticing that the zero Dirichlet boundary conditions together with (3.93) determine that

$$u^{(1)}_{ext,1} = u^{(M)}_{ext,2} = v^{(1)}_{ext,1} = v^{(M)}_{ext,2} = 0.$$

Inserting the relations (3.94) and (3.95) into the approximate variational formulation (1.22) on the element K_i, we obtain

$$L_{h,p}(u_{h,p}, v_{h,p})|_{K_i} = \begin{pmatrix} \boldsymbol{v}^{(i)}_{ext} \\ \boldsymbol{v}^{(i)}_{int} \end{pmatrix}^T \begin{pmatrix} \boldsymbol{L}^{(i)}_{ext,ext} & \boldsymbol{L}^{(i)}_{int,ext} \\ \boldsymbol{L}^{(i)}_{ext,int} & \boldsymbol{L}^{(i)}_{int,int} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^{(i)}_{ext} \\ \boldsymbol{u}^{(i)}_{int} \end{pmatrix} \tag{3.96}$$
$$= [\boldsymbol{v}^{(i)}_{ext}]^T \boldsymbol{L}^{(i)}_{ext,ext} \boldsymbol{u}^{(i)}_{ext} + [\boldsymbol{v}^{(i)}_{int}]^T \boldsymbol{L}^{(i)}_{int,int} \boldsymbol{u}^{(i)}_{int}$$
$$+ [\boldsymbol{v}^{(i)}_{ext}]^T \boldsymbol{L}^{(i)}_{int,ext} \boldsymbol{u}^{(i)}_{int} + [\boldsymbol{v}^{(i)}_{int}]^T \boldsymbol{L}^{(i)}_{ext,int} \boldsymbol{u}^{(i)}_{ext},$$

and

$$f_{h,p}(v_{h,p})|_{K_i} = [\boldsymbol{v}^{(i)}_{ext}]^T \boldsymbol{f}^{(i)}_{ext} + [\boldsymbol{v}^{(i)}_{int}]^T \boldsymbol{f}^{(i)}_{int}, \tag{3.97}$$

where the relation

$$\begin{pmatrix} \boldsymbol{L}^{(i)}_{ext,ext} & \boldsymbol{L}^{(i)}_{int,ext} \\ \boldsymbol{L}^{(i)}_{ext,int} & \boldsymbol{L}^{(i)}_{int,int} \end{pmatrix} = \boldsymbol{L}_{K_i}$$

represents a partition of the element stiffness matrix with respect to the external and internal degrees of freedom and

$$(\boldsymbol{f}^{(i)}_{ext}, \boldsymbol{f}^{(i)}_{int}) = \boldsymbol{f}_{K_i}$$

is the accordingly partitioned element load vector.

Now we arrive at the basic observation: when testing (1.22) on the element K_i with the bubble functions $\varphi^{(i)}_{int,k}$, $k = 1, \ldots, p_i - 1$ only, we obtain a linear system that decouples the matrices $\boldsymbol{L}^{(i)}_{ext,int}$ and $\boldsymbol{L}^{(i)}_{int,int}$,

$$\boldsymbol{L}^{(i)}_{ext,int} \boldsymbol{u}^{(i)}_{ext} + \boldsymbol{L}^{(i)}_{int,int} \boldsymbol{u}^{(i)}_{int} = \boldsymbol{f}^{(i)}_{int}.$$

This means that once we know the coefficients $\boldsymbol{u}^{(i)}_{ext}$ associated with the external DOF, the remaining coefficients $\boldsymbol{u}^{(i)}_{int}$ can be calculated *elementwise* from

$$\boldsymbol{u}^{(i)}_{int} = [\boldsymbol{L}^{(i)}_{int,int}]^{-1} \left(\boldsymbol{f}^{(i)}_{int} - \boldsymbol{L}^{(i)}_{ext,int} \boldsymbol{u}^{(i)}_{ext} \right). \tag{3.98}$$

What remains to be done is to eliminate the internal degrees of freedom from the discrete problem (1.22). We write it elementwise and insert (3.98) into it, after decomposing it into components corresponding to the external and internal DOF using (3.96) and (3.97). We end up with a condensed linear system

$$\sum_{i=1}^{M} [\boldsymbol{v}^{(i)}_{ext}]^T \bar{\boldsymbol{L}}^{(i)}_{ext} \boldsymbol{u}^{(i)}_{ext} = \sum_{i=1}^{M} [\boldsymbol{v}^{(i)}_{ext}]^T \bar{\boldsymbol{f}}^{(i)}_{ext},$$

where the condensed element stiffness matrices $\bar{\boldsymbol{L}}^{(i)}_{ext}$, defined by

$$\bar{\boldsymbol{L}}^{(i)}_{ext} = \boldsymbol{L}^{(i)}_{ext,ext} - \boldsymbol{L}^{(i)}_{int,ext}[\boldsymbol{L}^{(i)}_{int,int}]^{-1}\boldsymbol{L}^{(i)}_{ext,int},$$

are *Schur complements* of the original element stiffness matrices $\boldsymbol{L}_{K_i}$ with respect to the internal degrees of freedom, and the load vectors $\bar{\boldsymbol{f}}^{(i)}_{ext}$ are given by the relation

$$\bar{\boldsymbol{f}}^{(i)}_{ext} = \boldsymbol{f}^{(i)}_{ext} - \boldsymbol{L}^{(i)}_{int,ext}[\boldsymbol{L}^{(i)}_{int,int}]^{-1}\boldsymbol{f}^{(i)}_{int}.$$

Notice that the computation of element stiffness matrices and load vectors as well as the condensation step are *local* for all elements K_i, $i = 1, \ldots, M$, and therefore suitable for parallelization. The effect of the procedure becomes more remarkable as the number of bubble functions in the discretization increases.

3.6 Constrained approximation

The constrained approximation technique has long been used for first-order elements in various applications. To our knowledge, for higher-order elements it was first introduced by Demkowicz, Oden et al. in [66]. It is necessary for an efficient resolution of phenomena that require a high local concentration of degrees of freedom – those are typically boundary and internal layers, regions with steep gradients, singularities, etc. Attempts to resolve these features with regular meshes can lead to inefficient distribution of degrees of freedom and even can spoil the convergence of the finite element scheme.

Hence, the main idea is to employ *irregular meshes* (i.e., meshes with hanging nodes in the sense of Paragraph 1.1.3) in such a way that the approximation still satisfies global conformity requirements – continuity of the approximation across element interfaces for H^1-conforming approximations, and continuity of the tangential and normal component for $\boldsymbol{H}(\mathrm{curl})$- and $\boldsymbol{H}(\mathrm{div})$-conforming approximations, respectively. Perhaps the easiest way to understand the constrained approximation is to view it in terms of change of basis in a given polynomial space.

3.6.1 Continuous constrained approximation in 2D

Let us begin with the simplest case of H^1-conforming constrained approximation in two spatial dimensions. Since bubble functions are local in element interiors, all the action will take place along element edges, involving vertex and edge functions only. Let us assume a setting depicted in Figure 3.21, involving three elements K_1, K_3 and K_4. The small elements K_3, K_4 are descendants (sons) of an element K_2, which was a neighbor of K_1 at the previous refinement level. Recall from Paragraph 3.5 that each unconstrained physical mesh edge is equipped with a unique orientation given by the enumeration of its vertices. The edges $e_2 = v_1v_3$, $e_3 = v_3v_2$ are sons of the edge $e_1 = v_1v_2$ and *by definition they inherit its orientation.*

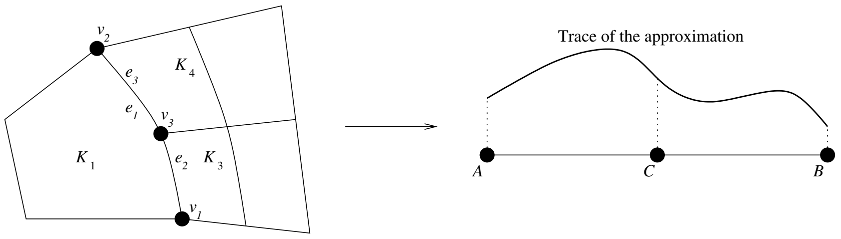

FIGURE 3.21: Constrained continuous approximation in two spatial dimensions. For triangular elements or hybrid meshes the situation is analogous.

In Figure 3.21 it is

$$\begin{aligned} v_1 &= \boldsymbol{x}^{e_1}(A), \\ v_2 &= \boldsymbol{x}^{e_1}(B), \\ v_3 &= \boldsymbol{x}^{e_1}(C), \end{aligned}$$

$\boldsymbol{x}^{e_1} : [A, B] \to \mathbb{R}^2$ being a polynomial parametrization of the edge e_1. This parametrization has to be compatible with the reference mapping $\boldsymbol{x}_{K_1}$ obtained by means of *projection-based interpolation* of a *transfinite interpolant* based on the original, *possibly nonpolynomial* parametrization of the edge e_1. The original and the resulting polynomial parametrization of edges should not be confused; recall the construction of reference mappings from Section 3.3.

We assume that the edge e_1 is equipped with a local polynomial order of approximation $p = p^{e_1} \geq 1$. By definition, the sons e_2, e_3 of e_3 will inherit this local polynomial order of approximation. Let us take a closer look at the relations between unconstrained and constrained DOF from Figure 3.21. In the following, coefficients corresponding to basis functions of the space $U_{h,p}$ that are associated with constrained DOF will be called *constrained*, too. Hence,

- unconstrained (constraining) are: the vertex coefficients $\alpha^{v_1}, \alpha^{v_2}$ and edge coefficients $\alpha_k^{e_1}$, $2 \leq k \leq p$.
- Constrained are: the vertex coefficient α^{v_3} and edge coefficients $\alpha_k^{e_2}, \alpha_k^{e_3}$, $2 \leq k \leq p$.

Consider the space V_{AB}^p of scalar polynomials of the order p or lower defined in the interval AB and by V_{AC}^p and V_{CB}^p denote the spaces of polynomials of the order p or lower defined in the subintervals AC and CB, respectively. Restrictions of a function $\theta \in V_{AB}^p$ (representing the trace of the finite element approximation to the edge e_1) to the subintervals AC, CB obviously lie in the polynomial spaces V_{AC}^p and V_{CB}^p, respectively. The space V_{AB}^p is equipped with the polynomial basis

$$\mathcal{B}_{AB}^p = \{l_j^{AB}\}_{j=0}^p = \left\{ l_k \left(-1 + 2\frac{\zeta - A}{B - A} \right);\ \zeta \in [A, B];\ k = 0, \ldots, p \right\},$$

obtained by transforming the one-dimensional shape functions $l_0, l_1, \ldots$, given by (2.6) and (2.8), to the interval AB. Recall from Chapter 2 that these basis functions match the traces of two-dimensional shape functions associated with the physical mesh element K_1 (which can be either a quadrilateral or a triangle). In the same way we equip the spaces V_{AC}^p and V_{CB}^p with the bases

$$\mathcal{B}_{AC}^p = \{l_j^{AC}\}_{j=0}^p = \{l_j^L\}_{j=0}^p = \left\{ l_k \left(-1 + 2\frac{\zeta - A}{C - A} \right);\ k = 0, \ldots, p \right\},$$

$\zeta \in [A, C]$ and

$$\mathcal{B}_{CB}^p = \{l_j^{CB}\}_{j=0}^p = \{l_j^R\}_{j=0}^p = \left\{ l_k \left(-1 + 2\frac{\zeta - C}{B - C} \right);\ k = 0, \ldots, p \right\},$$

$\zeta \in [C, B]$. Notice that the sets

$$\mathcal{B}_{AB}^p|_{AC} = \{\phi|_{AB};\ \phi \in \mathcal{B}_{AB}^p\}$$

and

$$\mathcal{B}_{AB}^p|_{CB} = \{\phi|_{CB};\ \phi \in \mathcal{B}_{AB}^p\},$$

obtained by restricting the long interval basis $\mathcal{B}_{AB}^p$ to the subintervals AC and CB, also constitute bases of the spaces V_{AC}^p and V_{CB}^p, respectively.

Now trivia of the linear algebra enter the picture – consider the *transition matrices*

$$\begin{aligned} M_L^p &: \mathcal{B}_{AB}^p|_{AC} \to \mathcal{B}_{AC}^p, \\ M_R^p &: \mathcal{B}_{AB}^p|_{CB} \to \mathcal{B}_{CB}^p, \end{aligned}$$

defining linear transformations that perform the change of basis in the polynomial spaces V_{AC}^p and V_{CB}^p, respectively. Recall that in our case the transition matrices are $(p+1) \times (p+1)$ matrices that have in their columns coefficients of original basis functions expressed in terms of the new basis.

Define a set of "long edge" coefficients

$$\boldsymbol{\alpha}^{e_1} = (\alpha^{v_1}, \alpha^{v_2}, \alpha_2^{e_1}, \dots, \alpha_p^{e_1})$$

and write the function θ in the form

$$\theta = \alpha^{v_1} l_0^{AB} + \alpha^{v_2} l_1^{AB} + \sum_{k=2}^{p} \alpha_k^{e_1} l_k^{AB} \in V_{AB}^p.$$

The *constrained "short edge" coefficients* $\boldsymbol{\alpha}^{e_2} = (\alpha^{v_1}, \alpha^{v_3}, \alpha_2^{e_2}, \dots, \alpha_p^{e_2})$ and $\boldsymbol{\alpha}^{e_3} = (\alpha^{v_3}, \alpha^{v_2}, \alpha_2^{e_3}, \dots, \alpha_p^{e_3})$ are finally obtained from the relations

$$\begin{aligned} \boldsymbol{\alpha}^{e_2} &= M_L^p \boldsymbol{\alpha}^{e_1}, \\ \boldsymbol{\alpha}^{e_3} &= M_R^p \boldsymbol{\alpha}^{e_1}. \end{aligned} \tag{3.99}$$

In the constrained approximation treatment of the orientation information is an essential issue. Recall from Section 3.5 how the global (physical mesh) DOF are represented on the master elements.

There are basically two algorithmic ways to handle the additional algebraic relations (3.99). One way is to treat the approximation as *discontinuous along the constrained edges* during the process of construction of the discrete problem. In this case the extra equations (3.99) are added to the resulting discrete system. It is convenient to have them eliminated prior to preconditioning and solving the discrete problem, which can be done easily with a suitable sparse matrix representation.

Another possibility is to use the relations (3.99) already during the assembling of the discrete system (see, e.g., [64] and other papers from this series). When evaluating the variational formulation, integral contributions corresponding to the constrained degrees of freedom are immediately transferred to the right-hand side. Hence, one spends more time during the assembling but saves on the solution of the discrete problem.

Although these two approaches are fully equivalent mathematically, they are not at all equivalent from the point of view of computer implementation. We prefer the first option as it increases the modularity of the code with respect to the construction and solution of the discrete problem.

Properties and evaluation of the transition matrices M_L^p, M_R^p

There are a few nice properties of the transition matrices M_L^p and M_R^p that simplify the implementation of the constrained approximation. First, they depend on the position of the point C within the interval AB, but they are independent of the length of AB. Second, with a hierarchic selection of one-dimensional shape functions (in our case $l_0, l_1, \dots$) the matrices are *hierarchic*

– M_L^{p+1} and M_R^{p+1} are obtained by adding one new row and one new column to the matrices M_L^p and M_R^p, respectively. Thus in principle it is sufficient to precompute and store two real matrices only that correspond to the maximum polynomial order of approximation.

Let us assume that the point C lies in the middle of the interval AB. Since the transition matrices are independent of the length of the interval AB, we can choose, for example, $A := -1$, $B := 1$ and $C := 0$. This choice simplifies things because

$$l_k^{AB} = l_k, \quad k = 0, 1, \dots$$

with the 1D hierarchic shape functions $l_0, l_1, \dots$ given by (2.6), (2.8). Basis functions for the right-hand side subinterval CB have the form

$$\begin{aligned}
l_0^R &= 1 - x, \\
l_1^R &= x, \\
l_2^R &= 2x^2 - 2x, \\
l_3^R &= 4x^3 - 6x^2 + 2x, \\
l_4^R &= 10x^4 - 20x^3 + 12x^2 - 2x, \\
&\vdots
\end{aligned} \tag{3.100}$$

It is not difficult to compute

$$\begin{aligned}
l_0|_{CB} &= \left(\frac{1}{2}, 0\right) \cdot \left(l_0^R, l_1^R\right)^T, \\
l_1|_{CB} &= \left(\frac{1}{2}, 1\right) \cdot \left(l_0^R, l_1^R\right)^T,
\end{aligned}$$

which means that

$$M_R^1 = \begin{pmatrix} 1/2 & 1/2 \\ 0 & 1 \end{pmatrix}.$$

For quadratic elements we have

$$l_2|_{CB} = \left(-\frac{1}{2}, 0, \frac{1}{4}\right) \cdot \left(l_0^R, l_1^R, l_2^R\right)^T$$

and thus

$$M_R^2 = \begin{pmatrix} 1/2 & 1/2 & -1/2 \\ 0 & 1 & 0 \\ 0 & 0 & 1/4 \end{pmatrix}.$$

The relation

$$l_3|_{CB} = \left(0, \frac{1}{2}, \frac{3}{8}, \frac{1}{8}\right) \cdot \left(l_0^R, l_1^R, l_2^R, l_3^R\right)^T$$

yields

$$M_R^3 = \begin{pmatrix} 1/2 & 1/2 & -1/2 & 0 \\ 0 & 1 & 0 & 1/2 \\ 0 & 0 & 1/4 & 3/8 \\ 0 & 0 & 0 & 1/8 \end{pmatrix}$$

and so on.

Analogously the basis functions

$$\begin{aligned} l_0^L &= -x, \\ l_1^L &= x+1, \\ l_2^L &= 2x^2 + 2x, \\ l_3^L &= 4x^3 + 6x^2 + 2x, \\ l_4^L &= 10x^4 + 20x^3 + 12x^2 + 2x, \\ &\vdots \end{aligned} \tag{3.101}$$

corresponding to the left-hand side subinterval AC, yield

$$\begin{aligned} l_0|_{AC} &= \left(1, \frac{1}{2}\right) \cdot \left(l_0^L, l_1^L\right)^T, \\ l_1|_{AC} &= \left(0, \frac{1}{2}\right) \cdot \left(l_0^L, l_1^L\right)^T, \\ l_2|_{AC} &= \left(0, -\frac{1}{2}, \frac{1}{4}\right) \cdot \left(l_0^L, l_1^L, l_2^L\right)^T, \\ l_3|_{AC} &= \left(0, 0, -\frac{3}{8}, \frac{1}{8}\right) \cdot \left(l_0^L, l_1^L, l_2^L, l_3^L\right)^T, \\ &\vdots \end{aligned}$$

Hence, the resulting hierarchic transition matrix M_L^3 for (at most) cubic approximation reads

$$M_L^3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1/2 & 1/2 & -1/2 & 0 \\ 0 & 0 & 1/4 & -3/8 \\ 0 & 0 & 0 & 1/8 \end{pmatrix}.$$

3.6.2 Vector-valued constrained approximation in 2D

The situation will be very similar to the scalar case that we discussed in the previous paragraph. Since the whole procedure including the resulting transition matrices is the same for both the spaces $\boldsymbol{H}(\mathrm{curl})$ and $\boldsymbol{H}(\mathrm{div})$ in 2D, we will confine ourselves to the $\boldsymbol{H}(\mathrm{curl})$-conforming case only. Hence it will be our task to preserve the continuity of the trace of the *tangential component* of the approximation across element interfaces. We will consider the same setting as in the previous paragraph that was depicted in Figure 3.21. Also the symbols V_{AB}^p, V_{AC}^p and V_{CB}^p will have the same meaning as before.

Recall from Chapter 2 that in both the $\boldsymbol{H}(\mathrm{curl})$- and $\boldsymbol{H}(\mathrm{div})$-conforming cases the traces of the tangential and normal component of master element shape functions coincide with the *Legendre polynomials* $L_0, L_1, \ldots$, given by (1.48). This is different from the scalar continuous case where we dealt with the functions $l_0, l_1, \ldots$ based on integrated Legendre polynomials. Therefore, an appropriate choice for the polynomial basis of the space V_{AB}^p is now

$$\mathcal{B}_{AB}^p = \{L_j^{AB}\}_{j=0}^p = \left\{ L_k\left(-1 + 2\frac{\zeta - A}{B - A}\right);\ \zeta \in [A, B];\ k = 0, \ldots, p \right\}. \tag{3.102}$$

These basis functions correspond to the traces of the tangential component of the edge functions associated with the element K_1. Again everything will also hold for triangular elements and hybrid meshes.

The polynomial bases for the spaces V_{AC}^p and V_{CB}^p are written as

$$\mathcal{B}_{AC}^p = \{L_j^{AC}\}_{j=0}^p = \{L_j^L\}_{j=0}^p = \left\{ L_k\left(-1 + 2\frac{\zeta - A}{C - A}\right);\ k = 0, \ldots, p \right\},$$

$\zeta \in [A, B]$, and

$$\mathcal{B}_{CB}^p = \{L_j^{CB}\}_{j=0}^p = \{L_j^R\}_{j=0}^p = \left\{ L_k\left(-1 + 2\frac{\zeta - C}{B - C}\right);\ k = 0, \ldots, p \right\},$$

$\zeta \in [C, B]$. In the same way as before, notice that the sets

$$\mathcal{B}_{AB}^p|_{AC} = \{\phi|_{AB};\ \phi \in \mathcal{B}_{AB}^p\}$$

and

$$\mathcal{B}_{AB}^p|_{CB} = \{\phi|_{CB};\ \phi \in \mathcal{B}_{AB}^p\},$$

obtained by restricting the long interval basis $\mathcal{B}_{AB}^p$ to the subintervals AC and CB, also represent bases in the spaces V_{AC}^p and V_{CB}^p, respectively. We define a set of "long edge" coefficients

$$\boldsymbol{\alpha}^{e_1} = (\alpha_0^{e_1}, \alpha_1^{e_1}, \alpha_2^{e_1}, \ldots, \alpha_p^{e_1})$$

and write the function θ in the form

$$\theta = \sum_{k=0}^{p} \alpha_k^{e_1} L_k^{AB} \in V_{AB}^p.$$

The *constrained "short edge" coefficients* $\boldsymbol{\alpha}^{e_2} = (\alpha_0^{e_2}, \alpha_1^{e_2}, \alpha_2^{e_2}, \dots, \alpha_p^{e_2})$ and $\boldsymbol{\alpha}^{e_3} = (\alpha_0^{e_3}, \alpha_1^{e_3}, \alpha_2^{e_3}, \dots, \alpha_p^{e_3})$ are given by

$$\begin{aligned} \boldsymbol{\alpha}^{e_2} &= M_L^p \boldsymbol{\alpha}^{e_1}, \\ \boldsymbol{\alpha}^{e_3} &= M_R^p \boldsymbol{\alpha}^{e_1}. \end{aligned} \tag{3.103}$$

The transition matrices M_L^p and M_R^p are different from the previous scalar continuous case. Recall that vertex functions are present neither among the $\boldsymbol{H}(\mathrm{curl})$ nor among the $\boldsymbol{H}(\mathrm{div})$ shape functions. Thus all components of the coefficient vectors $\boldsymbol{\alpha}^{e_2}, \boldsymbol{\alpha}^{e_3}$ correspond to *edge functions* associated with the edges e_2 and e_3 of the small elements.

For illustration purposes let us make a choice $A := -1$, $B := 1$ and $C := 0$, leading to

$$L_k^{AB} = L_k, \quad k = 0, 1, \dots.$$

Transforming the Legendre polynomials to the right-hand side subinterval CB, we obtain the basis functions

$$\begin{aligned} L_0^R &= 1, \\ L_1^R &= 2x - 1, \\ L_2^R &= 6x^2 - 6x + 1, \\ L_3^R &= 20x^3 - 30x^2 + 12x - 1, \\ &\vdots \end{aligned}$$

Expressing the long interval lowest-order basis function L_0 in terms of the local ones,

$$L_0|_{CB} = L_0^R,$$

we obtain the lowest-order transition matrix

$$M_R^0 = 1$$

of the size 1×1. Thus, the issue of constrained approximation is extremely simple when dealing with lowest-order approximation only. For first-order approximation we use the relation

$$L_1|_{CB} = \left(\frac{1}{2}, \frac{1}{2}\right) \cdot \left(L_0^R, L_1^R\right)^T$$

to find that the transition matrix has the form

$$M_R^1 = \begin{pmatrix} 1 & 1/2 \\ 0 & 1/2 \end{pmatrix}.$$

For quadratic elements we have

$$L_2|_{CB} = \left(0, \frac{3}{4}, \frac{1}{4}\right) \cdot \left(L_0^R, L_1^R, L_2^R\right)^T$$

which means that

$$M_R^2 = \begin{pmatrix} 1 & 1/2 & 0 \\ 0 & 1/2 & 3/4 \\ 0 & 0 & 1/4 \end{pmatrix}.$$

The relation

$$L_3|_{CB} = \left(-\frac{1}{8}, \frac{3}{8}, \frac{5}{8}, \frac{1}{8}\right) \cdot \left(L_0^R, L_1^R, L_2^R, L_3^R\right)^T$$

yields

$$M_R^3 = \begin{pmatrix} 1 & 1/2 & 0 & -1/8 \\ 0 & 1/2 & 3/4 & 3/8 \\ 0 & 0 & 1/4 & 5/8 \\ 0 & 0 & 0 & 1/8 \end{pmatrix}$$

and so on.

Transforming the Legendre polynomials $L_0, L_1, \ldots$ to the subinterval AC, we obtain the basis functions

$$\begin{aligned} L_0^L &= 1, \\ L_1^L &= 2x + 1, \\ L_2^L &= 6x^2 + 6x + 1, \\ L_3^L &= 20x^3 + 30x^2 + 12x + 1, \\ &\vdots \end{aligned}$$

that lead to the hierarchic transition matrix

$$M_L^3 = \begin{pmatrix} 1 & -1/2 & 0 & 1/8 \\ 0 & 1/2 & -3/4 & 3/8 \\ 0 & 0 & 1/4 & -5/8 \\ 0 & 0 & 0 & 1/8 \end{pmatrix}$$

good for cubic finite elements.

Exercise 3.1 *Extend the matrices M_L^p, M_R^p to $p \geq 4$ for both the continuous and vector-valued approximations.*

Exercise 3.2 *Find the relation between the matrices M_L^p and M_R^p. Consider first the vector-valued, then the continuous approximation.*

3.6.3 Continuous constrained approximation in 3D

While the mathematical issues related to the constrained approximation technique in 3D are at a similar level of complexity as in two spatial dimensions, its algorithmic aspects become significantly more pronounced. In the same way as in 2D it is highly recommended that the reader be familiar with the H^1-conforming case before dealing with vector-valued approximations, since the extension is straightforward. Moreover, the less tight structure of master element shape functions in the spaces $\boldsymbol{H}(\mathrm{curl})$ and $\boldsymbol{H}(\mathrm{div})$ does not complicate the situation.

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

1. two quadrilateral faces constrained by one quadrilateral face,
2. four quadrilateral faces constrained by one quadrilateral face,
3. four triangular faces constrained by one triangular face and
4. two triangular faces constrained by one quadrilateral face.

(The triangular faces may belong to prisms and/or tetrahedra.) The first three cases are quite standard – we will discuss only Case 1 in more detail and leave Cases 2 and 3 to the reader as an exercise. Case 4 represents a less known but advantageous alternative for joining higher-order bricks and tetrahedra in hybrid meshes *without prismatic elements.*

Case 1: Two quadrilateral faces constrained by another quadrilateral face

Let us consider the model situation depicted in Figure 3.22. In Section 3.5 we assigned a unique global orientation to all mesh edges and triangular and quadrilateral faces based on a unique enumeration of vertices. Hence, assume that the edges v_1v_2 and v_1v_4 specify the global orientation of the face s_1. The same orientation, by definition, is inherited by the small faces s_2 and s_3, respectively (or, more precisely, with the only change that the edge v_1v_4 is replaced by its sons v_1v_5 and v_5v_4). The face s_1 comes with two local directional orders of approximation $p^{s_1,1}$, corresponding to the first direction

v_1v_2, and $p^{s_1,2}$, associated with the other direction v_1v_4. By definition, these directional polynomial orders are inherited by the small faces s_2 and s_3.

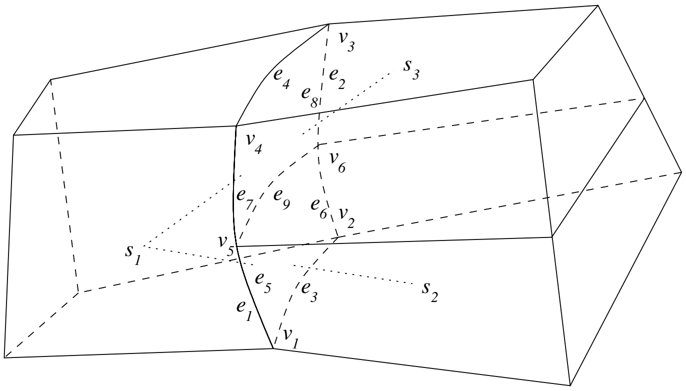

FIGURE 3.22: Constrained continuous approximation 3D. Case 1: small quadrilateral faces s_2, s_3 constrained by a big quadrilateral face s_1.

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

$$\begin{aligned} p^{s_2,1} = p^{s_3,1} &= p^{s_1,1} \\ p^{s_2,2} = p^{s_3,2} &= p^{s_1,2} \\ p^{e_5} = p^{e_7} &= p^{e_1}, \\ p^{e_6} = p^{e_8} &= p^{e_2}, \\ p^{e_9} = p^{s_1,1}&. \end{aligned} \tag{3.104}$$

Now one could strictly copy the 2D procedure, i.e., construct for each face s_2 and s_3 a transition matrix that converts the coefficients corresponding to DOF associated with the face s_1 to coefficients related to faces s_2 and s_3, respectively. In Cases 3 and 4 this may be the most convenient solution. However, in Cases 1 and 2, due to the product structure of quadrilateral faces, these transition matrices would contain almost exclusively zeros. In other words, the number of constraining relations is dramatically less than $m_b m_s$ where m_b and m_s stand for the total number of DOF associated with the big (constraining) and small (constrained) face, respectively. Thus especially in Cases 1 and 2 it is convenient to go deeper into the structure of the shape functions in order to avoid unnecessary extra algebraic equations.

REMARK 3.27 Notice that the transition matrices are of the type $m_s \times m_b$ where generally $m_s \neq m_b$. In our Case 1, the inequality occurs for s_2 if $p^{s_1,1} \neq p^{e_4}$ and for s_3 if $p^{s_1,1} \neq p^{e_3}$. $\square$

More to the point, the coefficients $\alpha^{v_5}, \alpha_k^{e_5}, \alpha_k^{e_7}$, $2 \leq k \leq p^{e_1}$ corresponding to the vertex v_5 and edges e_5, e_7 are only constrained by the coefficients $\alpha^{v_1}, \alpha^{v_4}, \alpha_k^{e_1}$, $2 \leq k \leq p^{e_1}$ associated with the edge e_1. Notice that this is exactly the same situation as in the 2D continuous case discussed in Paragraph 3.6.1. Therefore the transition matrices $M_L^{p^{e_1}}$, $M_R^{p^{e_1}}$ represent the algebraic relations between these coefficients. The global orientation of the edge e_1 can be either v_1v_4 or v_4v_1. Depending on this we relate the matrices $M_L^{p^{e_1}}$, $M_R^{p^{e_1}}$ to the edges e_5, e_7 in this or reverse order.

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

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

- The face s_1 is parametrized by a smooth bijective mapping from a master face $\hat{s}_q = [-1,1]^2$,

$$\boldsymbol{x}^{s_1} : \hat{s}_q \to \mathbb{R}^3.$$

- The edges e_3, e_4 and e_9 are parametrized by smooth bijective maps from a master edge $\hat{e} = [-1,1]$,

$$\boldsymbol{x}^{e_j} : \hat{e} \to \mathbb{R}^3, \quad j = 3,4,9,$$

  such that

$$\begin{aligned} \boldsymbol{x}^{e_3}(o^{e_3}\zeta) &= \boldsymbol{x}^{s_1}(\zeta,-1) \quad \forall \zeta \in [-1,1], \\ \boldsymbol{x}^{e_4}(o^{e_4}\zeta) &= \boldsymbol{x}^{s_1}(\zeta,1) \quad \forall \zeta \in [-1,1], \\ \boldsymbol{x}^{e_9}(\zeta) &= \boldsymbol{x}^{s_1}(\zeta,0) \quad \forall \zeta \in [-1,1], \end{aligned} \tag{3.105}$$

where o^{e_3}, o^{e_4} are ± 1 orientation factors that compensate for the inconsistency between the orientations of the edges e_3, e_4 and the orientation of the face s_1. The edge e_9 is, by definition, oriented in harmony with the orientation of its father, the face s_1.

All operations are done on the quadrilateral reference face $\hat{s}_q$ as shown in Figure 3.23. This is a natural setting since the global orientation of the face s_1 depends on the global enumeration of grid vertices only.

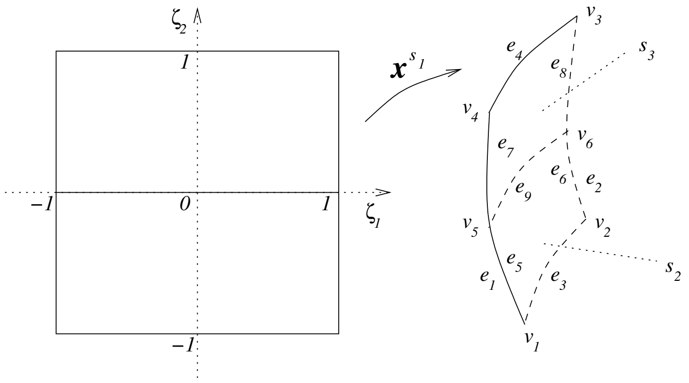

FIGURE 3.23: Quadrilateral reference face $\hat{s}_q = [-1,1]^2$, $s_1 = (\boldsymbol{x}^{s_1})(\hat{s}_q)$.

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

$$\underbrace{\sum_{k=2}^{p^{e_3}} o^{e_3} \alpha_k^{e_3} l_k(\zeta_1) l_0(\zeta_2)}_{\text{contribution of } e_3-\text{edge functions}} + \underbrace{\sum_{k=2}^{p^{e_4}} o^{e_4} \alpha_k^{e_4} l_k(\zeta_1) l_1(\zeta_2)}_{\text{contribution of } e_4-\text{edge functions}} \tag{3.106}$$

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

(recall the definition of edge and face functions for the master brick $\mathcal{K}_B^1$ from Chapter 2). Restricted to the edge $(\boldsymbol{x}^{s_1})^{-1}(e_9)$, (3.106) yields

$$\sum_{k=2}^{p^{s_1,1}} \alpha_k^{e_9} l_k(\zeta_1) = \sum_{k=2}^{p^{e_3}} o^{e_3}\alpha_k^{e_3} l_k(\zeta_1) l_0(0) + \sum_{k=2}^{p^{e_4}} o^{e_4}\alpha_k^{e_4} l_k(\zeta_1) l_1(0)$$
$$+ \sum_{k=2}^{p^{s_1,1}} \sum_{n_2=2}^{p^{s_1,2}} \alpha_{k,n_1}^{s_1} l_k(\zeta_1) l_{n_2}(0) \quad \forall \zeta \in [-1,1].$$

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

$$\alpha_k^{e_9} = o^{e_3}\alpha_k^{e_3} l_0(0) + o^{e_4}\alpha_k^{e_4} l_1(0) + \sum_{n_2=2}^{p^{s_1,2}} \alpha_{k,n_1}^{s_1} l_{n_2}(0), \quad k = 2, \ldots, p^{s_1,1}. \tag{3.107}$$

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

Again it is the *minimum rule* that ensures that $p^{e_3} \le p^{s_1,1}$ and $p^{e_4} \le p^{s_1,1}$, allowing us to simplify the summation in (3.106) to

$$\sum_{k=2}^{p^{s_1,1}} l_k(\zeta_1) \left(o^{e_3}\alpha_k^{e_3} l_0(\zeta_2) + o^{e_4}\alpha_k^{e_4} l_1(\zeta_2) + \sum_{n_2=2}^{p^{s_1,2}} \alpha_{k,n_2}^{s_1} l_{n_2}(\zeta_2) \right) \tag{3.108}$$

(the α's with newly introduced indices are zero). The expression in the brackets is already something that we know from the 2D situation in Paragraph 3.6.1. Hence, for each $k = 0, \ldots, p^{s_1,1}$ we define a vector coefficient

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

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

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

corresponding to the "short edge" basis functions $l_0^L, l_1^L, \ldots, l_{p^{s_1,2}}^L$ (defined in (3.101)) is obtained as

$$\boldsymbol{\alpha}_k^L = M_L^{p^{s_1,2}} \boldsymbol{\alpha}_k.$$

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

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

corresponding to the "short edge" basis functions $l_0^R, l_1^R, \ldots, l_{p^{s_1,2}}^R$ (defined in (3.100)), is obtained as

$$\boldsymbol{\alpha}_k^R = M_R^{p^{s_1,2}} \boldsymbol{\alpha}_k.$$

Exercise 3.3 *Perform in detail the procedure of deriving the algebraic relations between the constraining and constrained coefficients for Cases 2 and 3.*

Case 4: Two triangular faces constrained by a quadrilateral face

This situation is less standard than Cases 1, 2 and 3, and therefore we find it useful to outline the technique in more detail. As mentioned before, this is an alternative way to join bricks and tetrahedra in higher-order meshes without having to use prismatic elements.

Consider the model situation illustrated in Figure 3.24: a quadrilateral face s_1 with vertices $v_1, \ldots, v_4$ constrains two triangular faces s_2, s_3 with vertices v_1, v_2, v_3 and v_1, v_3, v_4, respectively. Without loss of generality assume that $index(v_1)$ is lowest on the face s_1 and that $index(v_2) < index(v_4)$. Thus the global orientation of the face s_1, according to Paragraph 3.5.2, is specified by two directions v_1v_2 and v_1v_4 on that face.

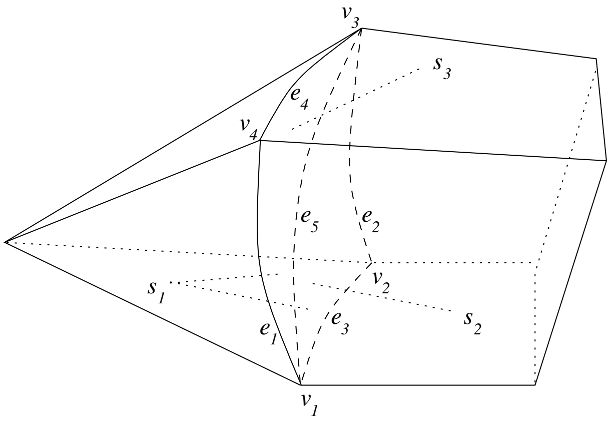

FIGURE 3.24: Case 4: two triangular faces s_2, s_3 constrained by a quadrilateral face s_1, with a single hanging edge.

In order to make life easier, we will orient the faces s_2 and s_3 in a different way than described in Paragraph 3.5.2. Recall that the only reason to have mesh faces (that do not lie on the boundary of the computational domain) globally oriented is to make the traces of the corresponding face basis functions unique because of contributions coming from different mesh elements, and that we are free to orient mesh faces in any way we wish as long as this goal is reached. In the standard situation with two adjacent elements sharing a common triangular face s, that we discussed in Paragraph 3.5.2, it was sufficient to orient s by selecting its vertex A with the lowest global index and determining the direction of its boundary as the direction AB where B was the vertex with the second lowest index on s. Recall that traces of the face functions associated with s are in general *not invariant with respect to rotation of the vertices*, and the vertex A with the lowest index plays a special role in their definition.

In fact, Figure 3.24 does not tell the whole truth, since two different configurations can occur, depending on whether the vertex v_1 with the lowest global index on the face s_1 belongs to the diagonal edge e_5 or not. New definitions of the global orientation for the faces s_2 and s_3 in both these cases are illustrated in Figure 3.25. The vertex that is closest to the arrow now takes the role of the lowest-index vertex A from Paragraph 3.5.2.

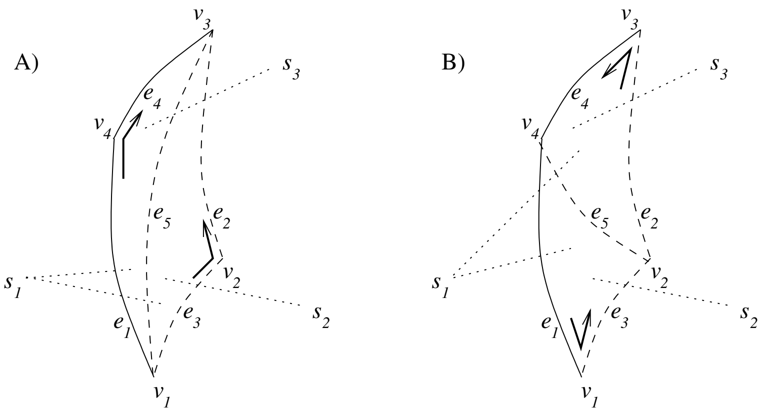

FIGURE 3.25: New definition of the global orientation for triangular faces constrained by a quadrilateral face. A) edge e_5 contains the vertex v_1 with the lowest index. B) the vertex v_1 does not belong to the edge e_5.

One has to take care of both the configurations A) and B) separately; however, fortunately they are not much different. From now on it is sufficient to work in the corresponding reference configurations on the quadrilateral reference face $\hat{s}_q = [-1, 1]^2$, shown in Figure 3.26, from which the face s_1 is parametrized. The cases A) and B) are related to the cases A) and B) in Figure 3.26 where the diagonal lines correspond to the two different positions of the edge e_5 within the faces s_1. As for the global orientation of the edge e_5, we define it in the standard sense following the global enumeration of vertices, i.e., $e_5 = v_1v_3$ in case A and $e_5 = v_2v_4$ in case B.
Without loss of generality, let us assume the following compatibility between parametrizations of the edges and faces:

- The face s_1 is parametrized by a smooth bijective map

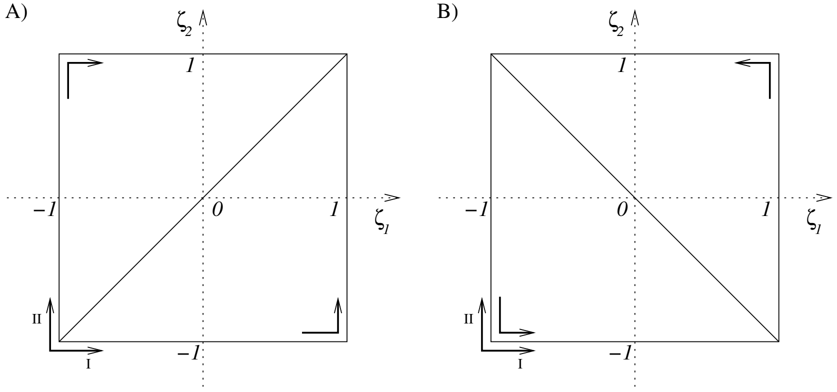

FIGURE 3.26: Reference configurations corresponding to Figure 3.25.

$$\boldsymbol{x}^{s_1} : \hat{s}_q \to s_1 .$$

- The faces s_2, s_3 are parametrized by smooth bijective maps

$$\boldsymbol{x}^{s_2} : \hat{s}_t \to s_2 \text{ and } \boldsymbol{x}^{s_3} : \hat{s}_t \to s_3,$$

where $\hat{s}_t = K_t$ stands for the triangular reference face.

- In case A,

$$\begin{aligned} \boldsymbol{x}^{s_2}(-\zeta_1, \zeta_2) &= \boldsymbol{x}^{s_1}(\zeta_1, \zeta_2) \ \ \forall \boldsymbol{\zeta} \in \hat{s}_q \text{ such that } \zeta_2 < \zeta_1, \\ \boldsymbol{x}^{s_3}(\zeta_1, -\zeta_2) &= \boldsymbol{x}^{s_1}(\zeta_1, \zeta_2) \ \ \forall \boldsymbol{\zeta} \in \hat{s}_q \text{ such that } \zeta_1 < \zeta_2. \end{aligned} \tag{3.109}$$

- In case B,

$$\begin{aligned} \boldsymbol{x}^{s_2}(\zeta_1, \zeta_2) &= \boldsymbol{x}^{s_1}(\zeta_1, \zeta_2) \ \ \forall \boldsymbol{\zeta} \in \hat{s}_q \text{ such that } \zeta_2 < -\zeta_1, \\ \boldsymbol{x}^{s_3}(-\zeta_1, -\zeta_2) &= \boldsymbol{x}^{s_1}(\zeta_1, \zeta_2) \ \ \forall \boldsymbol{\zeta} \in \hat{s}_q \text{ such that } -\zeta_1 < \zeta_2. \end{aligned}$$

- The edges $e_1, \ldots, e_5$ are parametrized by smooth bijective maps

$$\boldsymbol{x}^{e_j} : [-1, 1] \to \mathbb{R}^3$$

such that

$$\boldsymbol{x}^{e_1}(o^{e_1}\zeta) = \boldsymbol{x}^{s_1}(-1,\zeta) \quad \forall \zeta \in [-1,1], \tag{3.110}$$
$$\boldsymbol{x}^{e_2}(o^{e_2}\zeta) = \boldsymbol{x}^{s_1}(1,\zeta) \quad \forall \zeta \in [-1,1],$$
$$\boldsymbol{x}^{e_3}(o^{e_3}\zeta) = \boldsymbol{x}^{s_1}(\zeta,-1) \quad \forall \zeta \in [-1,1], \tag{3.111}$$
$$\boldsymbol{x}^{e_4}(o^{e_4}\zeta) = \boldsymbol{x}^{s_1}(\zeta,1) \quad \forall \zeta \in [-1,1],$$
$$\boldsymbol{x}^{e_5}(\zeta) = \begin{cases} \boldsymbol{x}^{s_1}(\zeta,\zeta) \quad \forall \zeta \in [-1,1] \text{ in case A}, \\ \\ \boldsymbol{x}^{s_1}(-\zeta,\zeta) \ \forall \zeta \in [-1,1] \text{ in case B}. \end{cases}$$

Here again $o^{e_1}, \ldots, o^{e_4}$ are ± 1 orientation factors that compensate for the generally different orientations of the edges $e_1, \ldots, e_4$ and the orientation of the face s_1.

Let the edges $e_1, \ldots, e_4$ be equipped with local orders of approximation $1 \leq p^{e_1}, \ldots, p^{e_4}$. The face s_1 comes with two local directional orders $1 \leq p^{s_1,1}$ and $1 \leq p^{s_1,2}$ in the appropriate local directions I and II. In order that the discussed version of the constrained approximation technique can be performed, the local orders of approximation associated with the edge e_5 and with the faces s_2, s_3 must satisfy the following conditions:

$$p^{e_9} \geq p^{s_1,1} + p^{s_1,2}, \tag{3.112}$$
$$p^{s_2} \geq p^{s_1,1} + p^{s_1,2}, \tag{3.113}$$
$$p^{s_3} \geq p^{s_1,1} + p^{s_1,2}. \tag{3.114}$$

The reason for the necessary conditions (3.112) – (3.114) lies in the different structure of the polynomial spaces $\mathcal{Q}_{p^{s_1,1},p^{s_1,2}}(\hat{s}_q)$ and $\mathcal{P}_{p^{s_2}}(\hat{s}_t)$ (see relations (2.12) and (2.19) for their definition).

In other words, one has to increase the local orders of approximation on the edge e_5 and on the faces s_2, s_3 quite significantly in order to allow the polynomial spaces on the faces s_1, s_2 and s_3 to fit together. In addition, it is important to notice that the *minimum rule* for H^1-conforming approximations (see Paragraph 3.5.5) enforces the order of approximation in the interiors of both of the elements containing the faces s_2 and s_3 to be at least equal to p^{s_2} and p^{s_3}, respectively. This can be considered the price for leaving out prismatic elements from hybrid tetrahedral/hexahedral meshes.

The rest – construction of the algebraic relations between the *constraining* DOF associated with the face s_1 and the *constrained* DOF related to the edge e_9 and faces s_2, s_3 – is easy. First of all, the coefficients related to vertex functions associated with the vertices $v_1, \ldots, v_4$, and of edge functions related to the edges $e_1, \ldots, e_4$, stay unchanged (i.e., these DOF are not constrained). The remaining relations depend on the position of the edge e_5 in the face s_1.

The higher-order DOF associated with the edge e_5 are constrained by the vertex DOF related to the vertices v_2, v_4 (case A) or vertices v_1, v_3 (case B), by all higher-order DOF associated with the edges $e_1, \ldots, e_4$ and moreover by all face DOF of s_1. Hence, the transition matrix is in both cases A and B of the type $m_1 \times m_2$ where

$$m_1 = p^{e_5} - 1$$

and

$$m_2 = 2 + \sum_{k=1}^{4} (p^{e_k} - 1) + (p^{s_1,1} - 1)(p^{s_1,2} - 1).$$

The explicit form of this matrix is not difficult to obtain thanks to the product structure of the traces of the shape functions associated with the face s_1, using the change of variables (3.109) and (3.110).

Finally, the face DOF associated with the faces s_2 and s_3 are constrained by the face DOF related to the face s_1 and by the higher-order DOF of the edge e_5 only. The explicit form of the transition matrices is again an easy exercise in basic algebraic calculus.

Exercise 3.4 *Perform in detail the procedure of deriving the algebraic relations between the constraining and constrained coefficients for Case 4 (two triangular faces constrained by one quadrilateral face).*

3.6.4 Vector-valued constrained approximation in 3D

The generalization of the constrained approximation procedure from scalar continuous discretizations to the vector-valued case is straightforward. The first and essential step toward its understanding is to become familiar with Paragraph 3.6.3. One might find it useful to solve Exercises 3.3 and 3.4 prior to proceeding.

Let us begin with the simpler case – in the $\boldsymbol{H}(\mathrm{div})$-conforming situation one has to guarantee the continuity of the *normal component* of the approximation on interelement *faces* only. The procedure described in Paragraph 3.6.3 applies, instead of function values, to the normal component of the approximation in almost unchanged form. The only significant difference is that while in the continuous case the trace was based on the *integrated Legendre polynomials*, now the original Legendre polynomials will be used. See Chapter 2 for the exact definition of the $\boldsymbol{H}(\mathrm{div})$-conforming master element shape functions and the definition of polynomial spaces for their normal components.

The $\boldsymbol{H}(\mathrm{curl})$-conforming case is more delicate in the sense that one has to guarantee the continuity of the trace of the *tangential component* of the approximation on both the interelement *faces* and *edges*. On both the edges and faces (recall that there are two linearly independent directions per face) the tangential component is based on the integrated Legendre polynomials. See Chapter 2 for the exact definition of the $\boldsymbol{H}(\mathrm{curl})$-conforming master element shape functions and the polynomial spaces for their tangential components both on the edges and faces.

3.7 Selected software-technical aspects

Readers who have some experience with adaptive schemes (not necessarily finite elements) will confirm that the amount of stored mesh-dependent information often determines the distribution of complexity between the solver and the mesh division algorithm. The total complexity of *hp*-adaptive schemes is such that its optimal management is a crucial issue on which the success or failure of the whole implementation can depend. Of special importance are questions where and in which manner to store the information about *element neighbors*, and how to incorporate the *connectivity* and *orientation* information into the data structure.

3.7.1 Data structure for *hp*-adaptivity

A significant contribution to the development and testing of data structures for *hp*-adaptivity was done in FORTRAN by Demkowicz et al. (see, e.g., [66, 63, 60] and other papers in this series). We find it useful to extract from these studies the basic principles that on the one hand can help the reader to avoid beginner's mistakes, but on the other hand leave her/him enough freedom to use her/his own programming experience to design suitable data structures, and algorithms operating over these data structures, in a way she/he prefers. We are aware of the fact that it is difficult to find an equilibrium that would satisfy all readers at once – in each case a lot of concrete details, including prints of parts of FORTRAN codes with concrete data structures and algorithms, can be found in the papers above.

Basic principles

- The complexity of *hp* mesh-division algorithms is such that minimum mesh-dependent data should be stored.
- It is advantageous to always start the adaptivity from a regular initial mesh (i.e., from a mesh with no hanging nodes).
- *Initial mesh.* The regular initial mesh plays a special role in the data structure since obviously all elements, faces, edges and vertices resulting from consequent *hp*-refinements are descendants of initial mesh elements, faces and edges. It is advantageous to view all objects in the *hp*-mesh, i.e., the initial mesh elements, refined elements, vertices, edges and faces, as abstract *nodes* in the sense of Paragraph 1.1.3. Vertices, edges, faces and elements of the initial mesh may be stored in independent arrays. The information about element neighbors as well as the orientation and connectivity information may be stored *for the initial mesh elements only.* More precisely, initial mesh elements may contain

links to their edge-neighbors (2D), face-neighbors (3D), vertex nodes, edge nodes, face nodes and the mid-element node, as well as the information about orientation of edges and faces as discussed in Paragraph 3.5.2. This implies that some work has to be done for each refined node in order to determine its orientation information, indices of the appropriate node neighbors and other necessary *horizontal* information; however, this *essentially simplifies* the *hp* mesh-division algorithm. Demkowicz [63] suggests storing initial mesh elements, all vertices and all higher-order nodes in three separate arrays.

- *Sequences of hp-meshes.* Sequences of *hp*-meshes obtained by consequent refinement of the initial mesh can be stored in the form of *trees of nodes* growing independently from initial mesh edge nodes, initial mesh face nodes and initial mesh mid-element nodes. It turns out that viewing these entities in the initial mesh as *independent* is in harmony with the mathematical theory and the nature of hierarchic *hp*-algorithms.
- *Refinement of nodes.* Both in 2D and 3D the *h*-refinement of an edge produces three new nodes: one new vertex and two new edges. For simplicity we call the new nodes *children* of the edge and the edge their *parent node.* In the same way, *h*-refinement of a triangular face into four yields seven new nodes: three new edges and four new faces (notice that the new vertices are children *of the edges*) as illustrated in Figure 3.27.

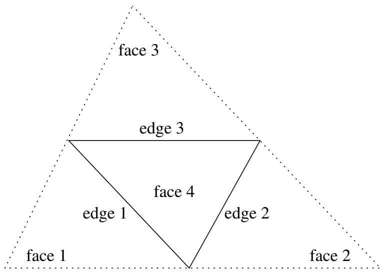

FIGURE 3.27: Seven nodes (children) resulting from a 4-refinement of a triangular face. The scheme obviously is the same for a 4-refinement of a triangular element with the only exception that new elements instead of new faces appear.

It is important to understand the relations between parents and children correctly – children of an edge e are not children of faces or elements where the edge e is contained and so on. Each child has, in this abstract sense, one single parent only.

The trees are frequently climbed and descended in the *vertical direction* while, except for the initial mesh, *no horizontal information is stored.*

Therefore, each parent node may store pointers to all of its children and each child may store a pointer to its parent.

In an object-oriented programming language vertex nodes, edge nodes, face nodes and mid-element nodes may be defined as descendant types of a single basic class that contains, for example, a single variable identifying the node type only. All other variables can be chosen independently for various descendants.

REMARK 3.28 (Natural order of elements) In Paragraph 3.5.6 we have shown that the enumeration of vertices and elements together determine a unique enumeration of basis functions of the space $V_{h,p}$. With many levels of refinement, the question of optimal enumeration of elements becomes more pronounced. A sophisticated enumeration scheme, based on *natural order of elements*, is used in [60]. One enumerates elements in the refinement trees rowwise as illustrated in Figure 3.28.

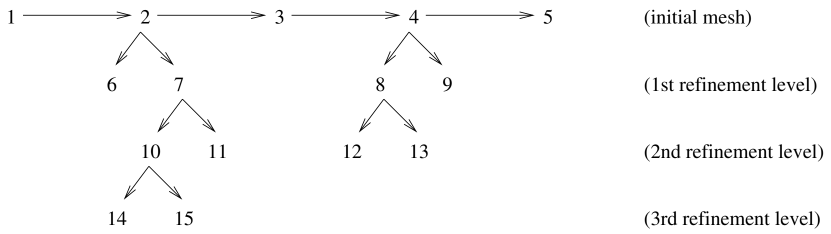

FIGURE 3.28: Natural order of elements, in this case 1, 6, 14, 15, 11, 3, 12, 13, 9, 5.

This approach is advantageous due to its additivity (newly created elements do not force the old ones to change their indices) and simplicity of computer implementation. ▯

3.7.2 One-irregular mesh division algorithms

The treatment of constrained approximations was described in Section 3.6. Recall that meshes are called *one-irregular* if the parent of a constrained node cannot be further constrained. The mesh 1-irregularity rule translates into the condition that an element can be refined only if *all of its edges and vertices are unconstrained.* If an element has a constrained edge or vertex, one first has to identify the coarse element sharing the constrained edge, and subdivide it before one can refine the smaller element. This is realized by means of a *waiting list*, where the smaller element is put while refinement of the coarse element is attempted:

```
ALGORITHM 3.3 (1-irregular mesh division algorithm)
As input assume a list of elements selected for h-refinement
(with corresponding anisotropic flags for quadrilaterals).
Do {
  Take the first element from the list.
  For all edges of the element do {
    Determine neighbors sharing the edge.
    If a neighbor lies at coarser level (thus the edge is
    constrained) {
      Add the coarse element to the end of the waiting list
      (if not yet there).
      Set on its refinement flag indicating requested
      refinement (in the direction of the edge for quads).
      }
    }
    If some edges were constrained {
      Do not break the element, move it at the end of
      list.
  }
  else {
    Break the element (triangles into four only, quads
    according to their anisotropic flags either to two or
    four).
    Remove the element from the list.
  }
} while the list is not empty.
```

See [63] for a three-dimensional version of this algorithm. Notice that enforcement of the 1-irregularity rule during the execution of the mesh refinement algorithm can result in (unwanted) refinement of additional mesh elements, as illustrated in Figure 3.29.

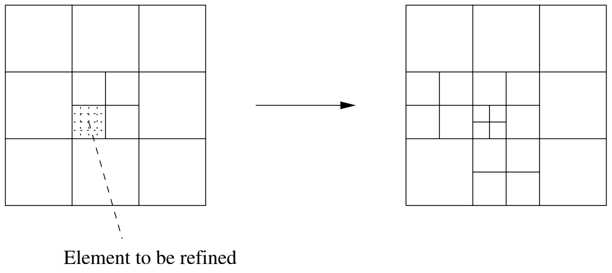

FIGURE 3.29: Unwanted refinements enforced by the 1-irregularity rule.

Chapter 4

Higher-order numerical quadrature

The numerical quadrature technique (sometimes referred to as *cubature* in three spatial dimensions) lies at the heart of all higher-order finite element codes. When evaluating the approximate variational formulation (1.22) on the reference element, one has to numerically integrate higher-order shape functions and their products, their derivatives, sometimes nonlinearities coming from the reference maps and other higher-order terms. The order of accuracy of the numerical integration should correspond to the highest polynomial order that appears behind the integral sign, otherwise one risks a loss of accuracy of the whole scheme and other disagreeable side-effects. Sometimes exact integration is not possible (e.g., because the variational formulation contains nonpolynomial terms) – in such cases one has to be very careful and choose the order of accuracy of the quadrature a little higher rather than a little lower.

We recommend the utmost care in the evaluation of matrices that have to be inverted: the safest way is to evaluate them exactly, which often means to use quadrature formulae at least twice as accurate as the polynomial order of the finite elements. When one underestimates the precision of evaluation of these matrices, their inversion can produce unexpected errors or they even may become singular.

In this chapter we introduce several types of higher-order quadrature schemes related to the reference domains K_a, K_q, K_t, K_B, K_T and K_P from Chapter 2. In our implementations we usually prefer the Gauss quadrature because of its efficiency but we will also mention other standard techniques.

Judgment of quality of quadrature schemes is a risky business – generally it cannot be said that one of the quadrature rules is *better* than the others. Obviously, results for polynomials up to the order n obtained by various quadrature schemes of the same order of accuracy n are *by definition* the same. But we may obtain dramatically different results by applying the same rules to polynomials of order higher than n, nonpolynomial functions or, in the worst case, to functions that oscillate. A limited-order numerical integration of oscillating functions may give arbitrary results. It is the responsibility of everyone to choose quadrature rules that fit as well as possible the nature of the solved problem.

All tables of integration points and weights that will be presented in the following can also be found on the companion CD-ROM, including many more that were too long to be included here. All of them are sufficient for polynomials of the order $p = 20$ or higher, except for economical Gauss quadrature on the reference brick K_B (where to the best of our knowledge the highest known order of accuracy is $p = 19$). Still higher (in fact almost unlimited) order of accuracy can be achieved by means of product rules based on one-dimensional Gauss quadrature.

4.1 One-dimensional reference domain K_a

The basic ideas of the numerical quadrature will be illustrated on one-dimensional schemes. Let $g(y)$ be a function continuous in interval $[a,b]$, $a < b$. The numerical quadrature of order n on this interval is defined as

$$\int_a^b g(y)\mathrm{d}y \approx \sum_{k=0}^{n} A_{n,k}g(y_{n,k}), \tag{4.1}$$

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

$$y = c\xi + d, \qquad c = \frac{b-a}{2}, \qquad d = \frac{b+a}{2}, \tag{4.2}$$

substituting into (4.1) and rearranging, we get a formula corresponding to the one-dimensional reference domain $K_a = (-1, 1)$,

$$\int_{-1}^{1} f(\xi)\mathrm{d}\xi \approx \sum_{k=0}^{n} w_{n,k}f(\xi_{n,k}) \tag{4.3}$$

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

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

Selection of shape functions

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

Probably the most natural choice is to use either the Legendre polynomials or H^1-hierarchic shape functions.

4.1.1 Newton-Cotes quadrature

The quadrature rules of the Newton-Cotes type are generally based on the summation of weighted function values at equidistantly distributed integration points. The $(n+1)$-point Newton-Cotes closed quadrature formula (of accuracy n) for polynomials of the n-th order is given by (4.3), where

$$\xi_{n,k} = -1 + kh_n, \quad k = 0, 1, \ldots, n, \tag{4.4}$$

$h_n = 2/n$ and $n > 0$.

The integration weights may be determined by several different methods based, for example, on the Taylor expansion of $f(\xi)$, Lagrange polynomials or Vandermonde matrix. We will illustrate the last method.

Let $P(\xi)$ be a polynomial of order n expressed as

$$P(\xi) = \sum_{k=0}^{n} p_{n,k}\xi^k \tag{4.5}$$

and let us put

$$\int_{-1}^{1} P(\xi)\mathrm{d}\xi = \sum_{k=0}^{n} \frac{p_{n,k}}{k+1}[1-(-1)^{k+1}] = \sum_{k=0}^{n} w_{n,k}P(\xi_{n,k}). \tag{4.6}$$

Comparison of terms on both sides of (4.6) containing the individual coefficients $p_{n,k}$ leads to a system of linear algebraic equations of the form

$$\begin{aligned}
w_{n,0} + w_{n,1} + \ldots + w_{n,n} &= [1-(-1)^1]/1 = 2, \\
w_{n,0}\xi_{n,0} + w_{n,1}\xi_{n,1} + \ldots + w_{n,n}\xi_{n,n} &= [1-(-1)^2]/2 = 0, \\
&\vdots \\
w_{n,0}\xi^k_{n,0} + w_{n,1}\xi^k_{n,1} + \ldots + w_{n,n}\xi^k_{n,n} &= [1-(-1)^{k+1}]/k+1, \\
&\vdots \\
w_{n,0}\xi^n_{n,0} + w_{n,1}\xi^n_{n,1} + \ldots + w_{n,n}\xi^n_{n,n} &= [1-(-1)^{n+1}]/n+1.
\end{aligned} \tag{4.7}$$

After rearranging the system and using substitution (4.4) for $\xi_{n,k}$ we obtain

$$\begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ 0 & 1 & 2 & \ldots & n \\ 0 & 1^2 & 2^2 & \ldots & n^2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 1^n & 2^n & \ldots & n^n \end{pmatrix} \begin{pmatrix} w_{n,0} \\ w_{n,1} \\ w_{n,2} \\ \ldots \\ w_{n,n} \end{pmatrix} = \begin{pmatrix} 2n^0/1 \\ 2n^1/2 \\ 2n^2/3 \\ \ldots \\ 2n^n/(n+1) \end{pmatrix}. \tag{4.8}$$

The system is characterized by the Vandermonde matrix that is regular and, thus, invertible. Moreover, the weights $w_{n,k}$, $k = 0, \ldots, n$ depend only on parameter n. On the other hand, the Vandermonde matrix is not well-conditioned and for $n > 7$ some weights are negative, which can lead to round-off problems during the evaluation of the right-hand side of (4.6). Therefore, the closed Newton-Cotes formulae are mostly used only for low values of n.

Integration points and weights for lower values of n are shown in Tables 4.1 – 4.7. As the integration points are symmetric with respect to zero, we list only the positive ones. Notice that in one dimension the $(n + 1)$-point closed Newton-Cotes quadrature rule has the order of accuracy n. In general, each closed Newton-Cotes formula is exact for all polynomials whose order is by one degree less than the order of the derivative in its error term. For even values of n the integration is exact for all polynomials of order $n + 1$.

TABLE 4.1: Closed Newton-Cotes quadrature on K_a, order $n = 1$ (trapezoidal rule).

Point #	**$\pm$ ξ-Coordinate**	**Weight**
1.	1	1

TABLE 4.2: Closed Newton-Cotes quadrature on K_a, order $n = 2$ (Simpson's 1/3 rule).

Point #	**$\pm$ ξ-Coordinate**	**Weight**
1.	1	1/3
2.	0	4/3

TABLE 4.3: Closed Newton-Cotes quadrature on K_a, order $n = 3$ (Simpson's 3/8 rule).

Point #	**$\pm$ ξ-Coordinate**	**Weight**
1.	1	1/4
2.	1/3	3/4

TABLE 4.4: Closed Newton-Cotes quadrature on K_a, order $n = 4$ (Bode's rule).

Point #	$\pm$ ξ-Coordinate	Weight
1.	1	7/45
2.	1/2	32/45
3.	0	12/45

TABLE 4.5: Closed Newton-Cotes quadrature on K_a, order $n = 5$.

Point #	$\pm$ ξ-Coordinate	Weight
1.	1	19/144
2.	3/5	75/144
3.	1/5	50/144

TABLE 4.6: Closed Newton-Cotes quadrature on K_a, order $n = 6$.

Point #	$\pm$ ξ-Coordinate	Weight
1.	1	41/420
2.	2/3	216/420
3.	1/3	27/420
4.	0	272/420

TABLE 4.7: Closed Newton-Cotes quadrature on K_a, order $n = 7$.

Point #	$\pm$ ξ-Coordinate	Weight
1.	1	751/8640
2.	5/7	3577/8640
3.	3/7	1323/8640
4.	1/7	2989/8640

In a similar way one can obtain *open Newton-Cotes quadrature formulae* approximating the integral only by function values at internal points of the interval $[a, b]$ (i.e., at the points $\xi_{n,1}, \xi_{n,2}, \ldots, \xi_{n,n-1}$), while the points $\xi_{n,0} = a$ and $\xi_{n,n} = b$ are omitted. These formulae can be used, for example, when values $f(a)$ and $f(b)$ are unavailable. As their errors are much greater than errors of the closed Newton-Cotes formulae, we will not discuss them in detail.

4.1.2 Chebyshev quadrature

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

$$\int_{-1}^{1} f(\xi)\mathrm{d}\xi \approx \frac{2}{n} \sum_{k=1}^{n} f(\xi_{n,k}). \tag{4.9}$$

Notice that the uniform weight $2/n$ is determined from the integration of constant functions. The integration points (abscissas) are obtained after inserting sufficiently many linearly independent functions with known integrals (e.g., the Legendre polynomials or the one-dimensional H^1-hierarchic shape functions for reasons mentioned earlier) into (4.9) and resolving the resulting system of nonlinear algebraic equations. It can be shown that these abscissas may be obtained by using terms up to y^n in the Maclaurin series of the function

$$s_n(z) = exp\left\{\frac{n}{2}\left[-2 + \ln[(1-z^2)(1-z^{-2})]\right]\right\}. \tag{4.10}$$

Then the abscissas are determined as roots of the function

$$C_n(\xi) = \xi^n s_n\left(\frac{1}{\xi}\right). \tag{4.11}$$

The roots are all real only for $n < 8$ and $n = 9$. These values of n represent the only permissible orders for the Chebyshev quadrature. The corresponding functions $C_n(\xi)$ follow:

$$\begin{aligned}
C_2(\xi) &= \frac{1}{3}(3\xi^2 - 1) \\
C_3(\xi) &= \frac{1}{2}(2\xi^3 - \xi) \\
C_4(\xi) &= \frac{1}{45}(45\xi^4 - 30\xi^2 + 1) \\
C_5(\xi) &= \frac{1}{72}(72\xi^5 - 60\xi^3 + 7\xi) \\
C_6(\xi) &= \frac{1}{105}(105\xi^6 - 105\xi^4 + 21\xi^2 - 1) \\
C_7(\xi) &= \frac{1}{6480}(6480\xi^7 - 7560\xi^5 + 2142\xi^3 - 149\xi) \\
C_9(\xi) &= \frac{1}{22400}(22400\xi^9 - 33600\xi^7 + 15120\xi^5 - 2280\xi^3 + 53\xi)
\end{aligned} \tag{4.12}$$

In the one-dimensional case the n-point Chebyshev quadrature rules achieve $(n+1)$-th order of accuracy.

Let us list the integration points for the permitted values of n calculated using Mathematica (and compared with [170]) in Tables 4.8 – 4.14. As the integration points are symmetric with respect to zero, again we list only the positive ones.

TABLE 4.8: Chebyshev quadrature on K_a, order $n + 1 = 3$.

Point #	**$\pm$ ξ-Coordinate**	**Weight**
1.	0.57735 02691 89625 76450 91488	1

TABLE 4.9: Chebyshev quadrature on K_a, order $n + 1 = 4$.

Point #	**$\pm$ ξ-Coordinate**	**Weight**
1.	0.70710 67811 86547 52440 08444	2/3
2.	0.00000 00000 00000 00000 00000	2/3

TABLE 4.10: Chebyshev quadrature on K_a, order $n + 1 = 5$.

Point #	**$\pm$ ξ-Coordinate**	**Weight**
1.	0.79465 44722 91766 12295 55309	1/2
2.	0.18759 24740 85079 89986 01393	1/2

TABLE 4.11: Chebyshev quadrature on K_a, order $n + 1 = 6$.

Point #	**$\pm$ ξ-Coordinate**	**Weight**
1.	0.83249 74870 00981 87589 25836	2/5
2.	0.37454 14095 53581 06558 60444	2/5
3.	0.00000 00000 00000 00000 00000	2/5

TABLE 4.12: Chebyshev quadrature on K_a, order $n+1=7$.

Point #	$\pm\ \xi$-Coordinate	Weight
1.	0.86624 68181 07820 59138 35981	1/3
2.	0.42251 86537 61111 52911 85464	1/3
3.	0.26663 54015 16704 72033 15346	1/3

TABLE 4.13: Chebyshev quadrature on K_a, order $n+1=8$.

Point #	$\pm\ \xi$-Coordinate	Weight
1.	0.88386 17007 58049 03570 42241	2/7
2.	0.52965 67752 85156 81138 50475	2/7
3.	0.32391 18105 19907 63751 96731	2/7
4.	0.00000 00000 00000 00000 00000	2/7

TABLE 4.14: Chebyshev quadrature on K_a, order $n+1=10$.

Point #	$\pm\ \xi$-Coordinate	Weight
1.	0.91158 93077 28434 47366 49486	2/9
2.	0.60101 86553 80238 07142 81279	2/9
3.	0.52876 17830 57879 99326 01816	2/9
4.	0.16790 61842 14803 94306 80319	2/9
5.	0.00000 00000 00000 00000 00000	2/9

REMARK 4.1 Expressions of the type

$$\frac{f(\xi)}{\sqrt{1-\xi^2}} \tag{4.13}$$

can be integrated by means of the Gauss-Chebyshev explicit formula

$$\int_{-1}^{1} \frac{f(\xi)}{\sqrt{1-\xi^2}}\mathrm{d}\xi \approx \frac{\pi}{n}\sum_{i=1}^{n} f\left[\cos\left(\frac{(2i-1)\pi}{2n}\right)\right]. \tag{4.14}$$

$\square$

4.1.3 Lobatto (Radau) quadrature

The quadrature rules of the Lobatto (Radau) type are based on the summation of weighted function values at nonequidistantly distributed integration points containing the endpoints of the interval of integration.

The n-point Lobatto (Radau) quadrature rule for the one-dimensional reference domain $K_a = (-1, 1)$ reads

$$\int_{-1}^{1} f(\xi)\mathrm{d}\xi \approx w_{n,1} f(-1) + \sum_{i=2}^{n-1} w_{n,i} f(\xi_{n,i}) + w_{n,n} f(1). \tag{4.15}$$

Analogously as for the Chebyshev rules, the integration points and weights are obtained after inserting sufficiently many linearly independent functions with known integrals into (4.15) and resolving the resulting system of nonlinear algebraic equations. Notice that for the n-point rule we have $n-2$ unknown points and n weights. Thus we need $2n-2$ equations and consequently this quadrature rule achieves in one spatial dimension only the order of accuracy $2n-3$.

The unknown abscissas $\xi_{n,i}$ are the roots of the polynomial $L_{n-1}^{'}(\xi)$, where $L_{n-1}(\xi)$ is the Legendre polynomial of order $n-1$. The weights of the unknown abscissas are expressed by

$$w_{n,k} = \frac{2}{n(n-1)L_{n-1}^2(\xi_i)}, \quad k = 2, \dots, n-1 \tag{4.16}$$

while the weights at the endpoints are

$$w_{n,1} = w_{n,n} = \frac{2}{n(n-1)}. \tag{4.17}$$

In Tables 4.15 – 4.20 we show the integration points and weights for a few selected orders of accuracy computed using Mathematica (and compared with [126]). As the integration points are symmetric with respect to zero, we list only the positive ones, with the corresponding weights. The weights for symmetric integration points are equal. Additional Lobatto (Radau) quadrature rules up to the order $p = 21$ can be found on the companion CD-ROM.

TABLE 4.15: Lobatto (Radau) quadrature on K_a, order $2n-3=3$.

Point #	**$\pm$ ξ-Coordinate**	**Weight**
1.	1.00000 00000 00000 00000 00000	0.33333 33333 33333 33333 33333
2.	0.00000 00000 00000 00000 00000	1.33333 33333 33333 33333 33333

TABLE 4.16: Lobatto (Radau) quadrature on K_a, order $2n - 3 = 5$.

Point #	$\pm$ ξ-Coordinate	Weight
1.	1.00000 00000 00000 00000 00000	0.16666 66666 66666 66666 66667
2.	0.44721 35954 99957 93928 18347	0.83333 33333 33333 33333 33333

TABLE 4.17: Lobatto (Radau) quadrature on K_a, order $2n - 3 = 7$.

Point #	$\pm$ ξ-Coordinate	Weight
1.	1.00000 00000 00000 00000 00000	0.10000 00000 00000 00000 00000
2.	0.65465 36707 07977 14379 82925	0.54444 44444 44444 44444 44444
3.	0.00000 00000 00000 00000 00000	0.71111 11111 11111 11111 11111

TABLE 4.18: Lobatto (Radau) quadrature on K_a, order $2n - 3 = 9$.

Point #	$\pm$ ξ-Coordinate	Weight
1.	1.00000 00000 00000 00000 00000	0.06666 66666 66666 66666 66667
2.	0.76505 53239 29464 69285 10030	0.37847 49562 97846 98031 66128
3.	0.28523 15164 80645 09631 41510	0.55485 83770 35486 35301 67205

TABLE 4.19: Lobatto (Radau) quadrature on K_a, order $2n - 3 = 11$.

Point #	$\pm$ ξ-Coordinate	Weight
1.	1.00000 00000 00000 00000 00000	0.04761 90476 19047 61904 76190
2.	0.83022 38962 78566 92987 20322	0.27682 60473 61565 94801 07004
3.	0.46884 87934 70714 21380 37719	0.43174 53812 09862 62341 78710
4.	0.00000 00000 00000 00000 00000	0.48761 90476 19047 61904 76190

TABLE 4.20: Lobatto (Radau) quadrature on K_a, order $2n - 3 = 13$. See the companion CD-ROM for additional Lobatto (Radau) quadrature rules up to the order $p = 21$.

Point #	$\pm$ ξ-Coordinate	Weight
1.	1.00000 00000 00000 00000 00000	0.03571 42857 14285 71428 57143
2.	0.87174 01485 09606 61533 74457	0.21070 42271 43506 03938 29921
3.	0.59170 01814 33142 30214 45107	0.34112 26924 83504 36476 42407
4.	0.20929 92179 02478 86876 86573	0.41245 87946 58703 88156 70530

4.1.4 Gauss quadrature

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

$$\int_{-1}^{1} f(\xi)\mathrm{d}\xi \approx \sum_{i=1}^{n} w_{n,i} f(\xi_{n,i}) \tag{4.18}$$

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

It can be shown that the integration points are roots of the Legendre polynomials $L_n(\xi)$, whose values are sufficiently well tabulated. Hence, the complexity of the problem reduces to the level of Newton-Cotes quadrature rules, since with known points the nonlinear system comes over to a system of linear algebraic equations. The analysis leads even further; it is known that the weights $w_{n,i}$ can be expressed as

$$w_{n,i} = \frac{2}{(1-\xi_{n,i}^2)L_n'(\xi)^2}, \quad i = 1, \dots, n \tag{4.19}$$

Let us list the integration points and weights for a few selected n-point rules, which again were computed in Mathematica and compared with [1], in Tables 4.21 – 4.35. As the integration points are symmetric with respect to zero, we list only the positive ones. As usual, symmetric integration points have identical weights. Additional Gauss quadrature rules up to the order $p = 127$ can be found on the companion CD-ROM.

TABLE 4.21: Gauss quadrature on K_a, order $2n-1=3$.

Point #	$\pm\ \xi$-Coordinate	Weight
1.	0.57735 02691 89625 76450 91488	1.00000 00000 00000 00000 00000

TABLE 4.22: Gauss quadrature on K_a, order $2n-1=5$.

Point #	$\pm\ \xi$-Coordinate	Weight
1.	0.00000 00000 00000 00000 00000	0.88888 88888 88888 88888 88889
2.	0.77459 66692 41483 37703 58531	0.55555 55555 55555 55555 55556

TABLE 4.23: Gauss quadrature on K_a, order $2n-1=7$.

Point #	$\pm$ ξ-Coordinate	Weight
1.	0.33998 10435 84856 26480 26658	0.65214 51548 62546 14262 69361
2.	0.86113 63115 94052 57522 39465	0.34785 48451 37453 85737 30639

TABLE 4.24: Gauss quadrature on K_a, order $2n-1=9$.

Point #	$\pm$ ξ-Coordinate	Weight
1.	0.00000 00000 00000 00000 00000	0.56888 88888 88888 88888 88889
2.	0.53846 93101 05683 09103 63144	0.47862 86704 99366 46804 12915
3.	0.90617 98459 38663 99279 76269	0.23692 68850 56189 08751 42640

TABLE 4.25: Gauss quadrature on K_a, order $2n-1=11$.

Point #	$\pm$ ξ-Coordinate	Weight
1.	0.23861 91860 83196 90863 05017	0.46791 39345 72691 04738 98703
2.	0.66120 93864 66264 51366 13996	0.36076 15730 48138 60756 98335
3.	0.93246 95142 03152 02781 23016	0.17132 44923 79170 34504 02961

TABLE 4.26: Gauss quadrature on K_a, order $2n-1=13$.

Point #	$\pm$ ξ-Coordinate	Weight
1.	0.00000 00000 00000 00000 00000	0.41795 91836 73469 38775 51020
2.	0.40584 51513 77397 16690 66064	0.38183 00505 05118 94495 03698
3.	0.74153 11855 99394 43986 38648	0.27970 53914 89276 66790 14678
4.	0.94910 79123 42758 52452 61897	0.12948 49661 68869 69327 06114

TABLE 4.27: Gauss quadrature on K_a, order $2n-1=15$.

Point #	$\pm$ ξ-Coordinate	Weight
1.	0.18343 46424 95649 80493 94761	0.36268 37833 78361 98296 51504
2.	0.52553 24099 16328 98581 77390	0.31370 66458 77887 28733 79622
3.	0.79666 64774 13626 73959 15539	0.22238 10344 53374 47054 43560
4.	0.96028 98564 97536 23168 35609	0.10122 85362 90376 25915 25314

TABLE 4.28: Gauss quadrature on K_a, order $2n - 1 = 17$.

Point #	$\pm$ ξ-Coordinate	Weight
1.	0.00000 00000 00000 00000 00000	0.33023 93550 01259 76316 45251
2.	0.32425 34234 03808 92903 85380	0.31234 70770 40002 84006 86304
3.	0.61337 14327 00590 39730 87020	0.26061 06964 02935 46231 87429
4.	0.83603 11073 26635 79429 94298	0.18064 81606 94857 40405 84720
5.	0.96816 02395 07626 08983 55762	0.08127 43883 61574 41197 18922

TABLE 4.29: Gauss quadrature on K_a, order $2n - 1 = 19$.

Point #	$\pm$ ξ-Coordinate	Weight
1.	0.14887 43389 81631 21088 48260	0.29552 42247 14752 87017 38930
2.	0.43339 53941 29247 19079 92659	0.26926 67193 09996 35509 12269
3.	0.67940 95682 99024 40623 43274	0.21908 63625 15982 04399 55349
4.	0.86506 33666 88984 51073 20967	0.14945 13491 50580 59314 57763
5.	0.97390 65285 17171 72007 79640	0.06667 13443 08688 13759 35688

TABLE 4.30: Gauss quadrature on K_a, order $2n - 1 = 21$.

Point #	$\pm$ ξ-Coordinate	Weight
1.	0.00000 00000 00000 00000 00000	0.27292 50867 77900 63071 44835
2.	0.26954 31559 52344 97233 15320	0.26280 45445 10246 66218 06889
3.	0.51909 61292 06811 81592 57257	0.23319 37645 91990 47991 85237
4.	0.73015 20055 74049 32409 34163	0.18629 02109 27734 25142 60980
5.	0.88706 25997 68095 29907 51578	0.12558 03694 64904 62463 46940
6.	0.97822 86581 46056 99280 39380	0.05566 85671 16173 66648 27537

TABLE 4.31: Gauss quadrature on K_a, order $2n - 1 = 23$.

Point #	$\pm$ ξ-Coordinate	Weight
1.	0.12523 34085 11468 91547 24414	0.24914 70458 13402 78500 05624
2.	0.36783 14989 98180 19375 26915	0.23349 25365 38354 80876 08499
3.	0.58731 79542 86617 44729 67024	0.20316 74267 23065 92174 90645
4.	0.76990 26741 94304 68703 68938	0.16007 83285 43346 22633 46525
5.	0.90411 72563 70474 85667 84659	0.10693 93259 95318 43096 02547
6.	0.98156 06342 46719 25069 05491	0.04717 53363 86511 82719 46160

TABLE 4.32: Gauss quadrature on K_a, order $2n - 1 = 31$.

Point #	$\pm$ ξ-Coordinate	Weight
1.	0.09501 25098 37637 44018 53193	0.18945 06104 55068 49628 53967
2.	0.28160 35507 79258 91323 04605	0.18260 34150 44923 58886 67637
3.	0.45801 67776 57227 38634 24194	0.16915 65193 95002 53818 93121
4.	0.61787 62444 02643 74844 66718	0.14959 59888 16576 73208 15017
5.	0.75540 44083 55003 03389 51012	0.12462 89712 55533 87205 24763
6.	0.86563 12023 87831 74388 04679	0.09515 85116 82492 78480 99251
7.	0.94457 50230 73232 57607 79884	0.06225 35239 38647 89286 28438
8.	0.98940 09349 91649 93259 61542	0.02715 24594 11754 09485 17806

TABLE 4.33: Gauss quadrature on K_a, order $2n - 1 = 39$.

Point #	**± ξ-Coordinate**	**Weight**
1.	0.07652 65211 33497 33375 46404	0.15275 33871 30725 85069 80843
2.	0.22778 58511 41645 07808 04962	0.14917 29864 72603 74678 78287
3.	0.37370 60887 15419 56067 25482	0.14209 61093 18382 05132 92983
4.	0.51086 70019 50827 09800 43641	0.13168 86384 49176 62689 84945
5.	0.63605 36807 26515 02545 28367	0.11819 45319 61518 41731 23774
6.	0.74633 19064 60150 79261 43051	0.10193 01198 17240 43503 67501
7.	0.83911 69718 22218 82339 45291	0.08327 67415 76704 74872 47581
8.	0.91223 44282 51325 90586 77524	0.06267 20483 34109 06356 95065
9.	0.96397 19272 77913 79126 76661	0.04060 14298 00386 94133 10400
10.	0.99312 85991 85094 92478 61224	0.01761 40071 39152 11831 18620

TABLE 4.34: Gauss quadrature on K_a, order $2n - 1 = 47$.

Point #	**± ξ-Coordinate**	**Weight**
1.	0.06405 68928 62605 62608 50431	0.12793 81953 46752 15697 40562
2.	0.19111 88674 73616 30915 86398	0.12583 74563 46828 29612 13754
3.	0.31504 26796 96163 37438 67933	0.12167 04729 27803 39120 44632
4.	0.43379 35076 26045 13848 70842	0.11550 56680 53725 60135 33445
5.	0.54542 14713 88839 53565 83756	0.10744 42701 15965 63478 25773
6.	0.64809 36519 36975 56925 24958	0.09761 86521 04113 88826 98807
7.	0.74012 41915 78554 36424 38281	0.08619 01615 31953 27591 71852
8.	0.82000 19859 73902 92195 39499	0.07334 64814 11080 30573 40336
9.	0.88641 55270 04401 03421 31543	0.05929 85849 15436 78074 63678
10.	0.93827 45520 02732 75852 36490	0.04427 74388 17419 80616 86027
11.	0.97472 85559 13094 98198 39199	0.02853 13886 28933 66318 13078
12.	0.99518 72199 97021 36017 99974	0.01234 12297 99987 19954 68057

TABLE 4.35: Gauss quadrature on K_a, order $2n - 1 = 63$. See the companion CD-ROM for additional Gauss quadrature rules up to the order $p = 127$.

Point #	**± ξ-Coordinate**	**Weight**
1.	0.04830 76656 87738 31623 48126	0.09654 00885 14727 80056 67648
2.	0.14447 19615 82796 49348 51864	0.09563 87200 79274 85941 90820
3.	0.23928 73622 52137 07454 46032	0.09384 43990 80804 56563 91802
4.	0.33186 86022 82127 64977 99168	0.09117 38786 95763 88471 28686
5.	0.42135 12761 30635 34536 41194	0.08765 20930 04403 81114 27715
6.	0.50689 99089 32229 39002 37475	0.08331 19242 26946 75522 21991
7.	0.58771 57572 40762 32904 07455	0.07819 38957 87070 30647 17409
8.	0.66304 42669 30215 20097 51152	0.07234 57941 08848 50622 53994
9.	0.73218 21187 40289 68038 74267	0.06582 22227 76361 84683 76501
10.	0.79448 37959 67942 40696 30973	0.05868 40934 78535 54714 52836
11.	0.84936 76137 32569 97013 36930	0.05099 80592 62376 17619 61632
12.	0.89632 11557 66052 12396 53072	0.04283 58980 22226 68065 68787
13.	0.93490 60759 37739 68917 09191	0.03427 38629 13021 43310 26877
14.	0.96476 22555 87506 43077 38119	0.02539 20653 09262 05945 57526
15.	0.98561 15115 45268 33540 01750	0.01627 43947 30905 67060 51706
16.	0.99726 38618 49481 56354 49811	0.00701 86100 09470 09660 04071

4.2 Reference quadrilateral K_q

In this section we present selected higher-order quadrature rules for the reference quadrilateral domain K_q.

4.2.1 Composite Gauss quadrature

Let us start with a technique that is easiest to implement – quadrature formulae based on the Cartesian product of two one-dimensional quadrature rules in the axial directions ξ_1 and ξ_2. Consider the formula

$$\int_{K_a} f(\xi)\mathrm{d}\xi \approx \sum_{i=1}^{M_a} w_{g_a,i} f(y_{g_a,i}), \tag{4.20}$$

where $y_{g_a,i}, w_{g_a,i}$ are Gauss integration points and weights on the one-dimensional reference domain $K_a = (-1,1)$ that integrate exactly all polynomials of the order p and lower. It is easy to see that the product formula

$$\int_{K_a}\int_{K_a} g(\xi_1,\xi_2)\mathrm{d}\xi_1\mathrm{d}\xi_2 \approx \sum_{i=1}^{M_a}\sum_{j=1}^{M_a} w_{g_a,i}w_{g_a,j}\, g(y_{g_a,i}, y_{g_a,j}) \tag{4.21}$$

is of the order p for polynomials of two independent variables ξ_1, ξ_2.

REMARK 4.2 In addition to its simple implementation, the product formula (4.21) has one more advantage – it can easily be generalized to polynomials with different orders of approximation in the axial directions ξ_1 and ξ_2. Such polynomials may appear naturally as a consequence of p-anisotropic refinements of quadrilateral elements, that may occur, e.g., within boundary and internal layers. Recall that in Chapter 2 the master element shape functions for the reference quadrilateral domain were designed to allow for p-anisotropy. $\square$

However, for *complete* polynomials of order p (with generally $n = (p+1)(p+2)/2$ nonzero terms), much more efficient formulae are known. We introduce them in Paragraph 4.2.2.

4.2.2 Economical Gauss quadrature

In this paragraph we introduce Gauss quadrature rules that require fewer integration points than their product counterparts from Paragraph 4.2.1. Some of them are even known to require *the minimum number* of integration points. For this reason sometimes one calls these quadrature rules *economical*. We design them for complete polynomials only, starting from the symmetry of the reference quadrilateral K_q.

First let us have a look at the integral

$$\int_{-1}^{1}\int_{-1}^{1} \xi_1^j \xi_2^k \, \mathrm{d}\xi_2 \, \mathrm{d}\xi_1, \tag{4.22}$$

where j and k are nonnegative integers. As long as j or k is odd, the integral is equal to zero and the corresponding terms need not be taken into account for calculation. If j and k are zero or even, its value is $4/(j+1)/(k+1)$. Moreover, due to the symmetry of the reference domain K_q with respect to the axes $\xi_1 = \xi_2$ and $\xi_1 = -\xi_2$, it is sufficient to consider only terms in which $j \geq k$.

To give an example, the analysis of complete polynomials of an even order p requires us to consider only the terms $1, \xi_1^2, \ldots, \xi_1^p, \xi_1^2\xi_2^2, \ldots, \xi_1^{p-2}\xi_2^2, \xi_1^4\xi_2^4, \ldots,$ $\xi_1^{p-4}\xi_2^4$, …, etc. Denoting the reduced number of terms by m, for $p = 4$ we have $m = 4$ while $n = (p+1)(p+2)/2 = 15$. For $p = 6$ we have $m = 6$ while $n = 28$, for $p = 20$ it is $m = 36$ while $n = 231$. We see that the symmetry considerations are essential.

4.2.3 Tables of Gauss quadrature points and weights

Dunavant [73] provides a useful overview of minimum numbers of Gaussian quadrature points for quadrilaterals in Table 4.36.

TABLE 4.36: Minimum numbers of quadrature points for the Gauss quadrature over quadrilaterals.

Polyn. order	Minimum num. of nonsym. points	Minimum num. of sym. points	Achieved num. of sym. points
1	1	1	1
2	3	4	4
3	4	4	4
4	6	8	8
5	7	8	8
6	10	12	12
7	12	12	12
8	15	20	20
9	17	20	20
10	21	25	25
11	24	25	25
12	28	36	36
13	31	36	36
14	36	44	45
15	40	44	45
16	45	56	60
17	49	56	60
18	55	68	72
19	60	68	72
20	65	84	88
21	71	84	88

Dunavant divides the integration points into four groups with different numbers of unknowns and tests their best choices. The results obtained by solving the corresponding systems of nonlinear equations are given in Tables 4.37 – 4.40 below. Additional economical Gauss quadrature rules up to the order $p = 21$ can be found on the companion CD-ROM.

TABLE 4.37: Gauss quadrature on K_q, order $p = 0, 1$.

Point #	**ξ_1-Coordinate**	**ξ_2-Coordinate**	**Weight**
1.	0.00000 00000 00000	0.00000 00000 00000	4.00000 00000 00000

TABLE 4.38: Gauss quadrature on K_q, order $p = 2, 3$.

Point #	**ξ_1-Coordinate**	**ξ_2-Coordinate**	**Weight**
1.	0.57735 02691 89626	0.57735 02691 89626	1.00000 00000 00000
2.	0.57735 02691 89626	-0.57735 02691 89626	1.00000 00000 00000
3.	-0.57735 02691 89626	0.57735 02691 89626	1.00000 00000 00000
4.	-0.57735 02691 89626	-0.57735 02691 89626	1.00000 00000 00000

TABLE 4.39: Gauss quadrature on K_q, order $p = 4, 5$.

Point #	**ξ_1-Coordinate**	**ξ_2-Coordinate**	**Weight**
1.	0.68313 00510 63973	0.00000 00000 00000	0.81632 65306 12245
2.	-0.68313 00510 63973	0.00000 00000 00000	0.81632 65306 12245
3.	0.00000 00000 00000	0.68313 00510 63973	0.81632 65306 12245
4.	0.00000 00000 00000	-0.68313 00510 63973	0.81632 65306 12245
5.	0.88191 71036 88197	0.88191 71036 88197	0.18367 34693 87755
6.	0.88191 71036 88197	-0.88191 71036 88197	0.18367 34693 87755
7.	-0.88191 71036 88197	0.88191 71036 88197	0.18367 34693 87755
8.	-0.88191 71036 88197	-0.88191 71036 88197	0.18367 34693 87755

TABLE 4.40: Gauss quadrature on K_q, order $p = 6, 7$. See the companion CD-ROM for additional economical Gauss quadrature rules up to the order $p = 21$.

Point #	**ξ_1-Coordinate**	**ξ_2-Coordinate**	**Weight**
1.	0.92582 00997 72551	0.00000 00000 00000	0.24197 53086 41975
2.	-0.92582 00997 72551	0.00000 00000 00000	0.24197 53086 41975
3.	0.00000 00000 00000	0.92582 00997 72551	0.24197 53086 41975
4.	0.00000 00000 00000	-0.92582 00997 72551	0.24197 53086 41975
5.	0.80597 97829 18599	0.80597 97829 18599	0.23743 17746 90630
6.	0.80597 97829 18599	-0.80597 97829 18599	0.23743 17746 90630
7.	-0.80597 97829 18599	0.80597 97829 18599	0.23743 17746 90630
8.	-0.80597 97829 18599	-0.80597 97829 18599	0.23743 17746 90630
9.	0.38055 44332 08316	0.38055 44332 08316	0.52059 29166 67394
10.	0.38055 44332 08316	-0.38055 44332 08316	0.52059 29166 67394
11.	-0.38055 44332 08316	0.38055 44332 08316	0.52059 29166 67394
12.	-0.38055 44332 08316	-0.38055 44332 08316	0.52059 29166 67394

4.3 Reference triangle K_t

In this section we present selected higher-order quadrature techniques for the reference triangular domain K_t. Algorithmically simpler but less efficient schemes are mentioned in Paragraphs 4.3.1 and 4.3.2. More efficient are of course economical Gauss quadrature rules that we introduce in Paragraph 4.3.3.

4.3.1 Translation of quadrature to the ref. quadrilateral K_q

This is probably the easiest way to perform numerical quadrature on K_t with practically unlimited order of accuracy. The idea of the technique consists in a transformation of the integrated function to the reference quadrilateral K_q and consequent application of the composite Gauss quadrature rules discussed in Paragraph 4.2.1. One can improve the efficiency of this type of quadrature by using the economical Gauss quadrature on K_q (see Paragraph 4.2.2) instead of the product rules.

Consider a function g defined on K_t. Intuitively the procedure can be viewed as "stretching" the function g to be defined on K_q so that the volume under g is conserved. The following proposition defines the technique precisely.

PROPOSITION 4.1
Let $g(\boldsymbol{\xi})$ be a continuous bounded function defined on the reference triangle K_t. Then

$$\int_{K_t} g(\boldsymbol{\xi})\, d\boldsymbol{\xi} = \int_{K_q} \frac{1-y_2}{2} g\left(-1 + \frac{1-y_2}{2}(y_1+1), y_2\right)\, d\boldsymbol{y}. \tag{4.23}$$

PROOF Consider the (degenerate) mapping

$$\boldsymbol{\xi}(\boldsymbol{y}) : \boldsymbol{y} \to \boldsymbol{\xi}(\boldsymbol{y}) = \begin{pmatrix} -1 + \frac{1-y_2}{2}(y_1+1) \\ y_2 \end{pmatrix}$$

that transforms K_q to K_t. Its Jacobian

$$\det\left(\frac{\mathrm{D}\boldsymbol{\xi}}{\mathrm{D}\boldsymbol{y}}\right) = \frac{1-y_2}{2}$$

is positive except for the upper edge $y_2 = 1$ where it vanishes. In spite of this the standard Substitution Theorem can be applied, and (4.23) follows immediately. ∎

REMARK 4.3 Notice that the transformation $\boldsymbol{\xi}(\boldsymbol{y})$ produces an additional linear factor $(1 - y_2)/2$ behind the integral sign. Hence, one has to increase the order of integration in the y_2-direction by one. $\square$

4.3.2 Newton-Cotes quadrature

The idea behind the construction of the Newton-Cotes quadrature rules for the reference triangle is the same as it was for the one-dimensional reference domain K_a – summation of weighted function values on equidistributed integration points.

Let us consider an integer number $n \geq 2$. The $n(n+1)/2$ equidistributed integration points $\boldsymbol{\xi}_{n,k_1,k_2} = [\xi_{1,n,k_1,k_2}, \xi_{2,n,k_1,k_2}]$; $k_1 = 1, 2, \ldots, n$; $k_2 = 1, 2, \ldots, n + 1 - k_1$, can be chosen as

$$\begin{aligned} \xi_{1,n,k_1,k_2} &= -1 + (k_1 - 1)h_n, \\ \xi_{2,n,k_1,k_2} &= -1 + (k_2 - 1)h_n, \end{aligned} \tag{4.24}$$

where $h_n = 2/(n-1)$. The distribution of integration points is illustrated for $n = 2, 3, 5$ in Figure 4.1.

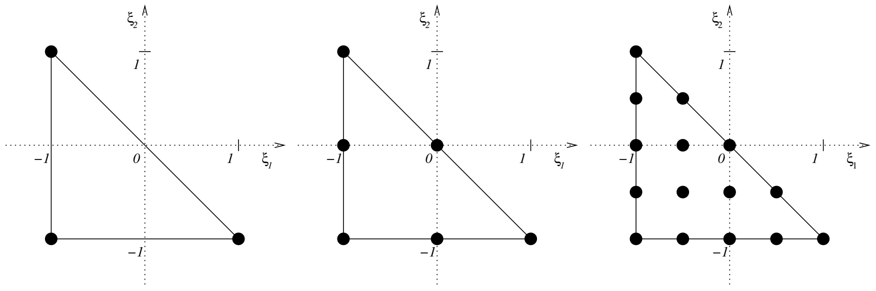

FIGURE 4.1: Newton-Cotes integration points for the reference triangle K_t, $n = 2, 3, 5$.

The $[n(n+1)/2]$-point Newton-Cotes quadrature rule for the reference triangle K_t reads

$$\int_{-1}^{1} \int_{-1}^{1-\xi_1} f(\xi_1, \xi_2)\,\mathrm{d}\xi_2\,\mathrm{d}\xi_1 \approx \sum_{k_1=1}^{n} \sum_{k_2=1}^{n+1-k_1} w_{n,k_1,k_2} f(\xi_{1,n,k_1,k_2}, \xi_{2,n,k_1,k_2}) \tag{4.25}$$

with the coordinates of the integration points defined in (4.24).

Thus, we have $n(n+1)/2$ unknown integration weights w_{n,k_1,k_2} which can be computed by inserting $n(n+1)/2$ linearly independent polynomials f_k of the order k, $k = 0, 1, \ldots, n-1$, into (4.25) and by solving the resulting linear system. One can use, e.g., the H^1-hierarchic shape functions defined in Paragraph 2.2.3. Notice that the $[n(n+1)/2]$-point Newton-Cotes quadrature rule has the order of accuracy $n-1$.

REMARK 4.4 Conditioning of the linear system for the integration weights can be improved by a choice of more sophisticated nodal points. Instead of the equidistributed points shown in Figure 4.1, one may want to use the *Gauss-Lobatto* points (see Figure 1.2). $\square$

4.3.3 Gauss quadrature

We have seen already in Paragraph 4.2.2 that although the Gauss quadrature rules in 2D are designed following the same principles as in one spatial dimension, the calculation of the corresponding integration points and weights is much more difficult.

The fundamental equation for the construction of the integration points and weights for the reference triangle K_t reads

$$\int_{-1}^{1}\int_{-1}^{1-\xi_1} f(\xi_1,\xi_2)\,\mathrm{d}\xi_2\,\mathrm{d}\xi_1 \approx \sum_{k=1}^{m} w_k f(\xi_{1,k},\xi_{2,k}), \tag{4.26}$$

where m denotes the number of integration points. Each point is characterized by three unknowns: w_k, $\xi_{1,k}$ and $\xi_{2,k}$. In order to illustrate complications we have to face when determining their values, let us briefly analyze the formula for complete polynomials of the order $p = 1$ and $p = 2$.

The first-order polynomial $f(\xi_1, \xi_2) = a_0 + a_1\xi_1 + a_2\xi_2$ does not cause any difficulties yet. In this case $m = 1$ and we easily obtain three equations,

$$w_1 = 2, w_1\xi_{1,1} = -2/3, w_1\xi_{2,1} = -2/3, \tag{4.27}$$

the solution of which is

$$w_1 = 2, \xi_{1,1} = -1/3, \xi_{2,1} = -1/3. \tag{4.28}$$

The number of terms of complete quadratic polynomial is 6 and the minimum number of the Gaussian points is 2 (leading to 6 unknowns). The system of equations for the coordinates and weights now reads

$$\begin{aligned} w_1 + w_2 &= 2, \\ w_1\xi_{1,1} + w_1\xi_{1,2} &= -2/3, \end{aligned} \tag{4.29}$$

$$\begin{aligned} w_1\xi_{2,1} + w_1\xi_{2,2} &= -2/3, \\ w_1\xi_{1,1}\xi_{2,1} + w_1\xi_{1,2}\xi_{2,2} &= 0, \\ w_1\xi_{1,1}^2 + w_1\xi_{1,2}^2 &= 2/3, \\ w_1\xi_{2,1}^2 + w_1\xi_{2,2}^2 &= 2/3. \end{aligned}$$

These equations, unfortunately, are not independent and an unambiguous solution does not exist. That is why at least three points instead of two have to be chosen to ensure that (4.26) is valid.

The same holds for polynomials of higher degrees. Moreover, the number n of terms of a complete polynomial of order p is often not divisible by three. For $p = 3$, for example, $n = (p+1)(p+2)/2 = 10$, so that we have to use at least four Gaussian points.

Difficulties of this kind are common for problems associated with solution of systems of nonlinear algebraic equations, and make the calculation of the integration points and weights for higher values of p practically infeasible by standard methods or mathematical software.

Lyness [133] proposed a sophisticated algorithm based on determining the points and weights within an *equilateral triangle* in polar coordinates, taking advantage of its multiple symmetries. The algorithm provides the minimum number of Gaussian points for any polynomial order p and strongly reduces the size of the nonlinear system. Dunavant [74] extended the algorithm and calculated the points and weights up to the order $p = 20$ (some weights being negative and some points lying outside the triangle). Position of integration points and values of the weights corresponding to other triangles can be obtained by a simple affine transformation.

The fact that some of the weights are negative means that the stability of the quadrature will tend to decrease when integrating oscillatory functions whose polynomial behavior exceeds the order of accuracy of the quadrature formulae. In this case the schemes still can be used, but one has to combine them with spatial refinements of the reference element. If oscillations (or other excessive nonlinearities) in the integrated functions are expected, an application of *adaptive formulae* that compare results from several refinement levels may be a good idea.

4.3.4 Tables of Gauss integration points and weights

Dunavant [74] provides an overview of minimum numbers of quadrature points for Gaussian quadrature over triangles in Table 4.41.

Tables 4.42 – 4.48 give an overview of Gaussian integration points and weights calculated in [133, 74] and transformed to the reference triangle K_t. Additional Gauss quadrature rules up to the order $p = 20$ can be found on the CD-ROM included with this book. A survey of numerical quadrature over triangles can be found in [132].

TABLE 4.41: Minimum numbers of quadrature points for the Gauss quadrature over triangles.

Polyn. order	**Known min. num. of points**	**Predicted min. num. of points**	**Achieved num. of points**
1	1		1
2	3		3
3	4		4
4	6		6
5	7		7
6	12		12
7	13		13
8	16		16
9	19		19
10	24		25
11		27	27
12		33	33
13		36	37
14		40	42
15		45	48
16		51	52
17		55	61
18		63	70
19		67	73
20		73	79

TABLE 4.42: Gauss quadrature on K_t, order $p = 1$.

Point #	**ξ_1-Coordinate**	**ξ_2-Coordinate**	**Weight**
1.	-0.33333 33333 33333	-0.33333 33333 33333	2.00000 00000 00000

TABLE 4.43: Gauss quadrature on K_t, order $p = 2$.

Point #	**ξ_1-Coordinate**	**ξ_2-Coordinate**	**Weight**
1.	-0.66666 66666 66667	-0.66666 66666 66667	0.66666 66666 66667
2.	-0.66666 66666 66667	0.33333 33333 33333	0.66666 66666 66667
3.	0.33333 33333 33333	-0.66666 66666 66667	0.66666 66666 66667

TABLE 4.44: Gauss quadrature on K_t, order $p = 3$.

Point #	**ξ_1-Coordinate**	**ξ_2-Coordinate**	**Weight**
1.	-0.33333 33333 33333	-0.33333 33333 33333	-1.12500 00000 00000
2.	-0.60000 00000 00000	-0.60000 00000 00000	1.04166 66666 66667
3.	-0.60000 00000 00000	0.20000 00000 00000	1.04166 66666 66667
4.	0.20000 00000 00000	-0.60000 00000 00000	1.04166 66666 66667

TABLE 4.45: Gauss quadrature on K_t, order $p = 4$.

Point #	**ξ_1-Coordinate**	**ξ_2-Coordinate**	**Weight**
1.	-0.10810 30181 68070	-0.10810 30181 68070	0.44676 31793 56022
2.	-0.10810 30181 68070	-0.78379 39636 63860	0.44676 31793 56022
3.	-0.78379 39636 63860	-0.10810 30181 68070	0.44676 31793 56022
4.	-0.81684 75729 80458	-0.81684 75729 80458	0.21990 34873 10644
5.	-0.81684 75729 80458	0.63369 51459 60918	0.21990 34873 10644
6.	0.63369 51459 60918	-0.81684 75729 80458	0.21990 34873 10644

TABLE 4.46: Gauss quadrature on K_t, order $p = 5$.

Point #	ξ_1-Coordinate	ξ_2-Coordinate	Weight
1.	-0.33333 33333 33333	-0.33333 33333 33333	0.45000 00000 00000
2.	-0.05971 58717 89770	-0.05971 58717 89770	0.26478 83055 77012
3.	-0.05971 58717 89770	-0.88056 82564 20460	0.26478 83055 77012
4.	-0.88056 82564 20460	-0.05971 58717 89770	0.26478 83055 77012
5.	-0.79742 69853 53088	-0.79742 69853 53088	0.25187 83610 89654
6.	-0.79742 69853 53088	0.59485 39707 06174	0.25187 83610 89654
7.	0.59485 39707 06174	-0.79742 69853 53088	0.25187 83610 89654

TABLE 4.47: Gauss quadrature on K_t, order $p = 6$.

Point #	ξ_1-Coordinate	ξ_2-Coordinate	Weight
1.	-0.50142 65096 58180	-0.50142 65096 58180	0.23357 25514 52758
2.	-0.50142 65096 58180	0.00285 30193 16358	0.23357 25514 52758
3.	0.00285 30193 16358	-0.50142 65096 58180	0.23357 25514 52758
4.	-0.87382 19710 16996	-0.87382 19710 16996	0.10168 98127 40414
5.	-0.87382 19710 16996	0.74764 39420 33992	0.10168 98127 40414
6.	0.74764 39420 33992	-0.87382 19710 16996	0.10168 98127 40414
7.	-0.37929 50979 32432	0.27300 49982 42798	0.16570 21512 36748
8.	0.27300 49982 42798	-0.89370 99003 10366	0.16570 21512 36748
9.	-0.89370 99003 10366	-0.37929 50979 32432	0.16570 21512 36748
10.	-0.37929 50979 32432	-0.89370 99003 10366	0.16570 21512 36748
11.	0.27300 49982 42798	-0.37929 50979 32432	0.16570 21512 36748
12.	-0.89370 99003 10366	0.27300 49982 42798	0.16570 21512 36748

TABLE 4.48: Gauss quadrature on K_t, order $p = 7$. See the companion CD-ROM for additional Gauss quadrature rules up to the order $p = 20$.

Point #	ξ_1-Coordinate	ξ_2-Coordinate	Weight
1.	-0.33333 33333 33333	-0.33333 33333 33333	-0.29914 00889 35364
2.	-0.47930 80678 41920	-0.47930 80678 41920	0.35123 05148 66416
3.	-0.47930 80678 41920	-0.04138 38643 16160	0.35123 05148 66416
4.	-0.04138 38643 16160	-0.47930 80678 41920	0.35123 05148 66416
5.	-0.86973 97941 95568	-0.86973 97941 95568	0.10669 44712 17676
6.	-0.86973 97941 95568	0.73947 95883 91136	0.10669 44712 17676
7.	0.73947 95883 91136	-0.86973 97941 95568	0.10669 44712 17676
8.	-0.37426 90079 90252	0.27688 83771 39620	0.15422 75217 80514
9.	0.27688 83771 39620	-0.90261 93691 49368	0.15422 75217 80514
10.	-0.90261 93691 49368	-0.37426 90079 90252	0.15422 75217 80514
11.	-0.37426 90079 90252	-0.90261 93691 49368	0.15422 75217 80514
12.	0.27688 83771 39620	-0.37426 90079 90252	0.15422 75217 80514
13.	-0.90261 93691 49368	0.27688 83771 39620	0.15422 75217 80514

4.4 Reference brick K_B

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

4.4.1 Composite Gauss quadrature

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

$$\int_{K_a} f(\xi)\,\mathrm{d}\xi \approx \sum_{i=1}^{M_a} w_{g_a,i} f(y_{g_a,i}), \tag{4.30}$$

where $y_{g_a,i}, w_{g_a,i}$ are Gauss integration points and weights corresponding to the one-dimensional reference domain K_a introduced in Paragraph 4.1.4, integrate exactly all polynomials of the order p and lower. It is easy to see that the formula

$$\int_{K_a^3} g(\xi_1, \xi_2, \xi_3)\,\mathrm{d}\xi_1\,\mathrm{d}\xi_2\,\mathrm{d}\xi_3 \approx \sum_{i=1}^{M_a}\sum_{j=1}^{M_a}\sum_{k=1}^{M_a} w_{g_a,i} w_{g_a,j} w_{g_a,k} g(y_{g_a,i}, y_{g_a,j}, y_{g_a,k}) \tag{4.31}$$

has the order of accuracy p for functions of three independent variables ξ_1, ξ_2, ξ_3 defined in K_B. The formula (4.31) can easily be generalized to polynomials with different directional orders of approximation.

Similarly as for quadrilaterals, more efficient formulae can be used when integrating *complete polynomials* of the order p (with generally $n = (p+1)(p+2)(p+3)/6$ nonzero terms). The formulae that we are going to introduce in the next paragraph were derived in [75].

4.4.2 Economical Gauss quadrature

The construction of economical Gauss quadrature rules for complete polynomials starts from the manifold symmetry of the reference brick K_B (nine symmetry planes). Let us first have a look at the integral

$$\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} \xi_1^j \xi_2^k \xi_3^l \,\mathrm{d}\xi_3\,\mathrm{d}\xi_2\,\mathrm{d}\xi_1, \tag{4.32}$$

where j, k and l are positive integers. As long as j, k or l is odd, the integral is equal to zero. Moreover, due to the symmetry with respect to various planes it is sufficient to consider terms in which $j \geq k$ and $k \geq l$ only.

4.4.3 Tables of Gauss integration points and weights

Dunavant [75] provides an overview of minimum numbers of quadrature points for odd polynomial orders in Table 4.49.

TABLE 4.49: Minimum numbers of quadrature points for the Gaussian quadrature over bricks.

Polyn. order	**Min. num. of nonsym. points**	**Min. num. of sym. points**	**Achieved num. of sym. points**
1	1	1	1
3	4	6	6
5	10	14	14
7	20	27	27
9	35	52	53
11	56	77	89
13	84	127	151
15	120	175	235
17	165	253	307
19	220	333	435

Dunavant divides the integration points into seven groups with different numbers of unknowns and tests to determine the best choice. The results obtained by solving the corresponding systems of nonlinear equations are given in Tables 4.50 – 4.53. A list of economical Gauss quadrature rules up to the order $p = 19$ with five more decimal digits can be found on the CD-ROM included with this book.

TABLE 4.50: Gauss quadrature on K_B, order $p = 0, 1$.

Point #	**ξ_1-Coordinate**	**ξ_2-Coordinate**	**ξ_3-Coordinate**	**Weight**
1.	0.00000 00000	0.00000 00000	0.00000 00000	8.00000 00000

TABLE 4.51: Gauss quadrature on K_B, order $p = 2, 3$.

Point #	**ξ_1-Coordinate**	**ξ_2-Coordinate**	**ξ_3-Coordinate**	**Weight**
1.	1.00000 00000	0.00000 00000	0.00000 00000	1.33333 33333
2.	-1.00000 00000	0.00000 00000	0.00000 00000	1.33333 33333
3.	0.00000 00000	1.00000 00000	0.00000 00000	1.33333 33333
4.	0.00000 00000	-1.00000 00000	0.00000 00000	1.33333 33333
5.	0.00000 00000	0.00000 00000	1.00000 00000	1.33333 33333
6.	0.00000 00000	0.00000 00000	-1.00000 00000	1.33333 33333

TABLE 4.52: Gauss quadrature on K_B, order $p = 4, 5$.

Point #	ξ_1-Coordinate	ξ_2-Coordinate	ξ_3-Coordinate	Weight
1.	0.79582 24257	0.00000 00000	0.00000 00000	0.88642 65927
2.	-0.79582 24257	0.00000 00000	0.00000 00000	0.88642 65927
3.	0.00000 00000	0.79582 24257	0.00000 00000	0.88642 65927
4.	0.00000 00000	-0.79582 24257	0.00000 00000	0.88642 65927
5.	0.00000 00000	0.00000 00000	0.79582 24257	0.88642 65927
6.	0.00000 00000	0.00000 00000	-0.79582 24257	0.88642 65927
7.	0.75878 69106	0.75878 69106	0.75878 69106	0.33518 00554
8.	0.75878 69106	-0.75878 69106	0.75878 69106	0.33518 00554
9.	0.75878 69106	0.75878 69106	-0.75878 69106	0.33518 00554
10.	0.75878 69106	-0.75878 69106	-0.75878 69106	0.33518 00554
11.	-0.75878 69106	0.75878 69106	0.75878 69106	0.33518 00554
12.	-0.75878 69106	-0.75878 69106	0.75878 69106	0.33518 00554
13.	-0.75878 69106	0.75878 69106	-0.75878 69106	0.33518 00554
14.	-0.75878 69106	-0.75878 69106	-0.75878 69106	0.33518 00554

TABLE 4.53: Gauss quadrature on K_B, order $p = 6, 7$. See the companion CD-ROM for additional economical Gauss quadrature rules up to the order $p = 19$.

Point #	ξ_1-Coordinate	ξ_2-Coordinate	ξ_3-Coordinate	Weight
1.	0.00000 00000	0.00000 00000	0.00000 00000	0.78807 34827
2.	0.84841 80014	0.00000 00000	0.00000 00000	0.49936 90023
3.	-0.84841 80014	0.00000 00000	0.00000 00000	0.49936 90023
4.	0.00000 00000	0.84841 80014	0.00000 00000	0.49936 90023
5.	0.00000 00000	-0.84841 80014	0.00000 00000	0.49936 90023
6.	0.00000 00000	0.00000 00000	0.84841 80014	0.49936 90023
7.	0.00000 00000	0.00000 00000	-0.84841 80014	0.49936 90023
8.	0.65281 64721	0.65281 64721	0.65281 64721	0.47850 84494
9.	0.65281 64721	-0.65281 64721	0.65281 64721	0.47850 84494
10.	0.65281 64721	0.65281 64721	-0.65281 64721	0.47850 84494
11.	0.65281 64721	-0.65281 64721	-0.65281 64721	0.47850 84494
12.	-0.65281 64721	0.65281 64721	0.65281 64721	0.47850 84494
13.	-0.65281 64721	-0.65281 64721	0.65281 64721	0.47850 84494
14.	-0.65281 64721	0.65281 64721	-0.65281 64721	0.47850 84494
15.	-0.65281 64721	-0.65281 64721	-0.65281 64721	0.47850 84494
16.	0.00000 00000	1.10641 28986	1.10641 28986	0.03230 37423
17.	0.00000 00000	-1.10641 28986	1.10641 28986	0.03230 37423
18.	0.00000 00000	1.10641 28986	-1.10641 28986	0.03230 37423
19.	0.00000 00000	-1.10641 28986	-1.10641 28986	0.03230 37423
20.	1.10641 28986	0.00000 00000	1.10641 28986	0.03230 37423
21.	-1.10641 28986	0.00000 00000	1.10641 28986	0.03230 37423
22.	1.10641 28986	0.00000 00000	-1.10641 28986	0.03230 37423
23.	-1.10641 28986	0.00000 00000	-1.10641 28986	0.03230 37423
24.	1.10641 28986	1.10641 28986	0.00000 00000	0.03230 37423
25.	-1.10641 28986	1.10641 28986	0.00000 00000	0.03230 37423
26.	1.10641 28986	-1.10641 28986	0.00000 00000	0.03230 37423
27.	-1.10641 28986	-1.10641 28986	0.65281 64721	0.03230 37423

4.5 Reference tetrahedron K_T

In this paragraph we introduce several higher-order quadrature schemes for the reference tetrahedral domain K_T. Of course we are mostly interested in economical Gauss quadrature rules, but for higher polynomial orders ($p > 9$) their design is extremely difficult. In Paragraph 4.5.2 we will present the best selection of optimal and suboptimal Gauss quadrature rules that we found.

Frequently used product rules, which are based on a degenerate mapping from K_B to K_T and the Substitution Theorem, will be described in Paragraph 4.5.1. This time we will not address explicitly the Newton-Cotes quadrature formulae which can be constructed analogously as in the triangular case (Paragraph 4.3.2).

4.5.1 Translation of quadrature to the reference brick K_B

This approach is relevant for extremely high orders of accuracy where better quadrature rules are not available. One also can use it as a quick fix for debugging purposes or when for another reason the efficiency of quadrature procedures is not important.

Transforming the integrated function to the reference brick K_B and applying either the composite or economical Gauss quadrature formulae (Paragraphs 4.4.1 and 4.4.2), one ends up with simple quadrature rules for K_T. The idea of the degenerate transform is the same as it was in 2D with the reference quadrilateral K_q and the reference triangle K_t.

PROPOSITION 4.2
Let $g(\boldsymbol{\xi})$ be a continuous bounded function defined on the reference tetrahedron K_T. Then

$$\int_{K_T} g(\boldsymbol{\xi})\, d\boldsymbol{\xi} = \int_{K_B} \frac{(1-y_2)(1-y_3)^2}{8} g \begin{pmatrix} -1 + \frac{(1-y_2)(1-y_3)}{4}(y_1+1) \\ -1 + \frac{(1+y_2)(1-y_3)}{2} \\ y_3 \end{pmatrix} d\boldsymbol{y}. \tag{4.33}$$

PROOF Consider the (degenerate) mapping

$$\boldsymbol{\xi}(\boldsymbol{y}) : \boldsymbol{y} \rightarrow \boldsymbol{\xi}(\boldsymbol{y}) = \begin{pmatrix} -1 + \frac{(1-y_2)(1-y_3)}{4}(y_1+1) \\ -1 + \frac{(1+y_2)(1-y_3)}{2} \\ y_3 \end{pmatrix}$$

that transforms K_B to K_T. Its Jacobian

$$\det\left(\frac{D\boldsymbol{\xi}}{D\boldsymbol{y}}\right) = \frac{(1-y_2)(1-y_3)^2}{8}$$

is positive except for faces $y_2 = 1, y_3 = 1$ where it vanishes. Hence, (4.33) immediately follows from the Substitution Theorem. $\square$

REMARK 4.5 Notice that the transformation $\boldsymbol{\xi}(\boldsymbol{y})$ produces an additional polynomial factor $(1-y_2)(1-y_3)^2/2$. One has to increase the order of integration by one and two in the y_2- and y_3-direction, respectively. $\square$

4.5.2 Economical Gauss quadrature

Economical Gauss quadrature formulae for complete polynomials of the order $p \le 9$ (with generally $n = (p+1)(p+2)(p+3)/6$ nonzero terms) have been derived for tetrahedra in [123] and others.

The construction of Gauss quadrature points and weights is based on topological symmetries within the tetrahedron with vertices $w_1 = [0,0,0]$, $w_2 = [1,0,0]$, $w_3 = [0,1,0]$, $w_4 = [0,0,1]$. We transform the results found in the literature to the reference tetrahedron K_T. The transform $\boldsymbol{\xi} = \boldsymbol{\xi}(\boldsymbol{y})$ is given by

$$\xi_1 = 2y_1 - 1, \quad \xi_2 = 2y_2 - 1, \quad \xi_3 = 2y_3 - 1, \tag{4.34}$$

where $\boldsymbol{y} = (y_1, y_2, y_3)$ and $\boldsymbol{\xi} = (\xi_1, \xi_2, \xi_3)$ stand for the original and new coordinates, respectively. The corresponding Jacobian is constant,

$$\det\left(\frac{D\boldsymbol{\xi}}{D\boldsymbol{y}}\right) = 8.$$

The quadrature rules for $p \le 7$ are presented in Tables 4.55 – 4.61.

The complexity of the problem increases rapidly as $p > 9$. For odd $p \ge 11$ one may construct suboptimal Gaussian rules based on combinatorial formulae of variable order for n-simplices derived in [101]. We evaluated these formulae by Mathematica using a notebook that the reader can utilize to obtain quadrature rules for high odd p's. The parameters `t1` and `t2` at its end are set to the polynomial orders for which we evaluate the integration points and weights. For example, for polynomial orders 1, 3, 5, ..., 21 we set `t1 := 1` and `t2 := 11`, i.e., `s = 2t - 1`, where `s` is the polynomial order and `t` is a counter from `t1` to `t2`. The notebook as well as the tabulated data (with five more decimal digits) are provided on the CD-ROM included with this book.

```
(************** Content-type: application/mathematica **************
                     CreatedBy='Mathematica 4.2'
                    Mathematica-Compatible Notebook*)
(*CacheID: 232*)
(*NotebookFileLineBreakTest NotebookFileLineBreakTest*)
(*NotebookOptionsPosition[      2916,         75]*)
(*NotebookOutlinePosition[      3587,         98]*)
(*  CellTagsIndexPosition[      3543,         94]*)
(*WindowFrame->Normal*)

Notebook[{
Cell[BoxData[
    \(\(\(\[IndentingNewLine]\)\(<<
      DiscreteMath`Combinatorica`\[IndentingNewLine]
      Do[m = 0; \[IndentingNewLine]\
      Do[s = t - 1; \ i = j - 1; \
        w = 8 \((\(-1\))\)^\((i)\)\ \(\((2  s + 4 - 2  i)\)^\((2  s + 1)\)/
                2^\((2  s)\)\)/\((\(i!\)\ \(\((2  s + 4 - i)\)!\))\)\ ; \
        ww = Abs[w]; b = s - i; d = 2  s + 4 - 2  i;
        f = \(\(\((3 + b)\)!\)/\(3!\)\)/\(b!\); \
        sezn = Compositions[b, 4]; \ \[IndentingNewLine]Do[
          a1 = sezn[\([k, 2]\)]; a2 = sezn[\([k, 3]\)];
          a3 = sezn[\([k, 4]\)];
          m = m + 1; \[IndentingNewLine]b1 = 2 \((\((2  a1 + 1)\)/d)\) - 1;
          b2 = 2 \((\((2  a2 + 1)\)/d)\) - 1;
          b3 = 2 \((\((2  a3 + 1)\)/d)\) - 1; \[IndentingNewLine]g1 =
            SetAccuracy[b1, 16]; g2 = SetAccuracy[b2, 16];
          g3 = SetAccuracy[b3, 16]; w1 = SetAccuracy[w, 16]; \
          pp = "\<   \>";
          Print[m, pp, w1, pp, g1, pp, g2, pp, g3,
            pp], \[IndentingNewLine]{k, f}], {j, t}];
      Print["\<  \>"]; \[IndentingNewLine]Print["\<  \>"], {t, t1,
        t2}]\[IndentingNewLine]
    \)\)\)], "Input"]
},
FrontEndVersion->"4.2 for Microsoft Windows",
ScreenRectangle->{{0, 1280}, {0, 951}},
WindowSize->{508, 740},
WindowMargins->{{-1, Automatic}, {Automatic, 0}},
CellLabelAutoDelete->True]

(*******************************************************************
Cached data follows.  If you edit this Notebook file directly, not
using Mathematica, you must remove the line containing CacheID at
the top of  the file.  The cache data will then be recreated when
you save this file from within Mathematica.
*******************************************************************)

(*CellTagsOutline CellTagsIndex->{}*)
(*CellTagsIndex CellTagsIndex->{}*)
(*NotebookFileOutline Notebook[{
Cell[1754, 51, 1158, 22, 350, "Input"]}]*)

(*******************************************************************
End of Mathematica Notebook file.
*******************************************************************)
```

4.5.3 Tables of Gauss integration points and weights

Table 4.54 gives an overview of numbers of integration points for the Gaussian quadrature over tetrahedra.

TABLE 4.54: Minimum numbers of quadrature points for the Gauss quadrature over tetrahedra.

Polynomial order	Known minimum number of points	Achieved number of points
1	1	1
2	4	4
3	5	5
4	11	11
5	14	14
6	24	24
7	28	31
8	40	43
9	52	53
10	68	
11		126
12		
13		210
14		
15		330
16		
17		495
18		
19		715

TABLE 4.55: Gauss quadrature on K_T, order $p = 1$.

Point #	ξ_1-Coordinate	ξ_2-Coordinate	ξ_3-Coordinate	Weight
1.	-0.50000 00000	-0.50000 00000	-0.50000 00000	1.33333 33333

TABLE 4.56: Gauss quadrature on K_T, order $p = 2$.

Point #	ξ_1-Coordinate	ξ_2-Coordinate	ξ_3-Coordinate	Weight
1.	-0.72360 67977	-0.72360 67977	-0.72360 67977	0.33333 33333
2.	0.17082 03932	-0.72360 67977	-0.72360 67977	0.33333 33333
3.	-0.72360 67977	0.17082 03932	-0.72360 67977	0.33333 33333
4.	-0.72360 67977	-0.72360 67977	0.17082 03932	0.33333 33333

TABLE 4.57: Gauss quadrature on K_T, order $p = 3$.

Point #	ξ_1-Coordinate	ξ_2-Coordinate	ξ_3-Coordinate	Weight
1.	-0.50000 00000	-0.50000 00000	-0.50000 00000	-1.06666 66666
2.	-0.66666 66666	-0.66666 66666	-0.66666 66666	0.60000 00000
3.	-0.66666 66666	-0.66666 66666	0.00000 00000	0.60000 00000
4.	-0.66666 66666	0.00000 00000	-0.66666 66666	0.60000 00000
5.	0.00000 00000	-0.66666 66666	-0.66666 66666	0.60000 00000

TABLE 4.58: Gauss quadrature on K_T, order $p = 4$.

Point #	ξ_1-Coordinate	ξ_2-Coordinate	ξ_3-Coordinate	Weight
1.	-0.50000 00000	-0.50000 00000	-0.50000 00000	-0.10524 44444
2.	-0.85714 28571	-0.85714 28571	-0.85714 28571	0.06097 77777
3.	-0.85714 28571	-0.85714 28571	0.57142 85714	0.06097 77777
4.	-0.85714 28571	0.57142 85714	-0.85714 28571	0.06097 77777
5.	0.57142 85714	-0.85714 28571	-0.85714 28571	0.06097 77777
6.	-0.20119 28476	-0.20119 28476	-0.79880 71523	0.19911 11111
7.	-0.20119 28476	-0.79880 71523	-0.20119 28476	0.19911 11111
8.	-0.79880 71523	-0.20119 28476	-0.20119 28476	0.19911 11111
9.	-0.20119 28476	-0.79880 71523	-0.79880 71523	0.19911 11111
10.	-0.79880 71523	-0.20119 28476	-0.79880 71523	0.19911 11111
11.	-0.79880 71523	-0.79880 71523	-0.20119 28476	0.19911 11111

TABLE 4.59: Gauss quadrature on K_T, order $p = 5$.

Point #	ξ_1-Coordinate	ξ_2-Coordinate	ξ_3-Coordinate	Weight
1.	-0.81452 94993	-0.81452 94993	-0.81452 94993	0.09799 07241
2.	0.44358 84981	-0.81452 94993	-0.81452 94993	0.09799 07241
3.	-0.81452 94993	0.44358 84981	-0.81452 94993	0.09799 07241
4.	-0.81452 94993	-0.81452 94993	0.44358 84981	0.09799 07241
5.	-0.37822 81614	-0.37822 81614	-0.37822 81614	0.15025 05676
6.	-0.86531 55155	-0.37822 81614	-0.37822 81614	0.15025 05676
7.	-0.37822 81614	-0.86531 55155	-0.37822 81614	0.15025 05676
8.	-0.37822 81614	-0.37822 81614	-0.86531 55155	0.15025 05676
9.	-0.09100 74082	-0.09100 74082	-0.90899 25917	0.05672 80277
10.	-0.09100 74082	-0.90899 25917	-0.09100 74082	0.05672 80277
11.	-0.90899 25917	-0.09100 74082	-0.09100 74082	0.05672 80277
12.	-0.09100 74082	-0.90899 25917	-0.90899 25917	0.05672 80277
13.	-0.90899 25917	-0.09100 74082	-0.90899 25917	0.05672 80277
14.	-0.90899 25917	-0.90899 25917	-0.09100 74082	0.05672 80277

TABLE 4.60: Gauss quadrature on K_T, order $p = 6$.

Point #	ξ_1**-Coordinate**	ξ_2**-Coordinate**	ξ_3**-Coordinate**	**Weight**
1.	-0.57079 42574	-0.57079 42574	-0.57079 42574	0.05323 03336
2.	-0.28761 72275	-0.57079 42574	-0.57079 42574	0.05323 03336
3.	-0.57079 42574	-0.28761 72275	-0.57079 42574	0.05323 03336
4.	-0.57079 42574	-0.57079 42574	-0.28761 72275	0.05323 03336
5.	-0.91865 20829	-0.91865 20829	-0.91865 20829	0.01343 62814
6.	0.75595 62487	-0.91865 20829	-0.91865 20829	0.01343 62814
7.	-0.91865 20829	0.75595 62487	-0.91865 20829	0.01343 62814
8.	-0.91865 20829	-0.91865 20829	0.75595 62487	0.01343 62814
9.	-0.35532 42197	-0.35532 42197	-0.35532 42197	0.07380 95753
10.	-0.93402 73408	-0.35532 42197	-0.35532 42197	0.07380 95753
11.	-0.35532 42197	-0.93402 73408	-0.35532 42197	0.07380 95753
12.	-0.35532 42197	-0.35532 42197	-0.93402 73408	0.07380 95753
13.	-0.87267 79962	-0.87267 79962	-0.46065 53370	0.06428 57142
14.	-0.87267 79962	-0.46065 53370	-0.87267 79962	0.06428 57142
15.	-0.87267 79962	-0.87267 79962	0.20601 13295	0.06428 57142
16.	-0.87267 79962	0.20601 13295	-0.87267 79962	0.06428 57142
17.	-0.87267 79962	-0.46065 53370	0.20601 13295	0.06428 57142
18.	-0.87267 79962	0.20601 13295	-0.46065 53370	0.06428 57142
19.	-0.46065 53370	-0.87267 79962	-0.87267 79962	0.06428 57142
20.	-0.46065 53370	-0.87267 79962	0.20601 13295	0.06428 57142
21.	-0.46065 53370	0.20601 13295	-0.87267 79962	0.06428 57142
22.	0.20601 13295	-0.87267 79962	-0.46065 53370	0.06428 57142
23.	0.20601 13295	-0.87267 79962	-0.87267 79962	0.06428 57142
24.	0.20601 13295	-0.46065 53370	-0.87267 79962	0.06428 57142

TABLE 4.61: Gauss quadrature on K_T, order $p = 7$. See the companion CD-ROM for additional Gauss quadrature rules up to the order $p = 21$.

Point #	ξ_1**-Coordinate**	ξ_2**-Coordinate**	ξ_3**-Coordinate**	**Weight**
1.	0.00000 00000	0.00000 00000	-1.00000 00000	0.00776 01410
2.	0.00000 00000	-1.00000 00000	0.00000 00000	0.00776 01410
3.	-1.00000 00000	0.00000 00000	0.00000 00000	0.00776 01410
4.	-1.00000 00000	-1.00000 00000	0.00000 00000	0.00776 01410
5.	-1.00000 00000	0.00000 00000	-1.00000 00000	0.00776 01410
6.	0.00000 00000	-1.00000 00000	-1.00000 00000	0.00776 01410
7.	-0.50000 00000	-0.50000 00000	-0.50000 00000	0.14611 37877
8.	-0.84357 36153	-0.84357 36153	-0.84357 36153	0.08479 95321
9.	-0.84357 36153	-0.84357 36153	0.53072 08460	0.08479 95321
10.	-0.84357 36153	0.53072 08460	-0.84357 36153	0.08479 95321
11.	0.53072 08460	-0.84357 36153	-0.84357 36153	0.08479 95321
12.	-0.75631 35666	-0.75631 35666	-0.75631 35666	-0.50014 19209
13.	-0.75631 35666	-0.75631 35666	0.26894 07000	-0.50014 19209
14.	-0.75631 35666	0.26894 07000	-0.75631 35666	-0.50014 19209
15.	0.26894 07000	-0.75631 35666	-0.75631 35666	-0.50014 19209
16.	-0.33492 16711	-0.33492 16711	-0.33492 16711	0.03913 14021
17.	-0.33492 16711	-0.33492 16711	-0.99523 49866	0.03913 14021
18.	-0.33492 16711	-0.99523 49866	-0.33492 16711	0.03913 14021
19.	-0.99523 49866	-0.33492 16711	-0.33492 16711	0.03913 14021
20.	-0.80000 00000	-0.80000 00000	-0.60000 00000	0.22045 85537
21.	-0.80000 00000	-0.60000 00000	-0.80000 00000	0.22045 85537
22.	-0.80000 00000	-0.80000 00000	0.20000 00000	0.22045 85537
23.	-0.80000 00000	0.20000 00000	-0.80000 00000	0.22045 85537
24.	-0.80000 00000	-0.60000 00000	0.20000 00000	0.22045 85537
25.	-0.80000 00000	0.20000 00000	-0.60000 00000	0.22045 85537
26.	-0.60000 00000	-0.80000 00000	-0.80000 00000	0.22045 85537
27.	-0.60000 00000	-0.80000 00000	0.20000 00000	0.22045 85537
28.	-0.60000 00000	0.20000 00000	-0.80000 00000	0.22045 85537
29.	0.20000 00000	-0.80000 00000	-0.60000 00000	0.22045 85537
30.	0.20000 00000	-0.80000 00000	-0.80000 00000	0.22045 85537
31.	0.20000 00000	-0.60000 00000	-0.80000 00000	0.22045 85537

4.6 Reference prism K_P

Finally we will deal with numerical quadrature on the reference prism K_P. To the best of our knowledge, no economical Gauss quadrature rules related to complete higher-order polynomials for this type of domain are known. Therefore we have to stay with product formulae combining the triangular and one-dimensional Gauss quadrature rules.

4.6.1 Composite Gauss quadrature

Let the quadrature rule

$$\int_{K_a} f(\xi)\mathrm{d}\xi \approx \sum_{i=1}^{M_a} w_{g_a,i} f(y_{g_a,i}), \tag{4.35}$$

where $y_{g_a,i}, w_{g_a,i}$ are Gauss integration points and weights on the one-dimensional reference domain $K_a = (-1,1)$, integrate exactly all polynomials of the order p and lower. Further, let the quadrature rule

$$\int_{K_t} g(\xi_1,\xi_2)\,\mathrm{d}\xi_1\,\mathrm{d}\xi_2 \approx \sum_{i=1}^{M_t} w_{g_t,i}\, g(y_{g_t,1,i}, y_{g_t,2,i}), \tag{4.36}$$

where $Y_{g_t,i} = [y_{g_t,1,i}, y_{g_t,2,i}]$ are Gauss integration points and weights for the reference triangle K_t, integrate exactly all polynomials of the order p and lower. It is easy to see that the formula

$$\int_{K_a}\int_{K_t} h(\xi_1,\xi_2,\xi_3)\,\mathrm{d}\xi_1\,\mathrm{d}\xi_2\,\mathrm{d}\xi_3 \approx \sum_{i=1}^{M_a}\sum_{j=1}^{M_t} w_{g_a,i} w_{g_t,j}\, h(y_{g_t,1,j}, y_{g_t,2,j}, y_{g_a,i}) \tag{4.37}$$

integrates exactly all polynomials (of three independent variables ξ_1,ξ_2,ξ_3) of the order p and lower on the reference prism K_P.

Notice that we can also use other higher-order quadrature rules for the reference domains K_a and K_t in order to produce composite quadrature rules for the reference prism K_P in the way presented.

Chapter 5

Numerical solution of finite element equations

Discretization of the boundary or initial-boundary value problem means the approximation of the original problem by a finite dimensional problem. This resulting problem is, however, nonlinear if the original problem was nonlinear. In the case of an elliptic problem, the necessary linearization can be applied to the original problem or to the finite dimensional problem. In both cases we finally obtain a system of *linear algebraic equations*. We are concerned with such real-valued systems in this chapter. Most methods presented can be applied to complex-valued systems, too, after proper modification.

The resulting system is usually *sparse*, i.e., its matrix has a very small number of nonzero entries. It is advantageous to take this special property of the matrix into account when solving the system since it can yield storage as well as computer time savings. On the other hand, the order of such a matrix may be even several million and the matrix and the system are called *large*. In some cases, the resulting matrix is *symmetric* and, moreover, *positive definite*. This feature as well as some other particular properties of the matrix of the system can also be utilized when the system is solved. (Note that if the matrix of the system is complex-valued then the property corresponding to the symmetry of the real-valued matrix is that the matrix be *Hermitian*.)

Numerical methods for solving linear algebraic systems can be split into two large groups, namely the *direct* and the *iterative methods*. Direct methods yield – if all computations were carried out without roundoff – the true solution of the system after a finite number of arithmetic operations that is known in advance. They are discussed in Section 5.1. Iterative methods start with some initial approximation (initial guess) to the solution and construct a sequence of approximations that converges to the true solution. Some basic methods of this kind are presented in Section 5.2. An important part of the analysis of convergence is then also the choice of efficient *stopping criteria*.

If the original problem is not elliptic but time-dependent, it can be first discretized only in the space variable(s). This approach, called the *method of lines*, is presented briefly in Paragraph 5.4.1. It results in an *initial value problem for a system of ordinary differential equations*, i.e., a problem that must be further discretized in the time variable in the next step. This is carried out by methods for approximate solving initial value problems. The process mentioned also implicitly includes solving linear algebraic equations and is the subject of Section 5.4.

We focus on some typical and frequently used methods and their available software implementation in this chapter. In general, we refer to the websites `gams.nist.gov` (Guide to Available Mathematical Software of the National Institute of Standards and Technology, U.S.A.) or `netlib.bell-labs.com` (Repository of Mathematical Software) with several mirrors (e.g., `netlib.no`) where there is a very comprehensive list of related software. Moreover, we also refer, e.g., to books [10, 24, 37, 158] with the corresponding software available on the web.

Some sections of this chapter are of an introductory nature only. It is, for example, hard to explain special elimination software for solving sparse linear systems without recalling the Gaussian elimination in its general form.

5.1 Direct methods for linear algebraic equations

Most direct methods for the solution of linear algebraic equations are based on the well-known principle of the Gaussian elimination or the algorithmically equivalent matrix factorization procedure described briefly in Paragraph 5.1.1. There are some particular implementations of the general algorithm for systems of special properties or special forms shown in the rest of this section.

The conjugate gradient method is also a direct method but it is used in practice as an iterative method, cf. Paragraph 5.2.2.

5.1.1 Gaussian elimination and matrix factorization

Throughout Sections 5.1 to 5.3 we consider the system of n linear algebraic equations

$$Ax = b \tag{5.1}$$

with a real-valued nonsingular square matrix A, right-hand part b and unknown vector x. Most methods presented in what follows can, however, also be implemented with complex-valued quantities.

The aim of the Gaussian elimination can be described as finding a matrix factorization

$$A = LU, \tag{5.2}$$

where L is a lower and U an upper triangular matrix. The system (5.1) is then rewritten as

$$LUx = b$$

and solved in two steps,

$$Ly = b, \qquad Ux = y. \tag{5.3}$$

These systems are triangular and can be solved very easily, the first from top to bottom (*forward substitution*), the other one from bottom to top (*backsubstitution*), using altogether $O(n^2)$ arithmetic operations.

In the traditional Gaussian elimination, the matrix factorization is looked for in $n-1$ steps in an implicit way: If $a_{11} \neq 0$ the first row successively multiplied by a proper number is subtracted from rows 2 to n (including the right-hand part components) to replace the subdiagonal entries of the first column of the matrix A by zeros. If the entry in position $(2,2)$ in this new matrix is nonzero the second row successively multiplied by a proper number is subtracted from rows 3 to n, etc. Finally, we get just the upper triangular matrix U and the individual multipliers constructed form the lower triangular matrix L. The right-hand part b is replaced by the vector y. We thus have constructed the factorization (5.2) and, at the same time, solved the lower tringular system $Ly = b$. It remains to solve the upper triangular system $Ux = y$. The Gaussian elimination just described is algorithmically equivalent to the matrix factorization approach and both the methods are interchangeable in theoretical considerations.

We have not yet said what to do if, e.g., $a_{11} = 0$. However, let us first present formulae for the matrix factorization. The factorization is unique if some further condition is added, e.g., the condition that all the diagonal entries of L are 1's. These diagonal entries thus need not be stored. The version of the factorization that builds successively the columns of L follows but is not the only possibility.

$$\begin{aligned}
u_{11} &= a_{11},\\
l_{i1} &= a_{i1}/u_{11}, \quad i = 2,\dots,n,\\
u_{1r} &= a_{1r},\\
u_{ir} &= a_{ir} - \sum_{j=1}^{i-1} l_{ij}u_{jr}, \quad i = 2,\dots,r,\\
l_{ir} &= \frac{1}{u_{rr}}\left(a_{ir} - \sum_{j=1}^{r-1} l_{ij}u_{jr}\right), \quad i = r+1,\dots,n,\\
&\qquad\qquad\qquad\qquad \text{for} \quad r = 2,\dots,n.
\end{aligned} \tag{5.4}$$

Notice that it is not necessary to use an extra storage for the matrices L and U since they successively replace the entries of A that are not further needed. In general, we need $O(n^3)$ arithmetic operations to solve the system (5.1).

The procedure (5.4) fails if the entry u_{rr} we want to divide by is zero. The easy solution is to interchange row r with some row that follows, i.e., a row between $r+1$ and n. It can be shown that if the matrix A is nonsingular there exists such a row with a nonzero entry in column r.

Moreover, the accumulation of roundoff error depends on the *condition number* of the matrix A and can be minimized [205] if we choose, for the elimination in step r, such a row among rows $r, \ldots, n$, say row k, whose entry in column r is maximal in magnitude. Such an entry is called the *pivot* and the procedure is called the *partial pivoting.* It apparently requires $O(n^2)$ additional arithmetic operations if applied in each step of the factorization. The factorization procedure can equivalently be partially pivoted looking for the maximal in magnitude pivot in some column of row r and interchanging the related columns. In addition, *complete pivoting* can be carried out in such a way that the pivot is looked for in the whole submatrix consisting of rows as well as columns $r, \ldots, n$. This complete pivoting, however, requires $O(n^3)$ arithmetic operations and is used rather rarely. In no case is any interchange of rows and/or columns carried out in the storage. It is sufficient to keep the row and column index permutations in the corresponding integer vectors.

For algorithmic reasons, the factorization (5.2) is often replaced by the factorization

$$A = LDU, \tag{5.5}$$

where D is a diagonal matrix. We have to solve a system with a diagonal matrix in addition to the systems (5.3) but solving a diagonal system needs only $O(n)$ arithmetic operations.

Moreover, if the matrix A is symmetric the factorization (5.2) can have the form (*Choleski factorization*)

$$A = LL^T \quad \text{or} \quad A = LDL^T, \tag{5.6}$$

where T denotes the transpose. We must, however, drop the requirement that the matrix L has 1's on its diagonal. We can save almost one half of the arithmetic operations in this way. But if the real-valued matrix A is not positive definite this factorization may lead to complex-valued entries of L. No pivoting can be applied since it would destroy the symmetry of the matrix.

Notice that as soon as we have computed the factorization (5.2) we can use it for solving the system (5.1) with more than one right-hand part via the equations (5.3) whose solution requires $O(n^2)$ arithmetic operations only.

No matter what method is used, the numerical solution of a linear algebraic system is, in general, more or less influenced or even destroyed by roundoff error that may accumulate in the course of the computation. The *backward analysis* [205] is used to study the error caused by roundoff in algebraic processes.

If necessary it is usually feasible to improve the computed solution x_0 of the system (5.1) by a simple iterative process requiring $O(n^2)$ arithmetic operations in each step [69]. Denote by $r_0 = b - Ax_0$ the *residual* of the system. If we could solve the system

$$Az = r_0 \tag{5.7}$$

exactly then the vector $x_0 + z$ would be the true solution of the system (5.1) since $A(x_0 + z) = Ax_0 + Az = Ax_0 + r_0 = b$.

Let us solve the system (5.7) numerically with the help of the same factorization (5.2) we used for solving the system (5.1). We need $O(n^2)$ arithmetic operations only, namely for solving the triangular systems (5.3). Denote by z_0 the computed solution of the system (5.7). If the loss of accuracy is not fatal the vector $x_1 = x_0 + z_0$ is an improved and more accurate solution of the system (5.1). We can continue this procedure as long as the norm of the residual decreases.

In general, we start with the initial approximation x_0 and compute successively

$$x_k = x_{k-1} + z_{k-1},$$

where z_{k-1} is the numerically computed solution of the system $Az = r_{k-1}$ and

$$r_{k-1} = b - Ax_{k-1} \tag{5.8}$$

is the numerically computed residual. Since we expect components of the residual r_{k-1} to be small and since they are computed as a difference of the corresponding components of the vectors b and Ax_{k-1} of almost the same magnitude it is sometimes recommended to calculate the residual (5.8) more precisely (cf. [69]). If the two or three iteration steps just described are not enough to improve the solution it usually means that the accumulation of roundoff error is fatal (the matrix A of the system is *ill-conditioned*) and the iterative procedure described cannot be efficient. Notice, however, that the error of the solution of the system (5.1) may be large even if the corresponding residual is small.

If the matrix A of the system (5.1) is symmetric positive definite, i.e., if

$$(Aw, w) > 0$$

for any nonzero vector w, then no pivoting can improve the accumulation of roundoff error [205] and, moreover, the Gaussian elimination can be carried out in an arbitrary order of rows and columns. (Naturally, any permutation of rows should have been accompanied by the same permutation of columns and vice versa. Otherwise the matrix would lose its symmetry.)

Further, if the matrix A of the system (5.1) is symmetric positive definite (or Hermitian positive definite in the complex-valued case) several iterative methods of Section 5.2 can be used to obtain the solution of the system in a very efficient way.

5.1.2 Banded systems

In the finite element discretization of 1D problems, the resulting matrix of the linear algebraic system is often symmetric and *tridiagonal.* This section is, in general, devoted to banded systems (systems with a banded matrix). The matrix is called *banded of bandwidth* $2m + 1$ if

$$a_{ij} = 0 \quad \text{for} \quad |i - j| > m. \tag{5.9}$$

This definition of a bandmatrix certainly does not exclude the situation when there are some zero entries inside the band.

A special case is a diagonal matrix with $m = 0$. To solve the system (5.1) with a diagonal matrix is a particularly easy task: it apparently requires $O(n)$ arithmetic operations only. For tridiagonal matrices (with $m = 1$), there is a straightforward modification of the Gaussian elimination (matrix factorization) that operates on nonzero entries of the matrix only and it only stores these nonzero entries concentrated on the three parallel diagonals of the matrix A as three vectors of n components and, in addition, two auxiliary vectors α and β. Such a special elimination procedure is usually called the *double sweep method*, requires $O(n)$ arithmetic operations only (as compared with $O(n^3)$ operations needed in general), and consists of the evaluation of the formulae

$$\begin{aligned}
\alpha_1 &= -a_{12}/a_{11}, \quad \beta_1 = b_1/a_{11}, \\
\alpha_i &= -\frac{a_{i,i+1}}{a_{i,i-1}\alpha_{i-1} + a_{ii}}, \quad i = 2, \ldots, n-1, \\
\beta_i &= \frac{b_i - a_{i,i-1}\beta_{i-1}}{a_{i,i-1}\alpha_{i-1} + a_{ii}}, \quad i = 2, \ldots, n,
\end{aligned} \tag{5.10}$$

for the factorization and forward substitution, and

$$\begin{aligned}
x_n &= \beta_n, \\
x_i &= \alpha_i x_{i+1} + \beta_i, \quad i = n-1, \ldots, 1,
\end{aligned} \tag{5.11}$$

for the backsubstitution.

We do not perform pivoting in the formulae (5.10) and (5.11) since interchanges of rows or columns of A would destroy its tridiagonal structure. It may thus happen that division by zero occurs in the course of the computation even when A is nonsingular or when the solution of the system computed is unfavorably influenced by roundoff. The conditions for feasibility of the algorithm are presented, e.g., in [92, 199]. An example of such a condition is the positive definiteness of A.

The method (5.10), (5.11) can be generalized even to systems whose matrices have nonzero entries located on more than three diagonals (see, e.g., [171]). The number of operations required to solve the system (5.1) is again $O(n)$ if the bandwidth $2m + 1$ is independent of n.

A method for solving the system (5.1) of n equations is called *fast* if it needs at most $O(n \log n)$ arithmetic operations to yield the solution. The logarithmic factor influences the number of operations weakly if n is large. The term "fast algorithm" is used in other branches of numerical analysis, too (cf., e.g., the fast Fourier transform [51]). The factorization method (5.10) and (5.11) for a tridiagonal system is thus fast in this sense. In general, if A is a bandmatrix of order n then the triangular matrices L and U computed by the matrix factorization (Paragraph 5.1.1) are also bandmatrices with the same m (i.e., $l_{ij} = 0$ for $i - j > m$ and $u_{ij} = 0$ for $j - i > m$). This property is used to construct special algorithms that require $O(m^2 n)$ arithmetic operations and storage of size $(2m + 1)n$ when the system (5.1) is solved. Pivoting is not possible in this case, either.

In the next section we are concerned with the fact that even if there are some zero entries in the band of the matrix A many of them change to nonzero entries in the bands of the matrices L and U.

Diagonal and tridiagonal matrices and bandmatrices are simple examples where sparsity can be exploited. Further standard types are, e.g., *profile matrices* [82] and many others [158].

The discretization of 1D boundary value problems often leads to tridiagonal matrices while the discretization of 2D and 3D problems on simple structured grids usually results in bandmatrices.

5.1.3 General sparse systems

The discretization of 2D and 3D problems on unstructured grids and adaptively locally refined or coarsened grids leads to matrices A that are large but sparse. Unfortunately, they have no regular *zero-nonzero structure* like, e.g., bandmatrices have. We will be concerned with solving such systems by direct methods in this section.

There are heuristic algorithms (see, e.g., [82, 156]) that transform the matrix A of the system – with more or less success – into a bandmatrix with a band as narrow as they can reach. We saw in the previous section that solving banded systems is very efficient.

Implementation of the matrix factorization for sparse matrices whose nonzero entries are placed in no regular pattern is somewhat more difficult. In the course of the factorization, nonzero entries may appear in the matrices L and U in places where the corresponding entry of A is zero. The sum of the numbers of nonzero entries of the matrices L and U, from which the number of nonzero entries of the original matrix A is subtracted, is called the *fill-in*. Heuristic methods that perform suitable permutations of rows of A (and in case of A symmetric also the same permutations of columns to preserve the symmetry) and are capable of minimizing (to some extent) this fill-in are applied before the factorization step (see, e.g., [82, 156, 193]). As a consequence, the number of required arithmetic operations is minimized, too, since the algorithm operates on nonzero entries only. Moreover, it is necessary to ensure in a proper way that only nonzero entries of A, L and U are stored (see, e.g., [156], cf. also Paragraph 1.3.5).

Let us demonstrate with a simple example the importance of proper permutations of rows and columns of a sparse matrix for the efficiency of the solution process [72, 82, 137, 156]. As the individual numerical values of nonzero entries are not important in this step of the algorithm we denote them by crosses and do not show zero entries at all in the following schemes.

Solving a system with the matrix

$$A = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & & & \\ \times & & \times & & \\ \times & & & \times & \\ \times & & & & \times \end{bmatrix}$$

of order 5 by the Gaussian elimination, we naturally start with elimination of the first unknown from all the equations except for the first one, which is just used for this elimination step (cf. Paragraph 5.1.1). The result is that the first column of the matrix contains zeros in all the rows except for the first one, i.e.,

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \end{bmatrix}$$

whose fill-in is now maximal. Further elimination steps lead finally to the filled upper triangular matrix

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \\ & & & & \times \end{bmatrix}.$$

Try another approach. The notion of symmetric matrix can be generalized in a natural way. We say that the matrix A possesses the *symmetric structure* when $a_{ij} \neq 0$ holds if and only if $a_{ji} \neq 0$. Therefore, every symmetric matrix has symmetric structure.

Before elimination, we thus carry out such a permutation of rows and (to preserve the symmetry of its structure) columns of A that transforms the order $\{1, 2, 3, 4, 5\}$ of rows and columns into the order $\{5, 4, 3, 2, 1\}$. We get the matrix

$$\begin{bmatrix} \times & & & & \times \\ & \times & & & \times \\ & & \times & & \times \\ & & & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}.$$

The same first elimination step as before now gives the matrix

$$\begin{bmatrix} \times & & & & \times \\ & \times & & & \times \\ & & \times & & \times \\ & & & \times & \times \\ & \times & \times & \times & \times \end{bmatrix}$$

with no fill-in. After the next elimination steps we finally obtain the upper triangular matrix

$$\begin{bmatrix} \times & & & & \times \\ & \times & & & \times \\ & & \times & & \times \\ & & & \times & \times \\ & & & & \times \end{bmatrix}$$

with no fill-in at all.

The above-mentioned permutations are not performed with matrix rows and columns in the storage. The order in which the Gaussian elimination (matrix factorization) is to be carried out is stored in an auxiliary integer vector of length n.

A simple application of the theory of directed graphs yields the description of fill-in in the course of the Gaussian elimination [156] and heuristic methods to minimize the fill-in (e.g., the minimum degree ordering, reverse Cuthill-McKee algorithm, Gibbs, Pool, and Stockmayer algorithm) are based on this graph theory description.

The efficiency of solving a system depends on the way of storing nonzero entries and on the particular elimination or factorization algorithm that employs this storing system and, moreover, minimizes the fill-in as much as possible. In computer memory, only nonzero entries of A as well as of L and U (including fill-in) are stored. A simple (but rather inefficient) model is to store a real value of the entry together with two integer values, i.e., its row and column indices, in three computer words.

The less the fill-in is, the fewer entries of the matrices L and U are computed, the fewer arithmetic operations are carried out, and the less storage is needed. Moreover, the fewer operations carried out, the less the influence of roundoff. This is the philosophy of very general program packages for solving large sparse systems of linear algebraic equations by direct methods described, e.g., in [72] or used, e.g., in [24]. They proceed in three steps: (1) Analyze, (2) Factorize, (3) Operate (or Solve). Let us briefly characterize these steps.

The Analyze step precedes the factorization. It carries out the minimization of fill-in and its result is a permutation of rows (and possibly also columns for the matrix of symmetric structure). In this step, we need neither numerical values of the entries of the matrix A of the system nor the components of the right-hand part. If our task is to solve several systems with matrices of the same zero-nonzero structure this Analyze step is performed only once.

If it is advantageous for the storage method for the nonzero entries of L and U the next part of Analyze step may be also the *symbolic factorization* whose aim is to find the exact positions of new nonzero entries after factorization (that correspond to the fill-in) and reserve memory for them in the storage scheme.

The next step is Factorize, i.e., the numerical factorization. Now the numerical values of the entries of A are needed (but not the values of the components of the right-hand part yet). The results of the step are factors L and U stored in some particular way in the memory. If we solve several systems with the same matrix A and different right-hand parts we can use the factorization obtained in this step several times.

The last step is Operate, i.e., solving the systems (5.3) or systems with the factors resulting from (5.5). Numerical values of the right-hand part components are now needed.

The aim of the Analyze/Factorize/Operate procedure is to save computer time as well as memory. Unfortunately, the Analyze step may be very time-consuming and, therefore, it pays to use procedures of this sort only when we are to solve a large number of systems with the same matrix and different right-hand parts or at least with matrices of the same zero-nonzero structure. This may be, e.g., the situation when the Newton method is used to linearize a nonlinear problem.

5.1.4 Fast methods for special systems

There are special direct methods that can provide the solution of a system if its matrix has a particular form. These methods are fast in the sense we introduced in Paragraph 5.1.2. In the finite element method, matrices of systems are rather rarely Vandermonde, Toeplitz, etc. Nevertheless, special direct algorithms for these matrices can be found, e.g., in [158].

On structured 2D grids, the discretization can lead to such a matrix that satisfies the conditions necessary for the use of the *cyclic reduction method* briefly described in what follows. It is a fast direct method that, for some discretizations, can also be used as part of some iterative process converging to the solution of the discrete problem.

Let us solve the system (5.1) now in the notation

$$Bu = v \tag{5.12}$$

and let it be written in the block form

$$\begin{bmatrix} A & -T & & & \\ -T & A & -T & & \\ & \ddots & \ddots & \ddots & \\ & & -T & A & -T \\ & & & -T & A \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \\ u_N \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{N-1} \\ v_N \end{bmatrix}. \tag{5.13}$$

Zero (null) blocks are not shown in the matrix B. It has N block rows as well as columns and blocks A and T are square matrices of order M where both M and N are integers. The square matrix B is called *block tridiagonal* and its "pointwise" order is $n = MN$ here. We assume that

$$N = 2^{s+1} - 1 \tag{5.14}$$

with a suitable integer s for the same reason as in the fast Fourier transform [51].

There are further necessary conditions for feasibility of the cyclic reduction [69, 190]. The strongest of them is that the matrices A and T commute, i.e.,

$$AT = TA. \tag{5.15}$$

This condition is fulfilled, e.g., in the trivial case $T = I$ that is rather frequent in discretizations. Some further conditions on A and T (their sparsity) are needed [190] in order that the algorithm be fast. If they are satisfied the number of arithmetic operations of the algorithm is $O(n \log n)$ as compared with $O(n^3)$ in the standard Gaussian elimination.

The derivation of the cyclic reduction algorithm follows the idea of the derivation of the fast Fourier transform. It consists of successive and systematic Gaussian elimination in the block system (5.13) that reduces the number of unknowns by about half in each algorithm step and that can also be split into the elimination and backsubstitution procedures.

Let us turn back to solving the system (5.13). Introduce formally the vectors

$$u_0 = 0 \quad \text{and} \quad u_{N+1} = 0$$

of M components. Then we can rewrite the odd block rows of the system (5.13) as

$$-Tu_{2j} + Au_{2j+1} - Tu_{2j+2} = v_{2j+1}, \tag{5.16}$$

$j = 0, \ldots, 2^s - 1$. As soon as we know the values of vectors u_{2j} and u_{2j+2} we can calculate u_{2j+1} from (5.16) in such a way that we solve the block equation

$$Au_{2j+1} = Tu_{2j} + v_{2j+1} + Tu_{2j+2}, \tag{5.17}$$

$j = 0, \ldots, 2^s - 1$. The block equation (5.17) is, if considered "pointwise," a system of M linear algebraic equations for M unknown components of the vector u_{2j+1} that can be solved, e.g., by the Gaussian elimination. We will, therefore, assume that the matrix A is nonsingular in what follows. We will discuss the procedure for solving (5.17) later.

Let us now write down an even block row and two odd rows that precede and follow it. We then have the system of three block equations

$$\begin{aligned}
-Tu_{2j-2} + Au_{2j-1} - Tu_{2j} \qquad\qquad &= v_{2j-1},\\
-Tu_{2j-1} + Au_{2j} - Tu_{2j+1} \qquad &= v_{2j},\\
-Tu_{2j} + Au_{2j+1} - Tu_{2j+2} &= v_{2j+1},
\end{aligned}$$

$j = 1, \ldots, 2^s - 1$. We multiply the first equation by the matrix T, the second one by the matrix A, and the third one by the matrix T again, every time from the left. We then get

$$\begin{aligned}
-T^2u_{2j-2} + TAu_{2j-1} - T^2u_{2j} \qquad\qquad &= Tv_{2j-1},\\
-ATu_{2j-1} + A^2u_{2j} - ATu_{2j+1} \qquad &= Av_{2j},\\
-T^2u_{2j} + TAu_{2j+1} - T^2u_{2j+2} &= Tv_{2j+1},
\end{aligned}$$

$j = 1, \ldots, 2^s - 1$. Adding the first and third equations to the second one and employing the commutativity property (5.15) we finally get

$$-T^2u_{2j-2} + (A^2 - 2T^2)u_{2j} - T^2u_{2j+2} = Tv_{2j-1} + Av_{2j} + Tv_{2j+1}, \quad (5.18)$$

$j = 1, \ldots, 2^s - 1$. The block equations (5.18) serve for the calculation of the unknown vectors u_{2j} with even indices and their number is about one half compared with the original number of equations $N = 2^{s+1} - 1$. If we write down the matrix of the system we have just obtained we have

$$\begin{bmatrix}
A^2 - 2T^2 & -T^2 & & & \\
-T^2 & A^2 - 2T^2 & -T^2 & & \\
 & \ddots & \ddots & \ddots & \\
 & & -T^2 & A^2 - 2T^2 & -T^2 \\
 & & & -T^2 & A^2 - 2T^2
\end{bmatrix}.$$

Apparently, this matrix has the same form as the matrix (5.13) of the system and satisfies the conditions for performing the next elimination step. If we proceed further in this way we obviously obtain, after s elimination (or *reduction*) steps, a single (block) equation for a single unknown vector. If we solve this equation we can, with the help of equations of the form (5.17), start the backsubstitution ending with all the unknown vectors $u_1, \ldots, u_N$ found. Since the steps of the algorithm are carried out cyclically we call this the cyclic reduction method.

Formally, the cyclic reduction method can be described by recurrent formulae for the computation of sequences of matrices T_i and A_i, and vectors u_j^i and v_j^i. We set

$$T_0 = T; \quad A_0 = A; \quad u_j^0 = u_j, \quad v_j^0 = v_j, \quad j = 1, \dots, 2^{s+1} - 1, \tag{5.19}$$

and, in the forward course, we successively compute

$$\begin{aligned} T_i &= T_{i-1}^2, \\ A_i &= A_{i-1}^2 - 2T_{i-1}^2, \\ u_j^i &= u_{2j}^{i-1}, \quad j = 1, \dots, 2^{s+1-i} - 1, \\ v_j^i &= T_{i-1} v_{2j-1}^{i-1} + A_{i-1} v_{2j}^{i-1} + T_{i-1} v_{2j+1}^{i-1}, \quad j = 1, \dots, 2^{s+1-i} - 1, \end{aligned} \tag{5.20}$$

for $i = 1, \dots, s$. The third of the above equations represents the renumbering of vectors of the unknowns only.

After s steps we obtain a single block equation

$$A_s u_1^s = v_1^s, \tag{5.21}$$

i.e., a system of M equations for M unknowns. We solve the equation (5.21) and carry out the backsubstitution. In each step, we successively renumber the unknown vectors

$$u_{2j}^i = u_j^{i+1}, \quad j = 1, \dots, 2^{s-i} - 1,$$

put formally

$$u_0^i = u_{2^{s+1-i}}^i = 0$$

and solve systems of order M

$$A_i u_{2j+1}^i = T_i u_{2j}^i + v_{2j+1}^i + T_i u_{2j+2}^i, \quad j = 0, \dots, 2^{s-i} - 1, \tag{5.22}$$

this time for $i = s - 1, \dots, 1, 0$. We assume that all the matrices A_i, $i = 0, \dots, s$, are nonsingular to be able to solve the systems (5.21) and (5.22).

We will discuss the numerical stability of this process later.

The most time-consuming components of the algorithm just presented are the computation of the right-hand parts in the last equation of (5.20) and in (5.22), and solution of the systems (5.21) and (5.22). Considering the recurrence formulae for the computation of T_i and A_i in (5.20), we see that if $T = T_0$ is diagonal then T_i is diagonal, too. However, it is easy to verify that if $A = A_0$ is tridiagonal (which is the simplest practical 2D case) then A_0^2 has five nonzero diagonals (it is banded with bandwidth $m = 2$, cf. (5.9)), A_1^2 has seven nonzero diagonals, etc. Moreover, for a fixed M and sufficiently large N there exists a number K such that the matrices A_i, $i \geq K$, are completely filled. Solving the system with the matrix A_i then requires $O(M^3)$ arithmetic operations. The same number of operations in order is needed for computation of the right-hand parts.

However, it can be shown that there exist factorizations

$$\begin{aligned} T_i &= T^{2^i}, \\ A_i &= \prod_{r=1}^{2^i} \left(A - 2\cos\frac{(2r-1)\pi}{2^{i+1}}\, T \right). \end{aligned} \tag{5.23}$$

Apparently, if A is a bandmatrix of bandwidth $2m_A + 1$ and T that of bandwidth $2m_T+1$ then the factors of A_i in (5.23) have bandwidth $2\max\{m_A, m_T\}$ $+1$ and those of T_i the same bandwidth as T, i.e., $2m_T + 1$.

We use the factorization formulae (5.23) for the computation of the right-hand parts in (5.20) and (5.22) in an obvious way. We employ the factorization of A given in (5.23) in the same way as the factorization (5.2) was used in (5.3) to solve (5.1). Each of the systems is solved by the double sweep method and requires thus $O(M)$ arithmetic operations. Taking into account the number 2^{2^i} of factors, the number s of steps of the algorithm, and the assumption (5.14), an easy calculation shows that if A as well as T are bandmatrices then the number of operations needed is $O(MN \log N)$. The base of the logarithm should be 2 but because we are interested only in the order of the number of operations the base of the logarithm can be arbitrary.

Therefore, the cyclic reduction method is a fast direct method in the sense of our definition in Paragraph 5.1.2. This direct method belongs to a wide class of *block methods* that usually are iterative (cf. Paragraph 1.2.5). The given system (5.13) is never needed in memory in the complete general form. We successively operate on $M \times M$ blocks only. This is advantageous with respect to the architecture of both serial and parallel computers where several "small" systems can even be solved at the same time.

It can be shown that the algorithm just described is, unfortunately, numerically unstable. There is a simple remedy to this drawback. It is sufficient to calculate, instead of the sequence v_j^i, two sequences, p_j^i and q_j^i, such that

$$v_j^i = A_i p_j^i + q_j^i$$

and replace the initial condition (5.19) by its obvious generalization (see, e.g., [171]). We now need twice the number of arithmetic operations to carry out the algorithm (we even have to solve a new sequence of linear algebraic equations with matrices A_i), but the order of the number of operations remains unchanged, i.e., $O(MN \log N)$.

There are several generalizations of the elliptic problem solved, boundary conditions imposed, and the assumption (5.14) leading to a more complex algorithm but, nevertheless, preserving the order of the number of arithmetic operations, cf., e.g., [190].

All the algorithms for the cyclic reduction method are available in [190] and also on the web as freeware mentioned in the introduction to this chapter.

5.2 Iterative methods for linear algebraic equations

Iterative methods for solving linear algebraic systems have become a classic part of numerical analysis and are treated in a vast literature (see, e.g., [120, 82, 92, 100, 127, 137, 159, 163, 169, 196, 199]). We only present some basic iterative methods useful for solving large sparse systems. The procedures given in Paragraphs 5.2.1 to 5.2.3 provide Krylov subspace approximations, i.e., approximations x_k having the property

$$x_k \in \operatorname{span}\{b, Ab, \ldots, A^{k-1}b\}. \tag{5.24}$$

We show later in Paragraphs 5.2.4 to 5.2.6 how a proper combination of iterative and direct methods (e.g., preconditioning) may lead to the acceleration of convergence of iterative methods.

In general, the number of arithmetic operations needed for solving a sparse system of equations iteratively depends on the number of operations required for the computation of the product Ay where y is some vector (for a single iteration step) and also on the number of iteration steps to be performed to reach the accuracy prescribed.

We do not consider roundoff error in this section. A detailed treatment of this subject can be found, e.g., in [100].

Any practical computation can involve a finite number of arithmetic operations only. It would thus be suitable to stop the iterative process in the kth step if $\|x_k - x_{\mathrm{t}}\| < \varepsilon$ for some tolerance ε chosen in advance. We, however, do not know the true solution x_{t} and thus choose stopping (termination) criteria mostly in the form

$$\|x_{k+1} - x_k\| < \varepsilon$$

or

$$\|r_{k+1}\| < \varepsilon,$$

where r_{k+1} is the residual (5.8).

5.2.1 ORTHOMIN and steepest descent methods

One-point matrix iterative methods for solving the system (5.1) of n equations consist of the following procedure: We choose an arbitrary initial approximation x_0 and compute further approximations by the formula

$$x_{k+1} = B_k x_k + C_k b, \quad k = 0, 1, \ldots,$$

where B_k and C_k are square matrices of order n constructed in an appropriate way from A. A new approximation x_{k+1} is thus computed from the last preceding approximation x_k only. If the method converges and is consistent (see further) then

$$\lim_{k\to\infty} x_k = x_{\mathrm{t}}, \tag{5.25}$$

where x_{t} is the true solution of (5.1) and the limit (5.25) is to be understood in the sense of certain metric, e.g.,

$$\lim_{k\to\infty} \|x_k - x_{\mathrm{t}}\| = 0,$$

where

$$\|y\| = \sqrt{(y, y)}$$

is the Euclidean norm of an n component vector y.

The iterative procedure

$$x_{k+1} = Bx_k + Cb, \quad k = 0, 1, \ldots, \tag{5.26}$$

where x_0 is an arbitrary initial approximation, and B and C are square matrices of order n (independent of k), is called a *stationary one-point matrix iterative method* for solving the system (5.1). We say that this method is *consistent* with the system (5.1) if

$$CA + B = I,$$

where I is the identity matrix and B is called the *iteration matrix.*

Consistency of the method ensures that the convergent sequence of approximations x_k tends to the true solution x_{t}. In fact, the iteration formula (5.26) is transformed into an identity after substituting x_{t} for both x_k and x_{k+1}.

The simplest stationary one-point matrix iterative method is obtained from (5.1) by adding x to both parts of the equation. Then

$$x = (I - A)x + b, \tag{5.27}$$

and now putting x_{k+1} instead of x on the left-hand part of (5.27) and x_k instead of x on the right-hand part, we arrive at the formula

$$x_{k+1} = (I - A)x_k + b \tag{5.28}$$

or

$$x_{k+1} = x_k + (b - Ax_k). \tag{5.29}$$

This procedure is usually called the *simple iteration method* and is obviously consistent.

A natural generalization of the procedure (5.29) consists of multiplying the term $b - Ax_k$ on the right-hand part by an inverse to a nonsingular matrix M. We obtain the *preconditioned simple iteration method*

$$x_{k+1} = x_k + M^{-1}(b - Ax_k) \tag{5.30}$$

that is the basis for deriving several classic iterative methods to be described in Paragraph 5.2.4.

Let us now consider elementary attempts to improve the simple iteration (5.29) by introducing dynamically computed parameters into the iteration. Put

$$x_{k+1} = x_k + a_k(b - Ax_k), \tag{5.31}$$

where a_k is a real parameter. Multiplying the equation (5.31) by $-A$ and adding b to both its parts, we find out that the residual r_{k+1} (cf. (5.8)) satisfies

$$r_{k+1} = r_k - a_k A r_k.$$

Moreover, introducing the error

$$e_k = x_{\mathrm{t}} - x_k = A^{-1}b - x_k,$$

we similarly find out that

$$e_{k+1} = e_k - a_k r_k. \tag{5.32}$$

A standard optimization procedure applied to the minimization of $\|r_{k+1}\|$ gives

$$a_k = \frac{(r_k, Ar_k)}{(Ar_k, Ar_k)}.$$

If we choose a_k in this way the method produces a residual r_{k+1} that is obviously equal to r_k minus its projection onto Ar_k. It follows that $\|r_{k+1}\| \leq \|r_k\|$ with equality if and only if r_k is already orthogonal to Ar_k. This is the reason why this procedure is called the *ORTHOMIN method.*

Let us now assume that the matrix A is symmetric positive definite. Introducing the A-*norm* (*energy norm*)

$$\|y\|_A = \sqrt{(y, Ay)}$$

of an n-component vector, we can consider the minimization of the error (5.32) in this norm. The same reasoning as above now gives

$$a_k = \frac{(e_k, Ar_k)}{(r_k, Ar_k)} = \frac{(r_k, r_k)}{(r_k, Ar_k)}.$$

The procedure (5.31) with this choice of a_k is called the *steepest descent method* since the problem (5.1) can then be identified with the problem of minimizing the quadratic functional

$$\tfrac{1}{2}(x, Ax) - (b, x), \tag{5.33}$$

whose minimum is x_{t}. A simple calculation shows that the negative gradient or direction of the steepest descent of this functional at $x = x_k$ is $r_k = b - Ax_k$.

We can show the behavior of the steepest descent method on a simple example. We consider the 2×2 system (cf. [177])

$$\begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ -8 \end{bmatrix} \tag{5.34}$$

with the true solution $(2, -2)^T$. In Figure 5.1, contour lines of the quadratic form (5.33) corresponding to the model system (5.34) and the course of the steepest descent algorithm are shown. Note the zigzag path that appears as each gradient is orthogonal to the previous gradient.

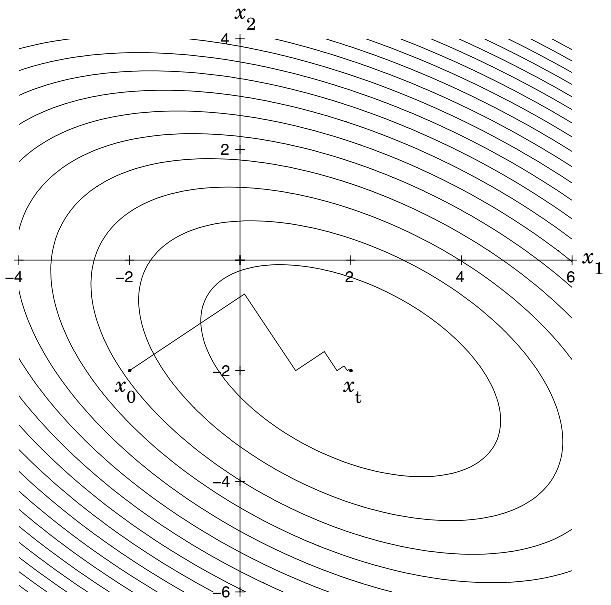

FIGURE 5.1: Steepest descent method. Contour lines of the quadratic form (5.33) corresponding to the model system (5.34) are shown. The initial approximation is $x_0 = (-2, -2)^T$, the true solution is $x_{\mathrm{t}} = (2, -2)^T$.

Both the specific methods just presented, the ORTHOMIN and the steepest descent method, are mostly of theoretical use. We refer to them in the next sections when developing some further practical iterative methods. The convergence and detailed error analysis of the above two methods can be found, e.g., in [11, 100, 159].

5.2.2 Conjugate gradient and biconjugate gradient methods

In the first part of this section we again assume that A is a symmetric positive definite matrix of the system (5.1). The iterations (5.31) of the steepest descent method then can be rewritten in the form

$$x_{k+1} = x_k + a_k p_k,$$

where the *direction vectors* p_k are constructed from the residual vectors r_k to be A-*orthogonal* (A-*conjugate*), i.e.,

$$(p_k, Ap_k) = 0.$$

The residual and error vectors then satisfy

$$r_{k+1} = r_k - a_k A p_k, \quad e_{k+1} = e_k - a_k p_k,$$

where the coefficient a_k is chosen so that e_{k+1} is A-orthogonal to p_k, i.e., $(e_{k+1}, Ap_k) = 0$. We then can modify the steepest descent method so that it eliminates the A-projection of the error in a direction that is already A-orthogonal to the previous direction vector, i.e., in the direction

$$\tilde{p}_k = r_k - \frac{(r_k, Ap_{k-1})}{(p_{k-1}, Ap_{k-1})} p_{k-1}.$$

Then we have

$$(e_{k+1}, A\tilde{p}_k) = (e_{k+1}, Ap_{k-1}) = 0$$

and the A-norm of the error is minimized over the two-dimensional *affine space* (manifold) $e_k + \text{span}\{r_k, p_{k-1}\}$ where $\text{span}\{y_1, \ldots, y_l\}$ is the set of all linear combinations of the vectors $y_1, \ldots, y_l$ for some l positive. The algorithm that does this is called the *conjugate gradient* or CG method first proposed by Hestenes and Stiefel [108] more than 50 years ago. It is implemented with an initial approximation x_0 and several equivalent coefficient formulae, cf. [11, 100, 137], one of which is

$$\begin{aligned}
r_0 &= b - Ax_0, \\
p_0 &= r_0, \\
a_{k-1} &= \frac{(r_{k-1}, r_{k-1})}{(p_{k-1}, Ap_{k-1})}, \\
x_k &= x_{k-1} + a_{k-1}p_{k-1}, \\
r_k &= r_{k-1} - a_{k-1}Ap_{k-1}, \\
b_{k-1} &= \frac{(r_k, r_k)}{(r_{k-1}, r_{k-1})}, \\
p_k &= r_k + b_{k-1}p_{k-1}, \quad k = 1, 2, \ldots .
\end{aligned} \tag{5.35}$$

If the conjugate gradient algorithm is only used with symmetric positive definite matrices A the coefficients are always defined and it can be shown [100] that the A-norm of the error is minimized over the affine space $e_0 + \operatorname{span}\{p_0, p_1, \ldots, p_k\}$. The algorithm then generates the exact solution to the linear system (5.1) in n steps at the most. The error, residual, and direction vectors generated before the exact solution is reached are defined and satisfy

$$(e_{k+1}, Ap_j) = (p_{k+1}, Ap_j) = (r_{k+1}, r_j) = 0 \quad \text{for all} \quad j \le k.$$

It follows [100] that of all vectors in the affine space

$$e_0 + \operatorname{span}\{Ae_0, A^2e_0, \ldots, A^{k+1}e_0\},$$

e_{k+1} has the smallest A-norm. Obviously the coefficients in the conjugate gradient algorithm are not defined if the residual vector is zero, in which case the exact solution has been found.

The conjugate gradient method can also be derived as a method for minimizing the quadratic functional (5.33). In Figure 5.2, contour lines of the quadratic form (5.33) corresponding to the model system (5.34) and the course of the conjugate gradient algorithm are shown. Note that in this case of two equations the second step of the algorithm gives the exact solution.

However, the conjugate gradient method is not important as a direct method giving the exact solution in n steps at the most. The analysis of convergence of the method shows [92, 100] that for large classes of finite element systems a sufficiently accurate solution can be obtained in far fewer steps. The rate of convergence of the method depends, in general, on the *condition number* $\varkappa(A) = \|A\|\|A^{-1}\|$ (*spectral condition number*) and it increases with decreasing $\varkappa(A)$ [100, 159]. (Note that the condition number cannot be less than 1.) The method is most efficient for large well-conditioned matrices A.

The algorithm can be carried out even for matrices that are not symmetric positive definite, but it can then fail any time (p_{k-1}, Ap_{k-1}) is zero. For such matrices, the *biconjugate gradient* or BiCG method was derived. While the conjugate gradient method can also be used for complex Hermitian positive definite matices the biconjugate gradient method is easily applicable to complex non-Hermitian matrices, too. We present, however, its real version.

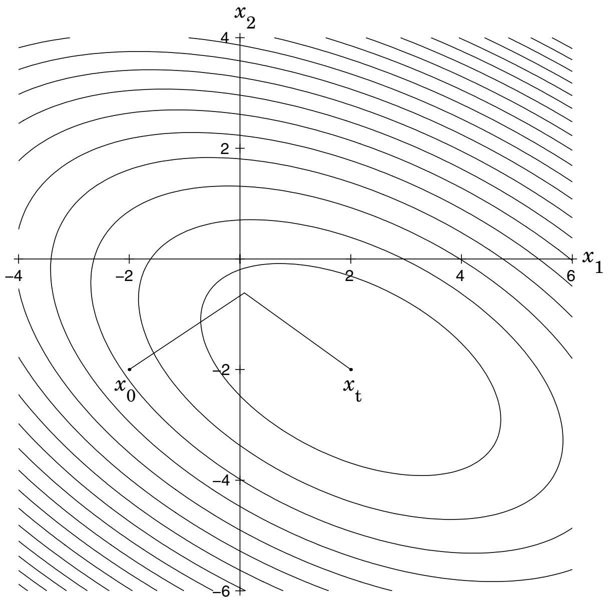

FIGURE 5.2: Conjugate gradient method. Contour lines of the quadratic form (5.33) corresponding to the model system (5.34) are shown. The initial approximation is $x_0 = (-2, -2)^T$, the true solution is $x_\mathrm{t} = (2, -2)^T$.

Given x_0, we compute

$$\begin{aligned} r_0 &= b - Ax_0, \\ p_0 &= r_0. \end{aligned}$$

We choose $\hat{r}_0$ such that $(r_0, \hat{r}_0) \neq 0$, set $\hat{p}_0 = \hat{r}_0$, and calculate

$$\begin{aligned} a_{k-1} &= \frac{(r_{k-1}, \hat{r}_{k-1})}{(Ap_{k-1}, \hat{p}_{k-1})}, \\ x_k &= x_{k-1} + a_{k-1} p_{k-1}, \\ r_k &= r_{k-1} - a_{k-1} A p_{k-1}, \\ \hat{r}_k &= \hat{r}_{k-1} - a_{k-1} A^T \hat{p}_{k-1}, \\ b_{k-1} &= \frac{(r_k, \hat{r}_k)}{(r_{k-1}, \hat{r}_{k-1})}, \\ p_k &= r_k + b_{k-1} p_{k-1}, \\ \hat{p}_k &= \hat{r}_k + b_{k-1} \hat{p}_{k-1}, \quad k = 1, 2, \ldots. \end{aligned} \tag{5.36}$$

Even this algorithm for a nonsymmetric matrix A of the system (5.1) may fail but there is a way to restart it, see, e.g., [100, 159]. If the matrix A of the system is symmetric positive definite the biconjugate gradient algorithm is apparently identical to the conjugate gradient algorithm.

Note that both the algorithms (5.35) and (5.36) require, as far as the matrix A of the system (5.1) is concerned, only a procedure for computing the matrix-vector product Ay. This provides us with great freedom in storing the large sparse matrix A and, moreover, a possibility to economize the number of operations in each iteration step.

The conjugate gradient method for a symmetric matrix A can also be derived from the *Lanczos algorithm*, see, e.g., [100]. The Lanczos algorithm starts with an arbitrary vector q_1 such that $\|q_1\| = 1$ and a parameter $\beta_0 = 0$. Then we calculate

$$\begin{aligned} \tilde{q}_{j+1} &= Aq_j - \beta_{j-1}q_{j-1}, \\ \alpha_j &= (\tilde{q}_{j+1}, q_j), \\ \tilde{q}_{j+1} &= \tilde{q}_{j+1} - \alpha_j q_j, \\ \beta_j &= (\tilde{q}_{j+1}, \tilde{q}_{j+1}), \\ q_{j+1} &= \tilde{q}_{j+1}/\beta_j, \quad j = 1, 2, \ldots. \end{aligned}$$

It can be shown that the Lanczos algorithm yields an orthonormal basis for the Krylov space (cf. (5.24)) formed by the matrix A and the vector q_1.

Analogously, there exists the *two-sided Lanczos algorithm* for a nonsymmetric (or complex-valued non-Hermitian) matrix A. We construct *biorthogonal bases* for the Krylov spaces corresponding to both A and A^T defined by a pair of three-term recurrences, cf., e.g., [100]. This two-sided Lanczos algorithm can be used for deriving the biconjugate gradient method. The method does not minimize the residual, hence in practical calculation we may observe many local peaks in the convergence curve.

There are several additional methods based on similar principles, see, e.g., [100, 159].

5.2.3 MINRES and GMRES methods

Returning to the case of general matrices A, the idea of minimizing over a larger subspace can be extended, at the price of having to save and orthogonalize vectors against additional vectors at each step. To minimize the Euclidean norm of the residual r_{k+1} over the j-dimensional affine space

$$r_k + \mathrm{span}\{Ap_k, Ap_{k-1}, \ldots, Ap_{k-j+1}\} = r_k + \mathrm{span}\{Ar_k, Ap_{k-1}, \ldots, Ap_{k-j+1}\},$$

we set

$$p_k = r_k - \sum_{l=1}^{j-1} b_{k-l}^{(k)} p_{k-l}, \quad b_{k-l}^{(k)} = \frac{(Ar_k, Ap_{k-l})}{(Ap_{k-l}, Ap_{k-l})}. \tag{5.37}$$

This defines the *ORTHOMIN*(j) *procedure*. Unfortunately, the algorithm can fail. The possibility of failure can be eliminated by replacing r_k in (5.37) with Ap_{k-1}. This algorithm is known as *ORTHODIR*(j).

In the case of Hermitian matrices the ORTHODIR(3) algorithm minimizes the Euclidean norm of the residual over the entire affine space

$$r_0 + \mathrm{span}\{Ap_0, Ap_1, \ldots, Ap_k\} = r_0 + \mathrm{span}\{Ar_0, A^2r_0, \ldots, A^kr_0\}. \quad (5.38)$$

This provides an implementation of the *minimal residual* or MINRES algorithm for Hermitian matrices [100].

Given x_0, we compute

$$\begin{aligned} r_0 &= b - Ax_0, \\ p_0 &= r_0, \\ s_0 &= Ap_0. \end{aligned}$$

We then calculate for $k = 1, 2, \ldots$

$$\begin{aligned} a_{k-1} &= \frac{(r_{k-1}, s_{k-1})}{(s_{k-1}, s_{k-1})}, \\ x_k &= x_{k-1} + a_{k-1}p_{k-1}, \\ r_k &= r_{k-1} - a_{k-1}s_{k-1}, \\ p_k &= s_{k-1}, \\ s_k &= As_{k-1}, \\ b_{k-l}^{(k)} &= \frac{(s_k, s_{k-l})}{(s_{k-l}, s_{k-l})}, \\ p_k &\leftarrow p_k - b_{k-l}^{(k)}p_{k-l}, \\ s_k &\leftarrow s_k - b_{k-l}^{(k)}s_{k-l}, \quad l = 1, 2. \end{aligned}$$

Unfortunately, roundoff error accumulation may cause the vectors s_k, which are supposed to be equal to Ap_k, to differ. This could be corrected occasionally by setting $s_k = Ap_k$ explicitly at the cost of a matrix-vector multiplication.

For general matrices, if $j = n$ the ORTHODIR(n) algorithm minimizes the Euclidean norm of the residual over the affine space (5.38) at each step k. Therefore, the exact solution is obtained in at most n iteration steps (if there is no roundoff error) but at the cost of storing up to n search directions p_k (as well as auxiliary vectors $s_k = Ap_k$) and orthogonalizing against k direction vectors at each step $k = 1, \ldots, n$. It can be shown [100] that the ORTHODIR(n) algorithm then requires $O(n^2)$ storage and $O(n^3)$ work, i.e., it has the same requirements as the Gaussian elimination with a dense (nonsparse) matrix. The advantage of the method is that, at each step, the residual norm is minimized over the space (5.38) and an acceptable approximate solution can be hoped to be obtained in far fewer than n steps.

There is another way to compute the approximation x_k for which the norm of r_k is minimized over the space (5.38). It is the *GMRES method* that is a *projection method* and has better numerical properties and more favorable storage requirements. The method uses a particular form of the Gram-Schmidt orthogonalization process (see, e.g., [92, 137, 196]) to construct an orthonormal basis for the Krylov space span$\{r_0, Ar_0, \ldots, A^k r_0\}$. In practice, the method is usually employed in its restarted version.

This modification is called the *Arnoldi algorithm.* It starts with choosing q_1, $\|q_1\| = 1$. We then compute for $j = 1, 2, \ldots$

$$\begin{aligned}
\tilde{q}_{j+1} &= Aq_j, \\
h_{ij} &= (\tilde{q}_{j+1}, q_i), \\
\tilde{q}_{j+1} &\leftarrow \tilde{q}_{j+1} - h_{ij} q_i, \quad i = 1, 2, \ldots, j, \\
h_{j+1,j} &= \|\tilde{q}_{j+1}\|, \\
q_{j+1} &= \tilde{q}_{j+1}/h_{j+1,j}.
\end{aligned}$$

The GMRES algorithm can be briefly described in terms of the Arnoldi algorithm and QR factorization algorithm (see, e.g., [92, 100, 163, 169]). Form an $n \times k$ matrix Q_k of the orthonormal basis vectors $q_1, \ldots, q_k$ and a $k \times k$ matrix H_k with the entry (i,j) equal to h_{ij} for $j = 1, \ldots, k$, $i = 1, \ldots, \min\{j+1, k\}$, and zero otherwise (upper Hessenberg matrix). Given x_0 and a unit column vector $\xi = (1, 0, \ldots, 0)$ of $k+1$ components, we compute

$$\begin{aligned}
r_0 &= b - Ax_0, \\
\beta &= \|r_0\|, \\
q_1 &= r_0/\beta.
\end{aligned}$$

We then carry out the following two steps for $k = 1, 2, \ldots$.

(1) Calculate q_{k+1} and h_{ik}, $i = 1, \ldots, k+1$, by the Arnoldi algorithm.
(2) Calculate $x_k = x_0 + Q_k y_k$, where y_k is the solution to the least squares problem $\min_y \|\beta\xi_1 - H_{k+1,k} y\|$.

Here H_k is the upper Hessenberg matrix with the (i,j)-entry equal to h_{ij} for $j = 1, \ldots, k$; $i = 1, \ldots, \min\{j+1, k\}$, and all other entries zero. The $(k+1) \times k$ matrix $H_{k+1,k}$ is the matrix whose top $k \times k$ block is H_k and whose last row is zero except for the $(k+1, k)$-entry which is $h_{k+1,k}$.

A standard method for solving the least squares problem $\min_y \|\beta\xi_1 - H_{k+1,k} y\|$ employs the QR factorization of the matrix $H_{k+1,k}$ accomplished using plane rotations [92, 163]. However, the (full) GMRES algorithm just described may be impractical because of increasing storage and work requirements if the number of iterations needed to solve the original linear system (5.1) is large. The *GMRES*(j) *algorithm* is defined by restarting GMRES every j steps, using the last iterate as the initial guess for the next GMRES computation. The convergence of full and restarted GMRES is analyzed in [100] in detail.

5.2.4 Classical iterative methods and preconditioning

We reconsider the simple iteration method (cf. Paragraph 5.2.1) and some Krylov subspace methods in this section to show that there are ways to improve the convergence of an iterative method by *preconditioning.* The rate of convergence of a matrix iterative method (5.26) depends on the spectral radius of the iteration matrix B [100] or, with the Krylov subspace methods treated in the previous sections, on the condition number of B [100]. We only present elementary examples of preconditioned iterative methods based on the difference between the formulae (5.29) and (5.30). The procedures that carry out the preconditioning, the *preconditioners*, i.e., the choice of the matrix M (or M^{-1}), are represented here by the simplest ones only.

The conclusion of this section should be a general rule: Iterative methods are always to be used with preconditioning only. We will show some very simple and very cheap ways of preconditioning that can be used for all systems. There are, certainly, also more sophisticated methods of preconditioning very efficient for special classes of linear algebraic systems.

Let us first present some "classic" iterative methods [92, 100, 137].

Considering the simple iterative method in the form (5.26), we can show [169] that the process is convergent with any initial approximation x_0 if and only if $\rho(B) < 1$, where

$$\rho(B) = \max_{i=1,\ldots,n} |\lambda_i|$$

is the spectral radius of the iteration matrix B and λ_i are eigenvalues of B. A stronger but practically verifiable condition for the convergence is that $\|B\| < 1$ for some matrix norm since then $\rho(B) < 1$ follows. The rate of convergence increases with decreasing $\rho(B)$.

Let us rewrite the matrix A of the system (5.1) in the form

$$A = D - E - F,$$

where D is a diagonal matrix, E is a lower triangular matrix with zero diagonal and F is an upper triangular matrix with zero diagonal. If all diagonal entries of A are nonzero we can substitute into (5.1), obtaining

$$Dx = (E + F)x + b,$$

from where consistency follows, and further

$$x = D^{-1}(E + F)x + D^{-1}b.$$

Finally, we have the iteration formula

$$x_{k+1} = D^{-1}(E + F)x_k + D^{-1}b$$

with an arbitrary x_0, known as the *Jacobi method.* For individual components (the components of x_k are denoted by $x_i^{(k)}$)

$$x_i^{(k+1)} = -\frac{1}{a_{ii}} \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(k)} + \frac{b_i}{a_{ii}}, \quad i = 1, \ldots, n, \tag{5.39}$$

which means that the ith component of x_{k+1} is calculated from the ith equation of the system (5.1) in which we have substituted the components of x_k for all the other components of x.

If we again calculate the ith component of x_{k+1} from the ith equation like in (5.39) but now also use the components $x_j^{(k+1)}$, $j < i$, of the new approximation that have just been computed we obtain the *Gauss-Seidel method*

$$x_i^{(k+1)} = -\frac{1}{a_{ii}} \left(\sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} + \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) + \frac{b_i}{a_{ii}}, \quad i = 1, \ldots, n, \quad (5.40)$$

or, in matrix notation,

$$x_{k+1} = (D - E)^{-1} F x_k + (D - E)^{-1} b.$$

It is easy to prove that the Gauss-Seidel method converges for an arbitrary initial approximation if the matrix A is positive definite.

The *successive overrelaxation* (SOR) *method* is an improvement of the Gauss-Seidel method. It includes, in addition, a real *relaxation parameter* ω. The corresponding formula is

$$x_i^{(k+1)} = -\omega \frac{1}{a_{ii}} \left(\sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} + \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} + b_i \right) + (1 - \omega) x_i^{(k)}, \quad (5.41)$$
$$i = 1, \ldots, n,$$

or, in the matrix notation,

$$x_{k+1} = (D - \omega E)^{-1} ((1 - \omega) D + \omega F) x_k + \omega (D - \omega E)^{-1} b.$$

If $\omega = 1$ the method coincides with the Gauss-Seidel method. For some classes of matrices, the SOR method converges for a certain range of values of ω and it is (at least theoretically) possible to find an optimal value ω_0 that minimizes the spectral radius of the iteration matrix (and thus maximizes the rate of convergence) through a fairly sophisticated eigenvalue analysis, see, e.g., [11, 92, 100, 199, 207]. There is also a symmetrized version of the SOR method called *symmetric successive overrelaxation method* (SSOR).

We have mentioned the conditions for convergence and the estimates of rate of convergence for Krylov subspace methods as well as methods of this section. We wish to show now how to increase the rate of convergence (to precondition an iterative method) at the cost of solving an auxiliary system of n linear algebraic equations in each step of the iterative method.

Let us solve the system of n equations (5.1) by an iterative method. Choose a nonsingular matrix M of order n with the following two properties:

M1. The system

$$Mx = c \tag{5.42}$$

can be solved by an efficient (preferably fast) direct method, cf. Paragraph 5.1.2.

M2. M is in some sense close to the matrix A of the system (5.1) solved. Setting

$$E = M - A, \tag{5.43}$$

"closeness" can mean that $\|E\|$ is small or that the product $M^{-1}A$ is close to the identity matrix, i.e., that $\|I - M^{-1}A\|$ is small.

Apparently, the choice ideal from the viewpoint of Property M1 would be $M = I$. This, however, is no preconditioning since, e.g., (5.30) then coincides with the original form (5.29). From the viewpoint of Property M2, the ideal choice is $M = A$. But M also has to possess Property M1, i.e., we must be able to solve (5.42) fast, and there is no reason to solve a system with the matrix A by an iterative method if a fast direct method can be applied.

Let us first consider stationary iterative methods. We have presented the simple iteration method in the preconditioned form (5.30) where the corresponding iteration matrix is $I - M^{-1}A$, i.e.,

$$Mx_{k+1} = (M - A)x_k + b. \tag{5.44}$$

The formula (5.44) is a system of n equations with the matrix M and a known right-hand part. The new approximation x_{k+1} is obtained as a solution of this system. According to Property M1, the system (5.44) can be solved by a fast direct method and the order of the number of arithmetic operations required is thus not increased by preconditioning. Naturally, the aim is to increase the rate of convergence, i.e., to decrease the number of iteration steps needed.

We thus now have the iteration matrix $B = I - M^{-1}A$ and the method converges if $\rho(I - M^{-1}A) < 1$. The rate of convergence increases with the decreasing spectral radius $\rho(I - M^{-1}A)$. The iterative method (5.30) is an example of a preconditioned iterative method.

A straightforward computation shows [100] that all the three classical iterative methods are particular cases of the preconditioned simple iteration (5.30). We obtain the Jacobi method for $M = D$, the Gauss-Seidel method for $M = D - E$, and the successive overrelaxation method for $M = \omega^{-1}D - E$. We recall that all these three matrices are triangular (D is even diagonal). They thus do not increase the order of the number of arithmetic operations in the respective iterative processes. The process of the solution of the system (5.42) is implicitly included in the formulae (5.39), (5.40), (5.41).

We can precondition other classical iterative methods and Krylov subspace iterative methods, e.g., the conjugate gradient method, as well. Now let M be a symmetric positive definite matrix and let it possess Properties M1 and M2. Then there exists a unique symmetric positive definite matrix $M^{-1/2}$

such that $M^{-1/2}M^{-1/2} = M^{-1}$ (see, e.g., [92]). Let us apply the conjugate gradient method (5.35) to the system

$$M^{-1/2}AM^{-1/2}(M^{1/2}x) = M^{-1/2}b \tag{5.45}$$

obtained from (5.1). Carrying out simple substitutions, we can achieve the result that the output of the final iterative process is the solution x of (5.1) sought. We thus choose an initial approximation x_0 and compute

$$\begin{aligned} r_0 &= b - Ax_0, \\ p_0 &= M^{-1}r_0, \\ a_{k-1} &= \frac{(r_{k-1}, M^{-1}r_{k-1})}{(p_{k-1}, Ap_{k-1})}, \\ x_k &= x_{k-1} + a_{k-1}p_{k-1}, \\ r_k &= r_{k-1} - a_{k-1}Ap_{k-1}, \\ b_{k-1} &= \frac{(r_k, M^{-1}r_k)}{(r_{k-1}, M^{-1}r_{k-1})}, \\ p_k &= M^{-1}r_k + b_{k-1}p_{k-1}, \quad k = 1, 2, \dots . \end{aligned} \tag{5.46}$$

As compared with the original method (5.35), we have to find the vector $y = M^{-1}r_k$ in each step of the preconditioned method (5.46). We compute it as the solution of an auxiliary system $My = r_k$ (cf. Property M1). By the conjugate gradient method, we solve the system (5.45) with the matrix $M^{-1/2}AM^{-1/2} = I - M^{-1/2}EM^{-1/2} = I - \tilde{E}$ (see (5.43)), that is similar to $M^{-1}A$, instead. Due to positive definiteness, its condition number is then

$$\varkappa(M^{-1/2}AM^{-1/2}) = \frac{\lambda_{\max}(I - \tilde{E})}{\lambda_{\min}(I - \tilde{E})}.$$

It is thus close to 1 if $\lambda_{\max}(\tilde{E})$ is small (Property M2) since $\lambda_{\min}(I - \tilde{E}) = 1 - \lambda_{\max}(I - \tilde{E})$.

The biconjugate gradient method of Paragraph 5.2.2 can also be preconditioned in the same way.

There is also a preconditioner based on one step of the SSOR method (mostly used with the CG algorithm) [11, 137].

We have seen some particular examples of the preconditioner M. The choice $M = D$ is very general and it is efficient for many iterative methods and many classes of linear algebraic systems. Another choice of a preconditioner is possible if the matrix A of the system differs only negligibly from a matrix M that satisfies the assumptions for the application of the cyclic reduction method described in Paragraph 5.1.4. The process of solving the system (5.42) is then fast.

We further present a very often used preconditioning method called *incomplete factorization* [137]. Let the matrix A of order n of our model system have the structure schematically shown in Figure 5.3, i.e., let it be sparse with nonzero entries placed on five diagonals. Such a matrix arises, e.g., from the

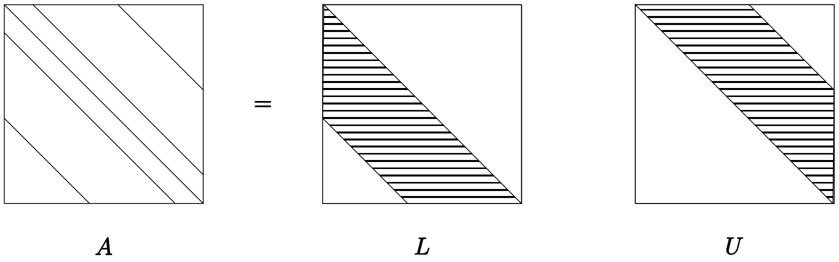

FIGURE 5.3: LU factorization of a sparse matrix A. Nonzero entries of A lie on the shown five diagonals only, nonzero entries of L and U in the indicated bands. The other entries of the matrices are zero.

discretization of a second-order differential problem on a rectangular domain in a regular rectangular grid. Considering A as a bandmatrix of bandwidth $2m + 1$ (where m depends on n, cf. Paragraph 5.1.2), we obtain, by the factorization from Paragraph 5.1.1, the triangular bandmatrices L and U that are filled. They are also shown in Figure 5.3. In general, all entries in their bands may be nonzero.

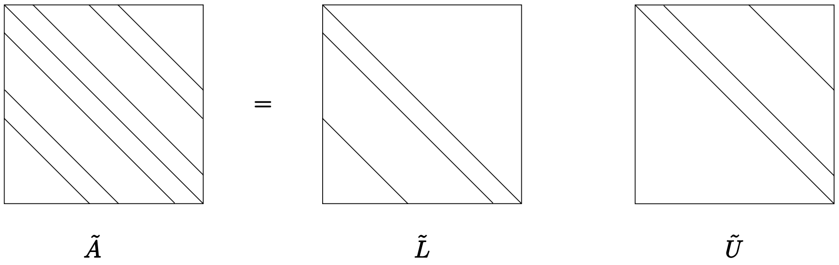

FIGURE 5.4: Incomplete factorization. Nonzero entries of the factors $\tilde{L}$ and $\tilde{U}$ lie on the shown diagonals but the product $\tilde{A} = \tilde{L}\tilde{U}$ differs from the original matrix A.

Conversely, we will now start with the desired form of the factors (Figure 5.4). Let nonzero entries of these factors (denoted by $\tilde{L}$ and $\tilde{U}$) be located only in places where there are nonzero entries of A (and only in the lower and upper triangle of the respective factor). We keep the assumption that the diagonal of $\tilde{L}$ consists of 1's only. Multiplying $\tilde{L}$ and $\tilde{U}$, we obtain a matrix $\tilde{A}$ different from A. In our particular case, nonzero entries of $\tilde{A}$ are placed on seven diagonals. The product $\tilde{A} = \tilde{L}\tilde{U}$ is called the *incomplete factorization of* A.

The factors $\tilde{L}$ and $\tilde{U}$ are determined from A by the standard factorization algorithm of Paragraph 5.1.1 but we only compute their entries that were declared nonzero in advance. The other entries of $\tilde{L}$ and $\tilde{U}$ involved in the formulae (5.4) are not computed but set equal to zero. The result of incomplete factorization may naturally depend on the order in which the operations are carried out. The formula (5.4) is the "columnwise" version of complete factorization. There is also a "rowwise" version that gives identical results in the case of complete factorization but results of this version used for incomplete factorization may be completely different.

The computation of the incomplete factorization of the matrix considered requires of order n arithmetic operations, the solution of the systems $\tilde{L}y = c$ and $\tilde{U}x = y$ also of order n operations. Putting $M = \tilde{A} = \tilde{L}\tilde{U}$, we have the "complete" factorization of M and we thus can solve (5.42) by a fast direct method. Therefore, the matrix M possesses Property M1 required for preconditioning.

In general, we usually know very little about the matrix E of (5.43) from the theoretical point of view. Regardless, the preconditioning based on the incomplete factorization is used very often and its effect is, as a rule, very good. We may admit even more nonzero entries in $\tilde{L}$ and $\tilde{U}$ than in the above example to make the norm of E smaller. We can also factorize general sparse matrices incompletely or, if the matrix A is symmetric, construct an *incomplete Choleski factorization* (cf. (5.6)). There are also procedures that compensate neglected entries of the matrices $\tilde{L}$ and $\tilde{U}$ on the diagonal of the matrix $\tilde{U}$ in some way (*modified incomplete factorization*) [11, 100].

5.2.5 Block iterative methods

For several reasons, it may be advantageous to split the matrix A of the system (5.1) into blocks (and the vectors of the right-hand part and unknowns in the corresponding way, too). In fact, we used this approach (cf. equation (5.12) in Paragraph 5.1.4) in the explanation of the cyclic reduction method.

The splitting of the matrix B of the system (5.12) into blocks may be useful not only with direct methods as in the case of cyclic reduction. There are a lot of iterative methods in which an iteration step can be carried out very efficiently if the matrix of the system is split and considered as a block matrix. This treatment of the matrix of the system may even enable us to perform the individual steps of an iterative method in a parallel way if such a computer architecture is available [137, 169, 197]. Some software packages take this fact into account, see, e.g., [10].

As an example, we present the *successive line overrelaxation method* investigated in [199] that is based on the Gaussian elimination for a tridiagonal system (double sweep method, see Paragraph 5.1.2). Let us consider equation (5.12) where now its block form is

$$\begin{bmatrix} A_1 & B_1 & & & \\ C_2 & A_2 & B_2 & & \\ & \ddots & \ddots & \ddots & \\ & & C_{N-1} & A_{N-1} & B_{N-1} \\ & & & C_N & A_N \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \\ u_N \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{N-1} \\ v_N \end{bmatrix}.$$

The square diagonal blocks A_j are of the order M_j where M_j corresponds to the number of grid nodes in the jth horizontal grid line of a discretization of an elliptic problem in a regular grid on a rectangular domain.

Let us derive the successive line overrelaxation method as a block analog of the SOR method (5.41). Start with the Jacobi method (5.39). Since it can be obtained from the simple iteration method (5.30) by setting the preconditioner M equal to the diagonal of the matrix A of the system, let us now set M equal to the block diagonal of the matrix of the system,

$$M = \begin{bmatrix} A_1 & & & & \\ & A_2 & & & \\ & & \ddots & & \\ & & & A_{N-1} & \\ & & & & A_N \end{bmatrix}.$$

We then get the block iteration matrix in the form

$$I - M^{-1}A = \begin{bmatrix} 0 & A_1^{-1}B_1 & & & \\ A_2^{-1}C_2 & 0 & A_2^{-1}B_2 & & \\ \ddots & & \ddots & \ddots & \\ & A_{N-1}^{-1}C_{N-1} & 0 & A_{N-1}^{-1}B_{N-1} \\ & & A_N^{-1}C_N & 0 \end{bmatrix}.$$

Introducing the relaxation parameter ω, we arrive at the block form of the SOR method. Choosing arbitrary initial vectors $u_j^{(0)}$, $j = 1, \ldots, N$, we compute for $k = 0, 1, \ldots$

$$u_j^{(k+1)} = \omega A_j^{-1}(-C_j u_{j-1}^{(k+1)} - B_j u_{j+1}^{(k)} + v_j - A_j u_j^{(k)}) + u_j^{(k)} \quad \text{for} \quad j = 1, \ldots, N,$$

or, algorithmically more conveniently,

$$A_j w_j^{(k+1)} = -C_j u_{j-1}^{(k+1)} - B_j u_{j+1}^{(k)} + v_j, \quad \text{for} \quad j = 1, \ldots, N, \tag{5.47}$$

and

$$u_j^{(k+1)} = \omega(w_j^{(k+1)} - u_j^{(k)}) + u_j^{(k)}, \quad \text{for} \quad j = 1, \ldots, N.$$

Obviously, the algorithm involves solving the family of systems (5.47) with the matrices A_j. If these matrices are tridiagonal and the matrices C_j and B_j are sparse, this process is fast, i.e., each iteration step is fast. It is possible to show [199] that the convergence rate can be improved for some classes of matrices by the application of a block iterative method.

5.2.6 Multigrid methods

Multigrid methods are a very general and efficient way of solving boundary and initial-boundary value problems for partial differential equations and some additional problems. They can be applied to linear as well as nonlinear problems. It is a particular case of *multilevel adaptive techniques* that can be characterized as a combination of the process of discretization of the problem and the process of solving the discrete problem, i.e., the corresponding system of linear algebraic equations.

We show the basic principle of the multigrid method in the case of two grids. We only introduce the very essential facts about the method and try to explain the foundations of a very general *algebraic multigrid method.* For the multigrid method, discretizations on several grids with different grid steps carried out by the finite element or finite difference methods are typical, see, e.g., [104]. It is an iterative method and its final result is an approximation of the solution of the continuous problem on the finest grid. The method employs the fact that approximate solutions constructed on different grids have a common property: they more or less exactly approximate the true solution. An advantage of this approach is that computational work is gradually moved to coarser and coarser grids where fewer operations are needed. If the multigrid method is considered as a way of solving linear algebraic systems it is a fast method.

An important phenomenon that has led to the idea of the multigrid method was the observation that classical matrix iterative methods of Paragraph 5.2.4 converge very fast in several first iterations and the error decreases rapidly. Then the convergence slows down and the error stagnates. The conclusion is that classical iterative methods are very efficient when used to smooth the error. Moving to a coarser grid, we can again use a classical iterative method to smooth the error efficiently there.

There is a vast literature about the multigrid method, e.g., [11, 24, 69, 100, 104, 134, 137, 159]. We confine ourselves to the principal algorithm of the two-grid method which will be used for an adaptive finite element strategy in Chapter 6, and we do not study its convergence.

We demonstrate the essence of the two-grid method on a simple model example of a boundary value problem for a partial differential equation

$$Lu = f, \tag{5.48}$$

where L is a linear elliptic differential operator, in a plane domain Ω with the corresponding boundary conditions. We discretize the problem (5.48) by the finite element (or finite difference) method on two grids, the fine grid with a characteristic discretization parameter h and the coarse one with a parameter H, $h < H$. Characteristic parameter of the grid is, e.g., the maximum length of a triangle side in the triangulation of the domain. We arrive at two discrete problems,

$$L_h u_h = f_h, \tag{5.49}$$

$$L_H u_H = f_H, \tag{5.50}$$

and assume that the discrete operators (matrices) L_h and L_H also include the discretized form of the boundary conditions. The equations (5.49), (5.50) are then systems of linear algebraic equations, and L_h and L_H are square matrices whose order will be denoted by n and m, $n > m$. Further, u_h and f_h are n-component vectors, and u_H and f_H m-component vectors.

The multigrid method consists of three basic procedures. The first, *smoothing* (or *relaxation*) on the individual levels, is usually represented by several steps of some classical iterative method, very often the Gauss-Seidel method.

As we move from a fine grid to a coarse one and back during the multigrid process we need an operator that maps a grid function defined on the fine grid to a grid function defined on the coarse grid and an operator that reversely maps a function on the coarse grid to the fine one. The first operator is $r : R^n \to R^m$ and is called a *restriction*. A simple example of the restriction is the *injection* that can be easily employed if the set of nodes of the coarse grid is a subset of nodes of the fine grid. Then it is sufficient to take, for the value of the grid function at a node of the coarse grid, the value of the function, defined on the fine grid, at the same node.

Better properties are possessed by the restriction called the *averaging* that takes for the value of a grid function at a node of the coarse grid an average of values of the function at neighboring nodes of the fine grid calculated in some way. The operator r is apparently a rectangular $m \times n$ matrix.

The operator $p : R^m \to R^n$ that assigns a grid function defined on the fine grid to the function defined on the coarse grid is called the *prolongation* or *interpolation*. In practice, the operator p is usually some interpolation of the values of the grid function at the nodes of the coarse grid. The operator p is apparently a rectangular $n \times m$ matrix.

The choice of the particular operators r and p may influence the performance of the multigrid method significantly. The choice $p = r^T$, i.e., the prolongation as an adjoint operator to the restriction, is used very frequently.

The two-grid method is an iterative method for the calculation of the unknown vector u_h. It starts from an initial approximation u_h^0 (e.g., equal to zero). We describe now one step of the two-grid method that computes a new approximation u_h^{i+1} to the solution from the old approximation u_h^i. The iteration step presented here is used for linear problems and has to be modified if the problem is nonlinear. The procedure is shown in Figure 5.5.

$$\begin{tikzcd}
u_h^i \arrow[r] & \tilde{u}_h^i = S(u_h^i) \arrow[d] & & & u_h^{i+1} = S(\tilde{\tilde{u}}_h^i) \\
& d_h = L_h\tilde{u}_h^i - f_h \arrow[d] & & v_h = pv_H \arrow[r] & \tilde{\tilde{u}}_h^i = \tilde{u}_h^i - v_h \arrow[u] \\
& d_H = rd_h \arrow[rr] & & L_H v_H = d_H \arrow[u] &
\end{tikzcd}$$

FIGURE 5.5: One iteration step of the two-grid method.

We start on the fine grid. We first apply a few relaxation steps that smooth the error and that are symbolically written in the form $\tilde{u}_h^i = S(u_h^i)$. On the fine grid, we then compute the vector

$$d_h = L_h\tilde{u}_h^i - f_h, \tag{5.51}$$

called the *defect* of the solution. In fact, it is, except for the sign, the residual of the linear algebraic system solved.

If the defect is zero the system is solved exactly. We thus try to calculate the *correction*, i.e., the vector that subtracted from $\tilde{u}_h^i$ gives an approximate solution with minimum defect. The ideal situation would be to compute the exact correction v_h from the equation

$$L_h v_h = d_h \tag{5.52}$$

as we can easily verify that substituting the vector $\tilde{u}_h^i - v_h$ into the original equation (5.49), we obtain the exact solution since we have $L_h(\tilde{u}_h^i - v_h) = d_h + f_h - d_h = f_h$ from (5.51), (5.52). However, to solve the equation (5.52) is as time consuming as solving the original equation (5.49). This is thus the reason to move to a coarse grid and approximate the correction on this coarse grid using fewer arithmetic operations.

The transfer to a coarse grid is denoted in Figure 5.5 by the arrow pointing downward. All the operations on the coarse grid are presented on the bottom line of the scheme. This is a widely used convention in the multigrid method. On the coarse grid, we first establish a restriction of the defect $d_H = rd_h$ and then compute the correction v_H on the coarse grid as a solution of the system of m linear algebraic equations

$$L_H v_H = d_H. \tag{5.53}$$

The matrix L_H of the system is given by the discretization (5.50) of the problem on the coarse grid. However, the system (5.53) possesses the right-hand part d_H (defect) and we look for the vector v_H (correction).

The arrow pointing upward represents the return to the fine grid. To this end, we have to prolong the correction $v_h = pv_H$ and subtract it from the actual approximation of the solution on the fine grid. We obtain an improved approximation $\tilde{\tilde{u}}_h^i = \tilde{u}_h^i - v_h$. Finally we smooth the error of this approximation since the transfer of the correction from the coarse to the fine grid again introduced high-frequency components of the error to be removed. The new approximation to the solution is then $u_h^{i+1} = S(\tilde{\tilde{u}}_h^i)$.

We have just described an iteration step of the two-grid method that consists of the computation of the defect on the fine grid, its restriction to the coarse grid, computation of the correction on the coarse grid, and its prolongation to the fine grid. In the beginning as well as the end of the iteration step, moreover, the approximate solution is relaxed, i.e., the error smoothed. It is substantial that in this relaxation, only very few steps of classical matrix iterative methods are performed (say, one or two) since the aim of this procedure is to smooth the error, not to minimize it.

The procedure shown is not the only possibility. Since it is based on the computation of the correction it is called the *correction scheme*. Its algorithm in Figure 5.5 employs the linearity of the operator L. In the nonlinear case, it is also necessary to restrict the approximate solution $\tilde{u}_h^i$ to the coarse grid (*full approximation scheme*).

In the multigrid method, it is necessary to solve a linear algebraic system for the correction, whose matrix arises in discretization of the continuous problem, on the coarser grid (lower level). This system can again be solved with the help of the two-grid procedure; however, we need another, still coarser grid and a discretization on it.

If we employ more than two grids we move from fine to coarser grids and back from coarse to finer grids. This procedure is called *cycling* and can be carried out in different ways. We usually start with an approximation of the solution on the finest grid and finally obtain such an approximation of the continuous solution on the finest grid that possesses the accuracy required.

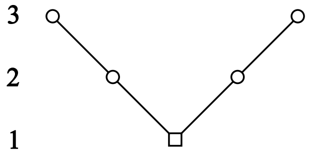

FIGURE 5.6: V-cycle on three grids.

We briefly describe the simplest cycling called *V-cycle* on three grids (Figure 5.6). We apply several relaxation steps to the initial approximation to the solution on the finest grid (the highest level 3 in the figure), compute the defect of the solution and restrict it to the coarser grid (level 2) to solve the equation for the correction of the solution. We apply several relaxation steps to the initial approximation to the correction of the solution, compute the defect of the correction of the solution and restrict it to next coarser grid (level 1) that is in Figure 5.6 the coarsest one. We compute here the correction of the correction (solve the corresponding system (5.53)) either by a direct method or, e.g., by the same iterative method we employ as a relaxation (this time, however, we carry out the full number of iteration steps needed to get a good approximation). This operation is denoted by a square in Figure 5.6 while current operations on the individual levels are denoted by a circle.

The correction of the correction is prolonged to the finer grid (level 2) and subtracted from the correction of the solution. This improved correction of the solution is relaxed, prolonged to a finer (in our case already the finest) grid (level 3) and subtracted from the solution. Finally, we apply several relaxation steps to the solution just obtained.

The procedure described above resembles the letter V in Figure 5.6 and this is the reason for its name. Just one V-cycle is usually enough for obtaining a sufficiently accurate solution of a very large class of systems (5.49). If we need a more accurate solution we can carry out a further V-cycle. For linear problems, we can use the zero initial approximation.

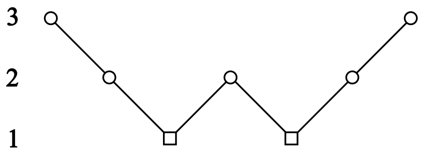

FIGURE 5.7: W-cycle on three grids.

A little bit more complicated strategy of the multigrid method is shown for three grids in Figure 5.7. When we first time move to level 1 and return to level 2 we immediately move back to level 1 and compute the correction there again. The rest of the cycle is the same as in the previous case. This strategy is called the *W-cycle*, again because of the resemblance of the figure to the letter W. It is easy to derive the W-cycle for more than three grids. The W-cycle requires more arithmetic operations than the V-cycle but its efficiency is substantially better. There are further strategies that include the return to the lowest level (i.e., to the coarsest grid) even more frequently than the W-cycle; they could be called "triple-V-cycle," etc.

If we look for an initial approximation to the solution on the finest grid it is quite natural to start with an initial approximation on a coarser grid and prolong it to the finest grid. If we start with an initial approximation on a grid that is several levels lower we can improve it gradually on the way to the finest grid by using standard multigrid steps. This idea finally leads to the *full multigrid method* and it is shown in Figure 5.8.

We start with an initial approximation on the coarsest grid and gradually carry out longer and longer cycles until we reach the finest grid. Usually, the solution of the system (5.49) is then accurate enough. If not, we can continue by cycling on the sequence of grids we have used up to now or construct the next, even finer, grid and add it to this sequence.

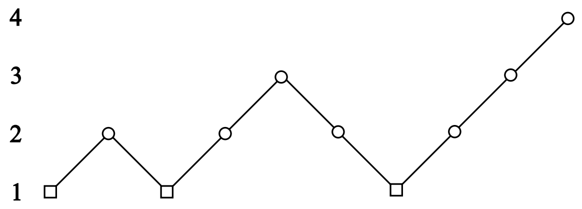

FIGURE 5.8: Full multigrid method on four grids.

We explained the multigrid method as a procedure that originates in a boundary value problem and its several discretizations on a sequence of grids. The reverse approach is also possible. We can start from a sparse linear algebraic system, consider it as a discretization of a continuous boundary value problem, and employ the multigrid method based on it. The computational process need not explicitly include any sequence of discretization levels (grids). The only aim of this process is to solve a linear algebraic system and its name is the *algebraic multigrid method* [100].

There are several strategies of the algebraic multigrid method, too. The three procedures included in the algebraic multigrid method are the same as in the previous case: relaxation, restriction, and prolongation. The usual restriction and prolongation methods are the *aggregation* and *disaggregation* of unknowns. The aggregation r is an analog of averaging the values of the solution at neighboring nodes. The disaggregation is then an analog of the interpolation and is often given by the transpose $p = r^T$ [100].

A good and reliable example of multigrid software is the PLTMG (Piecewise Linear Triangular MultiGrid) package [24] available on the web. It solves a boundary value problem for a second-order nonlinear elliptic partial differential equation on a 2D domain. The package is written in FORTRAN. The boundary value problem is (if necessary) linearized by the damped Newton method [25] and then discretized by the finite element method using triangular elements and piecewise linear approximation. The package computes an a posteriori estimate of the error of the solution calculated and can construct finer grids adaptively. The grid is made finer or coarser only locally, where the error estimator shows. This is the way to establish a sequence of grids for the multigrid method efficiently. The final output of the PLTMG package is the approximate solution on the finest grid constructed.

5.3 Choice of the method

Matrices of the systems that are most often met in practical problems fall into one of two large classes:

1. Full (filled or dense) matrices of moderate order (say, $n < 100$).
2. Sparse matrices of high and very high order (n equal to several hundred, several thousand, even several million is no exception).

Solving finite element problems, we naturally arrive at matrices of the second class. There is no simple rule for the choice of the method in this case. It may be hard to store a sparse matrix of order 1 000 or 1 000 000 (including its zero entries) and thus the use of both direct and iterative methods is usually accompanied by certain programming tricks.

When software for the solution of large linear algebraic systems is selected some quality criteria should be applied. Programs used for solving such linear systems, the characteristics of which are not completely and exactly known in advance, have to be highly *robust*. They have to be able to recognize ill-conditioned or numerically singular systems and provide the user with adequate information (condition and/or accuracy estimates, etc.).

Efficiency is of such great importance when solving very large linear algebraic systems that often the practical solvability of a problem depends on the degree of exploitation of the computer resources available. For very large sparse matrices, not only the computational efficiency but also the storage efficiency is of critical importance.

Usability of the software available is worth considering. Software for solving linear algebraic systems (e.g., LAPACK programs [10]) is generally quite user-friendly. Only some kinds of problems and classes of algorithms can cause difficulties. For example, programs designed for the iterative solution of large sparse systems are not suitable for "black box" applications as the user must have preliminary knowledge of initial approximations, parameters of the algorithm, or preconditioning.

The highest possible degree of *portability* together with good efficiency distinguish the programs of LAPACK whose excellent qualities make it the best software product for standard use. For large sparse linear algebraic systems, portability is provided primarily by the different storage schemes (cf. Paragraph 5.1.3).

If some fast direct method (Paragraphs 5.1.2, 5.1.4) is applicable then it certainly pays. Efficient employment of the LU factorization to sparse matrices with nonzero entries placed at random (Paragraph 5.1.3) has to consist of three steps: minimization of the fill-in (it is done once for a particular zero-nonzero structure), factorization of the matrix (it is performed once for particular numerical values of entries), and solution of the system (it is carried out for every right-hand part). This procedure as a whole is relatively time consuming. It may be advantageous if the result of minimizing the fill-in can be exploited for more matrices of the same zero-nonzero structure or if we solve several systems that differ in right-hand parts only.

All iterative methods involve the risk of terminating the process too early and thus getting a solution that is not accurate enough. However, this may be, in some cases, an asset. If we are interested in a less accurate solution (e.g., in its two significant digits), with iterative methods we need not expend unnecessary labor on solving the system accurately (except for roundoff error) by a direct method. We must be very cautious if we use iterative methods in which a parameter or parameters to be chosen occur. Taking a wrong value of parameter (substantially different from the optimal value), we may decelerate the convergence of the method. In some cases (cf. the SOR method in Paragraph 5.2.4) the determination of the optimal iteration parameter may lead to a more complex problem than solving the linear algebraic system.

The CG method can be recommended for symmetric positive definite systems while the restarted GMRES method is widely used for general systems.

The conjugate and biconjugate gradient method (Paragraph 5.2.2), GMRES method (Paragraph 5.2.3) as well as classical iterative methods (Paragraph 5.2.4) should never be used without preconditioning. The incomplete factorization preconditioner (Paragraph 5.2.4) can practically always be constructed and it requires relatively few arithmetic operations.

There are many special iterative methods for large sparse systems obtained from the discretization of boundary value problems for differential equations. These methods are usually derived with regard to the original continuous problem, are very sophisticated, and are not discussed here, see, e.g., [69, 120, 92, 100, 127, 197]. Their special features make them, as a rule, very efficient. We recommend the multigrid method (Paragraph 5.2.6).

The user always must be aware of the danger of accumulation of roundoff error in the individual methods. This problem usually does not occur if the matrix of the system is positive definite.

Let us mention some related software. We presented some websites where software can be found in the introduction to this chapter. Most programs and packages are written in FORTRAN. It definitely pays to look for a suitable program in the software available. In rather exceptional cases, it is necessary to code one's own program. General numerical packages like Maple, Matlab or Mathematica usually are not efficient when applied to very large sparse systems.

A basic source of software is LAPACK (Linear Algebra Package [10]) whose ancestor is the famous handbook [206] edited by Wilkinson and Reinsch. The algorithms in [206] are presented in Algol 60 language. LAPACK provides very efficient procedures. SPARSPAK [88] is a specialized package for sparse positive definite systems. The FISHPACK package (cf. [190]) is a realization of the cyclic reduction method (Paragraph 5.1.4). ITPACK [105] contains software modules for the iterative solution of linear algebraic systems.

An example of the multigrid technique is PLTMG [24] already mentioned in Paragraph 5.2.6. Numerous algorithms are included in Numerical Recipes

[158] (not only in FORTRAN). For further software, see, e.g., [197]. New methods have recently been developed to be used on parallel computers (see, e.g., [92, 169, 197]).

5.4 Solving initial value problems for ordinary differential equations

Most tools presented in this book up to now were concerned with solving elliptic partial differential equations. They may be used for solving evolutionary (parabolic, hyperbolic) equations as well if they are combined with a proper time discretization, e.g., a proper procedure for the solution of an initial value problem for a system of ordinary differential equations (ODEs).

We first present some types of initial value ODE problems. To motivate this section, we formulate a simple model linear parabolic problem and show how the method of lines transforms this problem into a system of ODEs in Paragraph 5.4.1. We then give numerical methods for its solution and the corresponding software.

Let us consider a general formulation of an initial value problem for an ODE system. On an interval $[T_0, T_1]$, we look for the solution $y(t) = (y_1(t), \dots, y_N(t))^T$, a vector function of N components, that fulfils a differential system on the interval $(T_0, T_1]$ and, moreover, satisfies the *initial condition*

$$y(T_0) = y^0$$

at the point T_0 where y^0 is a given vector of N components. The *system of ordinary differential equations of the first order* can be given in the simplest, *explicit form*

$$\dot{y}(t) = f(t, y), \tag{5.54}$$

where

$$\dot{y}(t) = \frac{\mathrm{d}y}{\mathrm{d}t},$$

the derivative of the solution y sought, only appears on the left-hand part whereas f on the right-hand part is a given vector function of N components. A further, *semi-implicit form*

$$A(t, y)\dot{y}(t) = f(t, y) \tag{5.55}$$

is more complex. The derivative of the solution on the left-hand part is multiplied by a given square matrix A of order N. Finally, the most complex is the *implicit form*

$$F(t, y, \dot{y}) = 0, \tag{5.56}$$

where F is a given vector function of N components.

The implicit form (5.56) is the most general and also includes forms (5.55) and (5.54). The form (5.55) can be transformed into the form (5.54) if the matrix A is nonsingular. For practical reasons, it is more advantageous to directly employ the software available to the system in the form (5.55) in this case. If the matrix A in (5.55) is singular and in the case (5.56) the system is, in general, *differential-algebraic*. It is then usually called and abbreviated DAE (differential-algebraic equations) [37].

In applications, it is certainly advantageous to consider the differential system in the simplest possible form and to use the corresponding software since it is less complex and thus brings time savings in practical computation.

Any system of differential equations of order higher than 1 can be transformed into a first-order system (of more equations) by proper substitutions.

The first numerical methods successfully used to solve differential-algebraic equations were linear multistep methods such as *backward differentiation formulae* (BDF) treated in Paragraph 5.4.2. Recently, one-step methods such as implicit Runge-Kutta and extrapolation methods are employed, too. We mention one-step methods briefly in Paragraph 5.4.3.

5.4.1 Method of lines

We consider a linear one-dimensional parabolic partial differential equation

$$\dot{u}(x,t) = (A(x)u'(x,t))' - B(x)u(x,t) + f(x,t), \quad 0 < x < 1, \quad 0 < t \le T \tag{5.57}$$

for some fixed $T > 0$, where x is called *space variable* and t *time variable*. We further prescribe the following conditions:

$$u(0,t) = u(1,t) = 0, \quad 0 \le t \le T \tag{5.58}$$

is the boundary condition and

$$u(x,0) = u_0(x), \quad 0 < x < 1 \tag{5.59}$$

the initial condition. We use the notation

$$u'(x,t) = \frac{\partial u}{\partial x}(x,t), \qquad \dot{u}(x,t) = \frac{\partial u}{\partial t}(x,t)$$

for space and time derivatives. A nonlinear parabolic eqution can be treated in an analogous way.

We assume that the above problem is parabolic, i.e., that $A(x) \ge A > 0$ is a positive smooth function and $B(x) \ge 0$ a nonnegative one.

We employ the usual Sobolev spaces $H^1(0,1)$ and $H_0^1(0,1)$, and introduce a weak solution of the model problem. We set

$$a(v,w) = \int_0^1 (v'A(x)w' + vB(x)w)\ \underset{\cdot}{\mathrm{x}}$$

and denote the usual L^2 inner product of functions v, w by (v,w). We then say that a function $u(x,t)$ is the *weak solution* of the problem (5.57), (5.58), (5.59) if it belongs, as a function of the variable t, in $H^1([0,T], H_0^1(0,1))$, if the identity

$$(\dot{u}, v) + a(u,v) - (f,v) = 0$$

holds for each $t \in (0,T]$ and all functions $v = v(x) \in H_0^1$, and if the identity

$$a(u,v) = a(u_0, v)$$

holds for $t = 0$ and all functions $v \in H_0^1$. Note that time t is considered a parameter in this formulation as well as in what follows.

There are three possibilities to carry out the discretization of the parabolic problem considered. *Full discretization* (i.e., simultaneously in space as well as time) is often used with the help of many various numerical procedures. The two remaining possibilities start with semidiscretizations: first a discretization in time, and then application of a numerical method to solve the resulting space dependent problem (the *Rothe method*) or first a discretization in space, and then application of a numerical method to solve the resulting time dependent problem (the *method of lines* [198, 194], cf. also Paragraph 1.1.7). We use the method of lines approach to solve the parabolic problem in what follows.

Finite element solutions of the model problem are then constructed from the weak formulation. Fixing a positive integer p, we can introduce finite dimensional subspaces $S_0^{N,p} \subset H_0^1$ of hierarchic basis functions (used as test functions as well) where p is the maximal degree of the piecewise polynomial basis functions and N is the total number of these basis functions.

We say that a function

$$U(x,t) = \sum_{j=1}^{N} U_j(t)\varphi_j(x) \tag{5.60}$$

is the *semidiscrete finite element approximate solution* of the model problem if it belongs, as a function of the variable t, into $H^1([0,T], S_0^{N,p})$, if the identity

$$(\dot{U}, V) + a(U,V) - (f,V) = 0 \tag{5.61}$$

holds for each $t \in (0,T]$ and all functions $V \in S_0^{N,p}$, and if the identity

$$a(U,V) = a(u_0, V) \tag{5.62}$$

holds for $t = 0$ and all functions $V \in S_0^{N,p}$.

Substituting (5.60) into (5.61) and (5.62), we get an initial value problem for a system of N ODEs for the N unknown functions $U_j(t)$ with the initial condition given by a system of linear algebraic equations. The procedure just described is the method of lines mentioned above. This resulting ODE system is of the form (5.55) and is *stiff* (cf., e.g., [37]). In practice, it is solved by proper numerical software, i.e., a proper ODE solver. The error tolerance for the time integration required by the user makes the solver proceed from a time level to another one. These time levels are called *natural time levels*. We can a posteriori evaluate the space discretization error of the approximate solution and employ an adaptive procedure for updating space mesh on each natural time level. The process can be carried out for linear as well as nonlinear initial-boundary value parabolic problems.

It is usually assumed that the "time" error tolerance is set so low that we need not consider this time discretization error as compared with the space discretization one. Recently, papers also taking into account the time discretization error have appeared (see, e.g., [201]).

5.4.2 Multistep methods

Historically, the first general technique for the numerical solution of DAEs was proposed by Gear [87] in 1971 and is based on the BDF idea. It has been extended to any implicit system (5.56), cf. [198, 37, 157].

The simplest first-order BDF method is the implicit Euler method that consists of replacing the derivative in (5.56) by a backward difference

$$F\left(t_n, y_n, \frac{y_n - y_{n-1}}{h}\right) = 0,$$

where $h = t_n - t_{n-1}$. The resulting system of nonlinear equations for y_n on each time level is then usually solved by the Newton method. The k-step (constant stepsize) BDF consists of replacing $\dot{y}$ by the derivative of the polynomial, which interpolates the computed solution on $k+1$ time levels $t_n, t_{n-1}, \ldots, t_{n-k}$, evaluated at t_n. This yields

$$F\left(t_n, y_n, \frac{\rho y_n}{h}\right) = 0,$$

where

$$\rho y_n = \sum_{i=0}^{k} \alpha_i y_{n-i}$$

and α_i, $i = 0, 1, \ldots, k$, are the coefficients of the BDF method. It can be shown [37] that the k-step BDF method is stable for ODEs for $k < 7$.

Following the paper by Gear [87], several codes implementing the BDF methods were written in the 1970s. A second generation of BDF implementations began to emerge in the early 1980s, along with a growing recognition of the importance of DAEs in many scientific and engineering applications. These are the codes DASSL [154] and LSODI [114].

Any method to be used in applications has to first be implemented in codes that are efficient, robust, user-friendly, portable, and well documented. The most widely used production codes for differential-algebraic (and ordinary differential) equations are based on BDF methods. A detailed description of the DASSL code designed for solving initial value problems of the form (5.56), including theoretical considerations, is contained in [37].

The software mentioned also includes, as an input parameter, the error tolerance required for the approximate solution and tries, employing a posteriori error estimates and decreasing the stepsize h if necessary, to yield the solution as accurately as prescribed. If this is impossible the user is provided with the corresponding information. Many problems have been successfully solved using the codes mentioned, thereby encouraging the BDF approach.

5.4.3 One-step methods

The order, stability, and convergence properties of one-step methods when applied to the system (5.56) are studied in [37]. *Implicit Runge-Kutta* (IRK) methods are of particular interest.

An M-stage IRK method applied to the equation (5.56) is given by

$$F\left(t_{n-1}+c_i h, y_{n-1}+h\sum_{j=1}^{M} a_{ij}\dot{Y}_j, \dot{Y}_i\right)=0, \quad i=1,2,\dots,M \tag{5.63}$$

and

$$y_n = y_{n-1}+h\sum_{i=1}^{M} b_i\dot{Y}_i, \tag{5.64}$$

where $h = t_n - t_{n-1}$. The quantities $\dot{Y}_i$ in (5.63), (5.64) are estimates for $\dot{y}(t_{n-1}+c_i h)$ and are called *stage derivatives.* Estimates for $y(t_{n-1}+c_i h)$ may be obtained by defining intermediate Y_is as

$$Y_i = y_{n-1}+h\sum_{j=1}^{M} a_{ij}\dot{Y}_j.$$

Recently, IRK methods have been the focus of increasing interest for the numerical solution of stiff ODEs. Due to their one-step nature, IRK methods are potentially more efficient because multistep methods have to be restarted at low order after every discontinuity, e.g., after the change of grid in adaptive procedures. Just IRK methods are often used to generate accurate starting values for higher-order BDF methods. Another potential advantage of RK methods applied to ODEs lies in the fact that, in contrast to the case of linear multistep methods, it is possible to construct high order A-stable IRK formulae (cf., e.g., [37]).

When solving a system by an IRK method, it is important to choose a class of methods that can be implemented efficiently. In the most general IRK method, when the square matrix $A = (a_{ij})$ of order M is completely dense and the method is applied to the system of N ODEs, we obtain a system of MN nonlinear algebraic equations that has to be solved for the stage derivatives at each integration step. Compared to the expense of a multistep method, the amount of work per step has to be reduced significantly before IRK methods can be competitive. Therefore, we are interested in particular classes of IRK formulae that can be implemented more efficiently than the general case.

If A is a lower triangular matrix the system of nonlinear equations to be solved at each step can be broken into M sets of N equations to be solved consecutively. IRK methods of this type are called *semi-implicit. Diagonally implicit* IRK methods or DIRK methods [37] are semi-implicit methods with equal diagonal entries in A. Finally, if the matrix A has one (real) eigenvalue of multiplicity M it has been shown that IRK can be implemented almost as efficiently as the DIRK methods. Such methods are called *singly implicit* IRK methods or SIRK methods [37].

Extrapolation methods may be viewed as IRK methods. They are thoroughly treated in [37].

Program packages for solving initial value problems for ODEs by IRK methods are not very common. We mention at least [70].

Chapter 6

Automatic mesh optimization, reference solutions and hp-adaptivity

Assuming that the reader is familiar with the basic principles of higher-order finite element discretization that were presented in Chapters 2 and 3, we can proceed now to more advanced topics related to automatic mesh optimization and adaptivity.

In this chapter we present a class of automatic goal-oriented h-, p-, and hp-adaptive strategies based on automatic optimization of the finite element mesh. This methodology was first proposed in [162] and further elaborated in [64, 62, 65]. Recently it was coupled with goal-oriented adaptivity in [185] with very promising results. The adaptivity is guided by a robust error indicator based on the difference $u_{ref} - u_{h,p}$ where $u_{h,p}$ is the solution on the current (coarse) mesh $\mathcal{T}_{h,p}$ and u_{ref} is a suitable *reference solution* on $\mathcal{T}_{h,p}$ (reference solutions are very accurate approximations of the exact solution on coarse grids based, e.g., on sophisticated postprocessing techniques, or obtained in some other "inexpensive" way). Our approach to the design of reference solutions will be discussed in Paragraph 6.2.1. The point is that by omitting conventional equation-specific error indicators the adaptive strategy is less sensitive to particular types of solved problems. The adaptive strategy itself is then based on automatic construction of a next optimal mesh via minimization of *projection-based interpolation error* of the reference solution u_{ref}. Projection-based interpolation operators introduced in Chapter 3 play an essential role in this kind of adaptive strategy.

It has been stated in [162, 64] and confirmed also by other sources (e.g., [19, 117, 185]) that the method is capable of delivering optimal convergence rates not only in the asymptotic sense, but also more importantly *in the preasymptotic range* of error. This is important for the solution of real engineering problems, since due to generally limited means (human resources, time, computing equipment...) one always achieves results on some limited level of accuracy only. The asymptotic level $h \to 0$, where various unknown constants C in theoretical convergence rates do not matter anymore, is *never reached* in practice.

Almost all commercial applications of hp-adaptive finite element schemes use a priori information about corner and edge singularities [15, 172] or boundary layers [135, 172] to design advantageous initial meshes. After this, either uniform or adaptive p-refinements are made. Such methods are very efficient if the nature of the singularities of the solution is known in advance, and can lead to exponential rates of convergence even without the use of higher-order discretizations (see, e.g., [118]).

However the situation changes if we do not have at our disposal a priori information about the solution. It is known that general initial meshes can lead p-adaptive schemes to less satisfactory results than standard h-adaptive methods. For many practical problems, quadratic elements combined with h-adaptivity offer a favorable balance between the quality of results and the complexity of implementation. Recall that with $p = 2$ in the H^1-conforming case we do not have bubble functions on triangles, tetrahedra and prisms, and only one bubble function appearing on quadrilaterals and hexahedra is orientation-independent.

There have been several attempts to correct inoptimal initial meshes by h-refinements, which lead to methods proposed in [62, 149, 147] performing h- and p-refinements interchangeably. But still, in general, the resulting meshes do not lead to optimal results.

We hope that this brief survey illustrated why we are convinced that a fully automatic method that works *without any a priori information about the solution* is a real breakthrough in the adaptive finite element solution of engineering problems.

We present the method in a mathematically precise but at the same time also intuitive way, in order to provide the reader with a deep understanding that he/she can use in his/her own applications. We confine ourselves to H^1-conforming schemes in this presentation, as for edge and face elements the principles are the same. It is convenient to split the presentation into several successive steps:

First, we introduce algorithms capable of automatic adjustment of the finite element mesh toward an *a priori given*, fixed function u (*mesh optimization algorithms*) in Section 6.1. This automatic adaptivity will be performed *off-line*, independently of the finite element scheme.

Next, in Section 6.2 we incorporate the automatic mesh optimization algorithms into the finite element scheme, utilizing the concept of reference solutions. *Goal-oriented adaptivity* will be implemented into the automatic adaptive strategies in Section 6.4, and the whole methodology will be extended into two spatial dimensions in Section 6.5.

6.1 Automatic mesh optimization in one dimension

In this section we will deal with a class of automatic h-, p- and hp-mesh optimization strategies capable of progressive minimization of projection-based interpolation error

$$err_k = \|u - \Pi^k_{h,p} u\|_{H^1_0} \tag{6.1}$$

of a function $u \in H^1(\Omega)$, where Ω is a one-dimensional domain, by means of a sequence of finite element meshes $\mathcal{T}^k_{h,p}$, $k = 1, 2, \dots$ (the one-dimensional projection-based interpolation operator $\Pi_{h,p}$ was introduced in Section 3.1). Depending on the type of adaptivity that one selects, the sequence of optimal meshes $\mathcal{T}^k_{h,p}$, $k = 1, 2, \dots$ will be obtained by successive h-, p- or hp-refinements.

We begin with an initial finite element mesh $\mathcal{T}^0_{h,p} = \{K_1, K_2, \dots, K_{N^{(0)}}\}$, consisting of $N^{(0)}$ disjoint subintervals K_i such that

$$\Omega = \bigcup_{i=1}^{N^{(0)}} \overline{K}_i.$$

Each element K_i is assigned an order of polynomial approximation $1 \le p_i = p(K_i)$.

6.1.1 Minimization of projection-based interpolation error

The basic idea of the algorithm is to determine the next optimal mesh in such a way that a controlled increase in the number of degrees of freedom brings the maximum decrease of the projection-based interpolation error. We proceed in two steps.

Step 1 – determining the optimal refinement type for all elements

The first step is local, done for each element independently of the rest of the mesh.

Step 1, h-adaptivity:

There are, of course, many possible ways a one-dimensional element can be h-refined. We confine ourselves only to subdivision into two equally long subelements, which generalizes most naturally into two and three spatial dimensions. With this simplification, we end up with only one refinement option per element.

By $\Pi_{coarse,i} u = \Pi_{h,i} u$ we denote the projection-based interpolant of the function u on a coarse mesh element K_i. According to Section 3.1, this is a $p(K_i)$th order polynomial matching the values of u at the endpoints of the element K_i. Similarly we define the projection-based interpolant $\Pi_{fine,i} u = \Pi_{h/2,i} u$ on the h-refined element as a function that matches u at both endpoints of K_i and at its midpoint, and which is a polynomial of the order $p(K_i)$ on both the element sons. We compute the projection-based interpolation error decrease rate

$$\triangle err_i = \|u - \Pi_{coarse,i}u\|_{H_0^1,K_i} - \|u - \Pi_{optimal,i}u\|_{H_0^1,K_i}$$

with $\Pi_{optimal,i} = \Pi_{fine,i}$ (obviously the single refinement option is optimal). As in this case polynomial spaces corresponding to the coarse and refined element are *nested* (h-refinement produces a polynomial space that contains the *whole* polynomial space corresponding to the original coarse element), this reduces to

$$\triangle err_i = \|\Pi_{fine,i}u - \Pi_{coarse,i}u\|_{H_0^1,K_i}. \tag{6.2}$$

Decrease of projection-based interpolation error for h-refinement is illustrated in Figure 6.1.

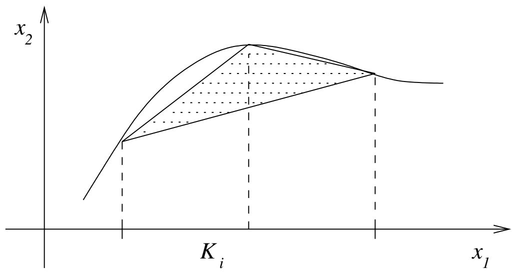

FIGURE 6.1: Decrease of projection-based interpolation error – h-refinement of a linear element.

Step 1, p-adaptivity:

For the sake of simplicity, in the case of p-refinements we also confine ourselves to a single refinement option – the order of polynomial approximation will only be allowed to increase by one per element in each step of the algorithm.

By $\Pi_{coarse,i}u = \Pi_{p,i}u$ we denote the projection-based interpolant of the function u onto the space of polynomials of order p or lower on the element K_i, which is a polynomial function of order $p(K_i)$ matching the interpolated function u at the endpoints of K_i. Analogously, the fine mesh interpolant $\Pi_{fine,i}u = \Pi_{p+1,i}u$, defined in accord with Section 3.1, is a polynomial of order $p + 1$ in K_i matching u at the element endpoints. Projection-based interpolation error decrease rate is computed similarly to the previous case using the relation

$$\triangle err_i = \|u - \Pi_{coarse,i}u\|_{H_0^1,K_i} - \|u - \Pi_{optimal,i}u\|_{H_0^1,K_i},$$

with $\Pi_{optimal,i} = \Pi_{fine,i}$. This again reduces to

$$\triangle err_i = ||\Pi_{fine,i}u - \Pi_{coarse,i}u||_{H_0^1,K_i}, \tag{6.3}$$

as polynomial spaces corresponding to the coarse and refined element are nested. The situation is depicted in Figure 6.2.

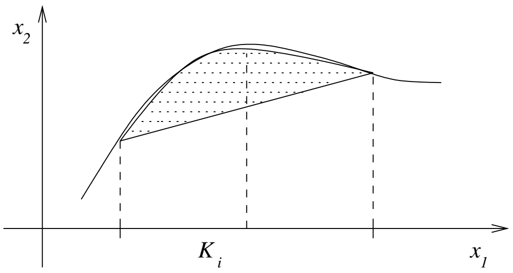

FIGURE 6.2: Decrease of projection-based interpolation error – p-refinement of a linear element.

Step 1, *hp*-adaptivity:

The situation is much more interesting in the case of *hp*-refinements. In harmony with simplifications made for h- and p-refinements we allow that the order p may only be increased by one and the element may only be subdivided into two equal subintervals during one step of the algorithm. Hence, we consider a p-refinement together with a sequence of *competitive h-refinements* (h-refinements that result in the same increase in the number of DOF on the mesh element). Specifically in 1D it means that orders p_L, p_R corresponding to the element sons satisfy the condition

$$p_L + p_R = p + 1,\ 1 \le p_L, p_R,$$

where p is the polynomial order associated with the coarse element. Thus we have p possible ways an element of the polynomial order p can be h-refined plus one option of pure p-refinement. For a linear element the choice is between a quadratic element and two first-order subelements. A quadratic element can either become a cubic element or two equally long subelements with linear+quadratic or quadratic+linear polynomial orders, etc. Projection-based interpolation error decrease rates are computed for each of these $p+1$ *hp*-refinement possibilities separately using the relation

$$\triangle err_i = ||u - \Pi_{coarse,i}u||_{H_0^1,K_i} - ||u - \Pi_{optimal,i}u||_{H_0^1,K_i}, \tag{6.4}$$

which in this case does not simplify as the polynomial spaces corresponding to the coarse and refined element are in general *not nested.* We choose *hp*-refinement, which brings the maximum decrease rate $\triangle err_i$, and by $\Pi_{optimal,i}$ we denote the corresponding projection-based interpolation operator.

REMARK 6.1 Although the application of nonnested spaces in general complicates the theoretical analysis of adaptive finite element procedures, they are crucial for automatic *hp*-adaptivity. Only in this way is the *flow of degrees of freedom* in the computational domain Ω fast enough to allow for exponential convergence of the adaptive scheme – we will have a chance to discuss this feature in more detail and observe it in concrete examples. ▯

Step 2 – selection of elements that will be refined

In this global step we compute the maximum element projection error decrease rate in the mesh $\triangle err_{max}$. Here additional information must enter the decision process to quantify the number of elements to be refined. In accord with an optimality criterion derived in [162], which corresponds to an integer version of the steepest descent method, we select elements whose error decrease rates satisfy

$$\triangle err_i \geq \frac{1}{3} \triangle err_{max}. \tag{6.5}$$

Obviously this is one of many possible ways the optimization process can be driven. It is based on the selection of candidates with most significant projection-based interpolation decrease rate. Although the criterion is optimal within one step of the algorithm, due to the nonlinear nature of the optimization problem it is not necessarily *globally optimal.* Simply put, globally optimal adaptive strategies may involve decisions that are *locally nonoptimal.*

Since the criterion (6.5) completely neglects the information about the actual *magnitude* of the interpolation error, it is practical to add one more criterion

$$err_i \geq 10 err_{average}. \tag{6.6}$$

This criterion selects for refinement elements whose projection-based interpolation error magnitude is significantly larger than the average (we choose concretely one order of magnitude in (6.6); similar to the factor 1/3 in (6.5), this constant is somehow arbitrary). Motivation for (6.6) is illustrated in Figure 6.3.

6.1.2 Automatic mesh optimization algorithms

Let us now summarize the contents of the previous paragraph into a mesh optimization algorithm. We consider a sufficiently smooth function u defined

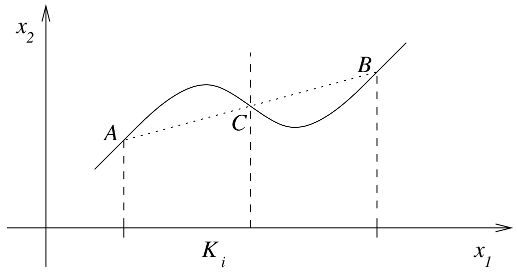

FIGURE 6.3: Motivation for the second criterion (6.6). The solid line represents the function u on a linear mesh element K_i, $p(K_i) = 1$. The dashed line shows both the coarse and fine mesh interpolants which lie very close to each other if the point C happens to lie close to the midpoint $(A + B)/2$. Thus, $\triangle err_i = \|\Pi_{fine,i}u - \Pi_{coarse,i}u\|_{H_0^1,K_i} \approx 0$ although the magnitude of the projection-based interpolation error itself is large, and criterion (6.5) leaves this element untouched.

in the domain Ω and an initial finite element mesh $\mathcal{T}_{h,p}^0$. The goal of the algorithm is to construct a sequence of finite element meshes $\mathcal{T}_{h,p}^k$, $k = 1, 2, \ldots$, that minimizes the projection-based interpolation error of the function u,

$$err^k = \|u - \Pi_{h,p}^k u\|_{H_0^1}. \tag{6.7}$$

ALGORITHM 6.1 (Automatic mesh optimization in 1D)

1. *Put $k := 0$ and consider a function u and an initial finite element mesh $\mathcal{T}_{coarse}^k = \mathcal{T}_{h,p}^k$.*
2. *Compute elementwise projection-based interpolant $\Pi_{coarse}^k u = \Pi_{h,p}^k u$.*
3. *Construct a uniformly refined grid $\mathcal{T}_{fine}^k$:*
   (a) *$\mathcal{T}_{fine}^k = \mathcal{T}_{h/2,p}^k$ (h-adaptivity),*
   (b) *$\mathcal{T}_{fine}^k = \mathcal{T}_{h,p+1}^k$ (p-adaptivity),*
   (c) *$\mathcal{T}_{fine}^k = \mathcal{T}_{h/2,p+1}^k$ (hp-adaptivity).*
4. *Compute elementwise projection-based interpolant $\Pi_{fine}^k u$ corresponding to the mesh $\mathcal{T}_{fine}^k$.*
5. *If the projection-based interpolation error*

$$err^k = \|u - \Pi_{coarse}^k u\|_{H_0^1}$$

satisfies a given tolerance TOL, stop.

6. *Select optimal refinement type for all elements $K_i \in \mathcal{T}^k_{coarse}$ (with our simplifications the decision is nontrivial only in the case of hp-adaptivity), and compute projection-based interpolation error decrease rates $\triangle err^k_i$, using (6.2), (6.3) or (6.4). Compute maximum error decrease rate*

$$\triangle err^k_{max} = \max_{K_i \in \mathcal{T}^k_{coarse}} \triangle err^k_i.$$

7. *Determine which elements will be refined, using the values $\triangle err^k$, err^k, corresponding element contributions and criteria (6.5) and (6.6).*

8. *Perform (optimal) refinement of selected elements, by $\mathcal{T}^{k+1}_{coarse} = \mathcal{T}^{k+1}_{h,p}$ denote the new coarse mesh. Put $k := k + 1$.*

9. *Go to 2.*

Let us now illustrate this mesh optimization algorithm on a few concrete examples.

6.1.3 Automatic h-adaptive mesh optimization

Let us begin with the h-version of the above automatic mesh optimization procedure. To challenge the algorithm, we use a function u of the form

$$u(x) = \frac{\exp(-500(x-0.4)^2)}{2}, \quad x \in [0,1] \tag{6.8}$$

(depicted in Figure 6.4). By the local exponential peak we want to simulate multiscale behavior of the solution that the automatic adaptivity (and automatic mesh optimization in the first place) is supposed to handle. We may choose, for example, a quite inoptimal initial mesh $\mathcal{T}^0_{h,p}$ consisting of $N^{(0)} = 3$ equally long linear elements.

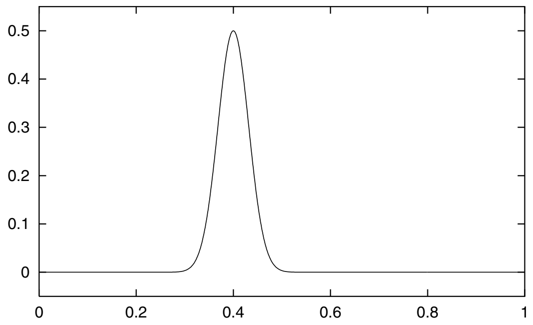

FIGURE 6.4: Example function u forming a local peak at $x = 0.4$.

h-refinements with linear elements

Figures 6.5 – 6.10[1] correspond to h-refinements with linear elements. On the left we depict the interpolant on the coarse grid (solid line) together with the function u (dashed line). On the right we show the corresponding decrease rates for the interpolation error that determine which elements will be refined. The value $err^2 = \|u - \Pi^k_{coarse} u\|^2_{H^1_0}$ means the total projection-based interpolation error on the coarse mesh (squared). The values of the interpolants at the endpoints $x = 0, 1$ are determined by the function u.

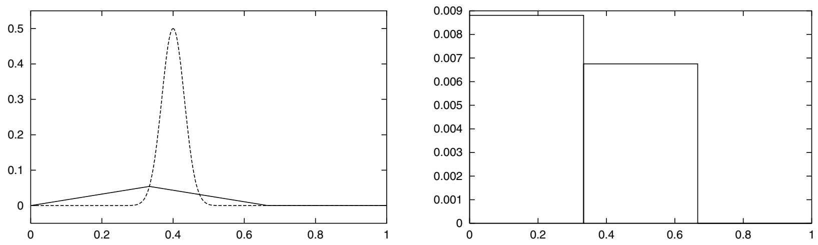

FIGURE 6.5: Step 1: initial mesh interpolant and projection error decrease rates on the initial mesh; number of DOF = 2, err^2 = 6.98862. Elements K_1 and K_2 selected for refinement.

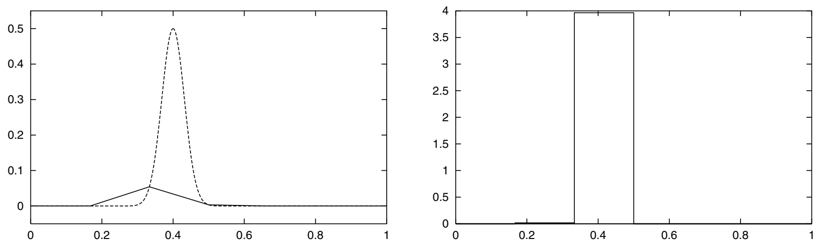

FIGURE 6.6: Step 2: interpolant after one h-refinement and corresponding projection error decrease rates; number of DOF = 4, err^2 = 6.97306. Only element K_3 selected for refinement.

[1]The 1D results presented in this chapter were obtained by a C++ code MESHOPT, which for noncommercial purposes can be downloaded free of charge from the website of the first author, `http://www.iee.cas.cz/staff/solin/` (or `http://www.caam.rice.edu/~solin/`), together with a few additional numerical software packages related to mesh generation, computational fluid dynamics, computational electromagnetics and higher-order finite element methods.

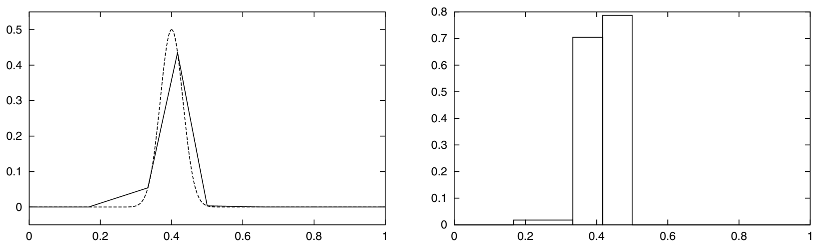

FIGURE 6.7: Step 3: interpolant after two h-refinements and corresponding projection error decrease rates; number of DOF = 5, err^2 = 3.00938. Elements K_3, K_4 selected for refinement.

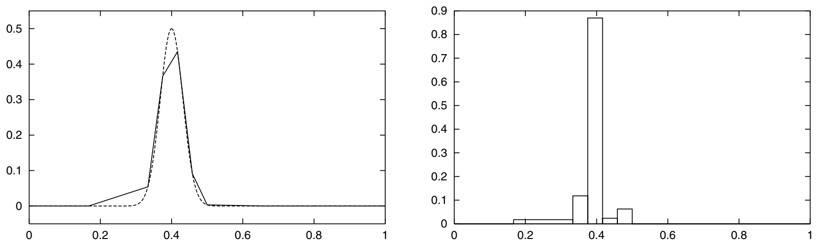

FIGURE 6.8: Step 4: interpolant and error decrease rates after three h-refinements; number of DOF = 7, $err^2 = 1.51809$.

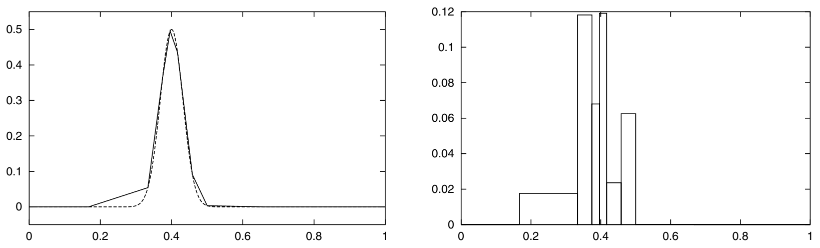

FIGURE 6.9: Step 5: interpolant and error decrease rates after four h-refinements; number of DOF = 8, $err^2 = 0.648154$.

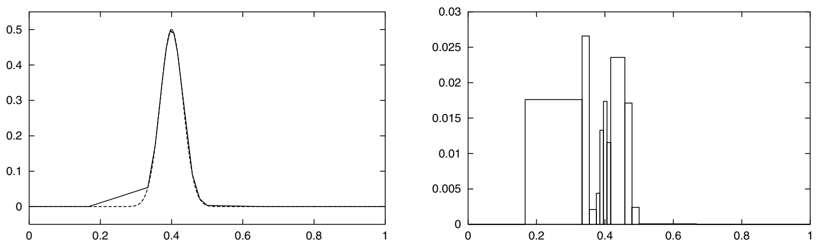
FIGURE 6.10: Step 6: interpolant and error decrease rates after five h-refinements; number of DOF = 12, err^2 = 0.280355. "Long" element K_2 finally selected for refinement. We can observe that despite the inoptimal initial mesh the algorithm *fully automatically* recovers the shape of the function u.

h-refinements with quadratic elements

In the second example we remain with the h-version of the automatic mesh-optimization Algorithm 6.1, but this time we apply quadratic elements. Figures 6.11 – 6.14 depict the same quantities that were shown in the previous paragraph.

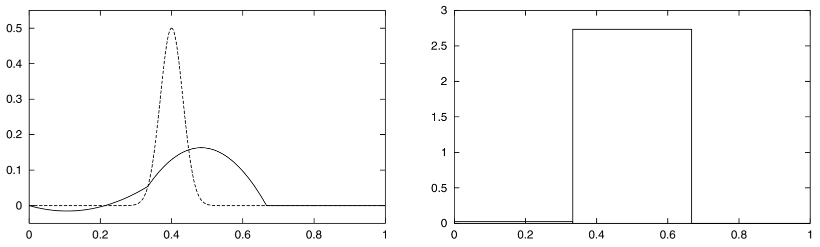
FIGURE 6.11: Step 1: piecewise quadratic interpolant on the initial mesh and corresponding projection error decrease rates; number of DOF = 5, err^2 = 6.67628. Element K_2 selected for refinement (compare with Figure 6.5).

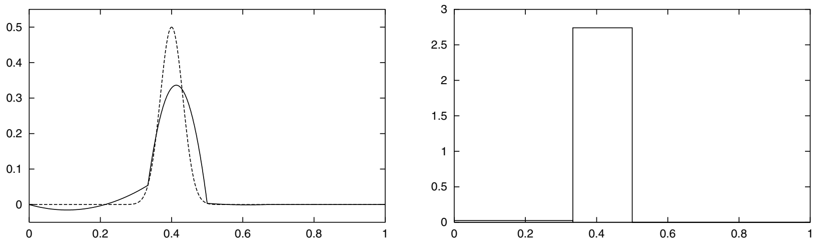
FIGURE 6.12: Step 2: piecewise quadratic interpolant after one h-refinement and corresponding projection error decrease rates; number of DOF $= 7$, $err^2 = 3.94301$.

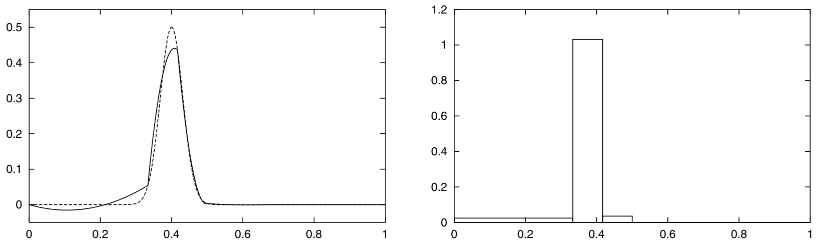
FIGURE 6.13: Step 3: piecewise quadratic interpolant after two h-refinements and corresponding projection error decrease rates; number of DOF $= 9$, $err^2 = 1.20319$.

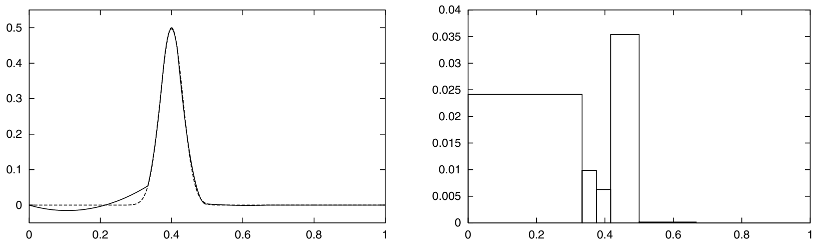
FIGURE 6.14: Step 4: piecewise quadratic interpolant after three h-refinements and corresponding projection error decrease rates; number of DOF $= 11$, $err^2 = 0.171832$.

Convergence of h-adaptive version of the mesh optimization algorithm

Convergence of the projection-based interpolation error with respect to the invested number of degrees of freedom is an essential piece of information about the quality of a mesh optimization algorithm. In Figure 6.15 we show four curves corresponding to the h-version of the automatic mesh optimization Algorithm 6.1 with linear, quadratic, cubic and fourth-order elements, respectively, in decimal logarithmic scale. The horizontal axis represents the number of degrees of freedom.

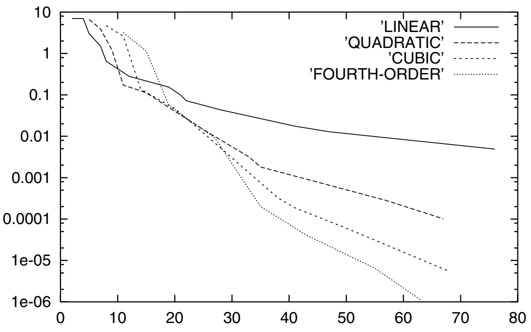

FIGURE 6.15: Automatic h-adaptive mesh optimization. Convergence of projection-based interpolation error err^2 with respect to the number of degrees of freedom for linear, quadratic, cubic and fourth-order elements.

Notice that all the presented convergence curves are monotone – this follows from the fact that the h-version of the mesh optimization Algorithm 6.1 produces a sequence of *nested finite element spaces*.

Further notice that the most significant improvement of the convergence occurs between linear and quadratic elements. This aspect, together with a relative simplicity of the computer implementation of quadratic elements, is the main reason for their great popularity in the framework of h-adaptive schemes.

6.1.4 Automatic p-adaptive mesh optimization

Next in the line is the p-version of the automatic mesh optimization Algorithm 6.1. For the sake of comparison we begin with three equally long linear mesh elements as in Paragraph 6.1.3. A few decisions of the adaptive procedure are shown in Figures 6.16 – 6.18.

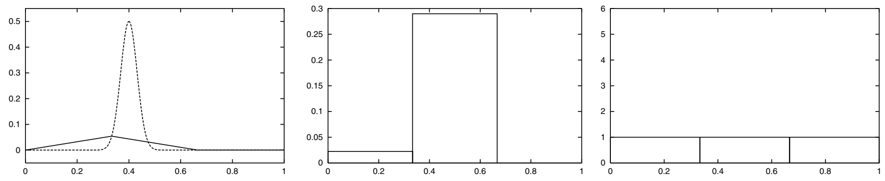

FIGURE 6.16: Step 1: piecewise linear interpolant on the initial mesh, projection error decrease rates and polynomial orders; number of DOF = 2, err^2 = 6.98862. Element K_2 selected for refinement (compare with Figure 6.5).

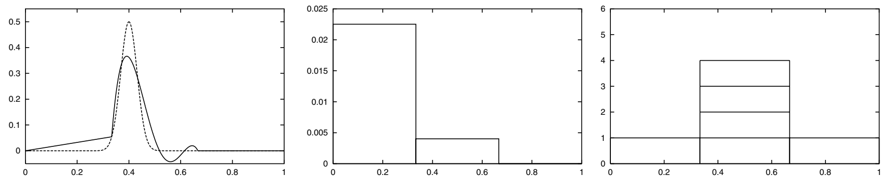

FIGURE 6.17: Step 4: number of DOF = 5, err^2 = 3.26507. This is the first time element K_1 is selected for refinement.

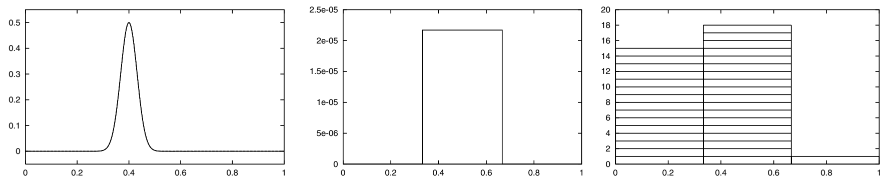

FIGURE 6.18: Step 29: number of DOF = 33, $err^2 = 3.66016 \cdot 10^{-5}$.

Comparing these results with Figures 6.5 – 6.10, we observe that the simple h-adaptive strategy works better than the p-adaptive version illustrated here. This confirms the well-known fact that purely p-adaptive schemes can lead to less satisfactory results than standard h-adaptive methods when starting from inoptimal initial meshes.

6.1.5 Automatic *hp*-adaptive mesh optimization

Finally, we come to an example illustrating performance of the hp-version of the automatic mesh optimization Algorithm 6.1. Figures 6.19 – 6.25 document some of its decisions, showing the corresponding interpolant (solid line) with the interpolated function u (dashed line) in the background, and with the interpolation error decrease rates and the distribution of the polynomial order in the mesh. It is sufficient to start with Step 3, since the first two steps are identical to those of the p-version.

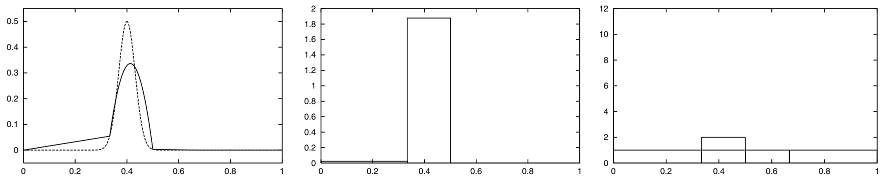

FIGURE 6.19: Step 3: number of DOF = 4, err^2 = 3.96569; quadratic element K_2 (second from the left) selected for *genuine hp-refinement* with $p_L = 2, p_R = 1$. Here we touch on the very origin of exponential convergence of hp-adaptive schemes – the flow of degrees of freedom allowed by pure h- and p-refinements *can never be this fast.*

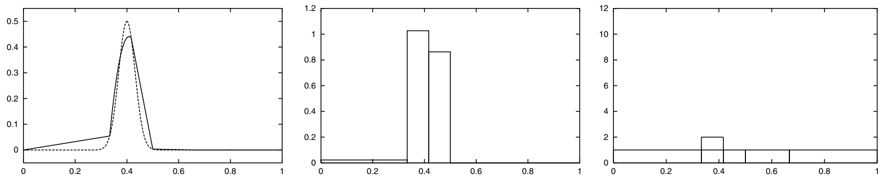

FIGURE 6.20: Step 4: number of DOF = 5, err^2 = 2.08843; elements K_2 and K_3 selected for pure p-refinement.

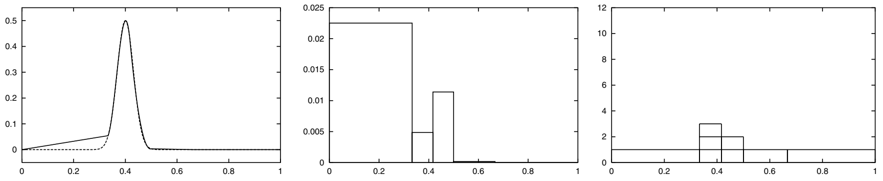

FIGURE 6.21: Step 5: number of DOF = 7, err^2 = 0.199365 (at this number of DOF, this result beats both the h- and p-versions *by one order of magnitude*). Elements K_1 and K_3 selected for pure p-refinement.

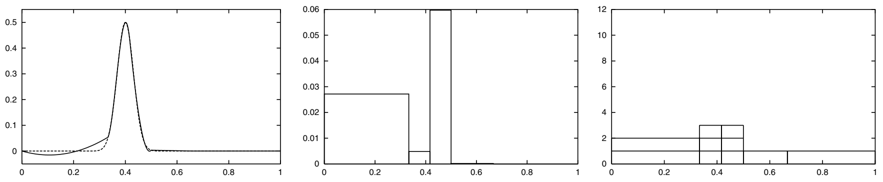

FIGURE 6.22: Step 6: number of DOF = 9, err^2 = 0.165453; again, elements K_1 and K_3 selected for pure p-refinement.

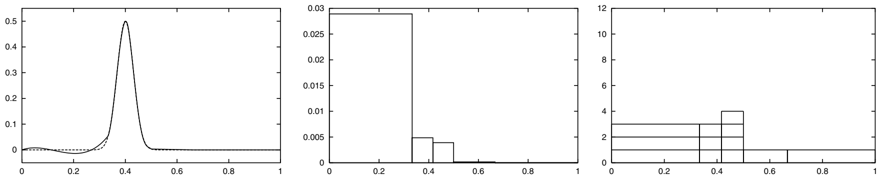

FIGURE 6.23: Step 7: number of DOF = 11, err^2 = 0.0785747; cubic element K_1 selected for genuine hp-refinement with $p_L = 1$, $p_R = 3$.

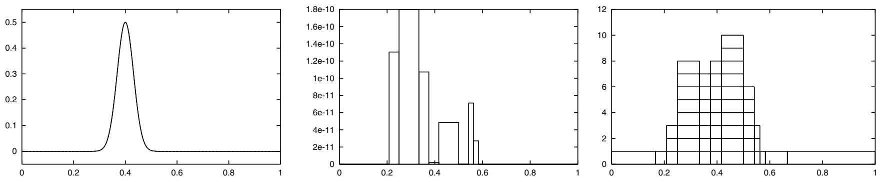

FIGURE 6.24: Step 26: number of DOF = 49, $err^2 = 7.52046 \cdot 10^{-10}$.

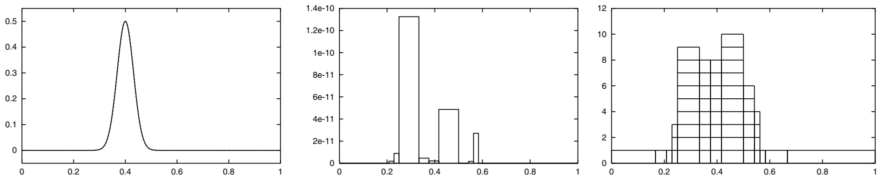

FIGURE 6.25: Step 27: number of DOF = 53, $err^2 = 2.63415 \cdot 10^{-10}$.

Comparison of h-, p- and hp-adaptive mesh optimization procedures

Figure 6.26 compares the performance of h-, p- and hp-adaptive versions of the mesh optimization Algorithm 6.1.

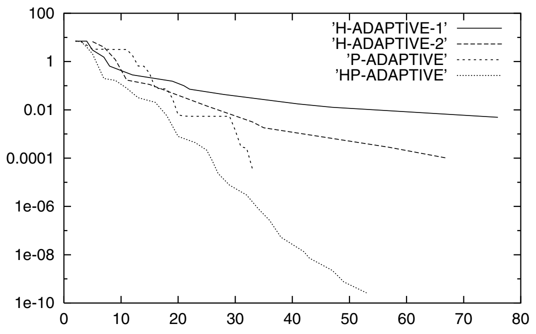

FIGURE 6.26: Convergence of projection-based interpolation error err^2 in decimal logarithmic scale. The horizontal axis represents the number of degrees of freedom. The four curves correspond to the h-adaptive version with linear and quadratic elements, and p- and hp-adaptive versions starting from linear elements.

Notice that the convergence curve related to the p-version ends abruptly after the polynomial order of approximation in our algorithm achieves its maximum, $p_{max} = 23$, in our code. The almost straight line corresponding to the hp-algorithm documents its exponential convergence.

The convergence curves for the h- and p- adaptive mesh optimization procedures have to be monotone because in both cases the resulting sequence of finite element spaces is nested. Since for the p-adaptive scheme this is not quite obvious, we show the appropriate values of err^2 in Table 6.1.

TABLE 6.1: Projection-based interpolation error err^2 for the p-adaptive scheme.

# DOF	err^2	# DOF	err^2	# DOF	err^2
2	6.98862	13	0.656073	24	0.00541498
3	6.69879	14	0.63802	25	0.00540822
4	4.74929	15	0.325973	26	0.00540527
5	3.26507	16	0.0793164	27	0.00540465
6	3.24255	18	0.0735678	29	0.00540449
7	3.21539	19	0.0301057	30	0.00142186
8	3.19222	20	0.00624085	31	0.000300026
9	3.17716	21	0.00551845	32	0.000258359
11	3.1656	22	0.00542142	33	0.000036602
12	1.80304	23	0.00542056		

6.2 Adaptive strategies based on automatic mesh optimization

In Section 6.1 we presented a strategy capable of automatic progressive adjustment of the finite element mesh to an a priori given function u by adaptive minimization of its projection-based interpolation error (6.1),

$$err_k = \|u - \Pi^k_{h,p} u\|_{H^1_0}. \tag{6.9}$$

The next natural step is to generalize this technique to a strategy for automatic adaptive adjustment of the finite element mesh to *an unknown function* u representing the solution of an investigated variational problem.

Hence, our model problem is to find a solution u lying in a Hilbert space V and satisfying the equation

$$b(u, v) = f(v) \quad \text{for all } v \in V. \tag{6.10}$$

Here b is an elliptic bilinear form defined on $V \times V$, and $f \in V'$. If the form b is positive definite, we can define the standard *energy norm*,

$$\|u\|^2_e = b(u, u) \quad \text{for all } u \in V. \tag{6.11}$$

If the form b is *not positive definite* the situation becomes more delicate, and we may try to split it into a positive definite part and a compact perturbation that can be neglected – see, e.g., [185] for more details. Here we will assume that b is positive definite.

Hence, from now on, the symbol u stands for the *unknown exact solution* of the underlying variational problem. Instead of minimizing projection-based interpolation error (6.9), we are interested in the minimization of *the discretization error*

$$e_{h,p}^k = \|u - u_{h,p}^k\|_e, \tag{6.12}$$

where $u_{h,p}^k$ stands for approximate solutions obtained on the sequence of optimal finite element meshes $\mathcal{T}_{h,p}^k$, $k = 1, 2, \ldots$. Although obviously we cannot provide the mesh optimization procedure with the exact solution u, at least we can supply a good estimate. There are several ways this can be done based on the evaluation of *reference solutions* (see, e.g., [64]).

6.2.1 Reference solutions

By *reference solution* u_{ref} we mean an approximate solution that lies significantly closer to the exact solution u than the approximation $u_{h,p}$ on the current (let us call it *coarse*) finite element mesh. The reference solution u_{ref} should satisfy the following requirements:

- it should be computable using only the coarse mesh, coarse mesh solution $u_{h,p}$, and the data to the problem,
- its evaluation should be significantly faster than the solution of the original finite element problem,
- the difference $u_{ref} - u_{h,p}$ should provide good approximation of the discretization error $e_{h,p}$.

Usually we are interested in reference solutions that are at least by one order of accuracy better than the coarse mesh approximation (here we mean in h for h-adaptive procedures, in p for p-adaptive ones, and both in h and p for hp-adaptive algorithms).

Solution on globally uniformly refined grids

The situation is simpler for pure h-adaptivity where, for example, highly accurate approximations based on Babuška's *extraction formulae* (see, e.g., [162]) can be used. More difficult are p- and hp-adaptive methods since the extraction techniques fail for higher ps. A robust way to obtain a reference solution originally proposed by Demkowicz [64] is to use globally uniformly refined grids. For h-adaptivity this means $h \to h/2$ refinement where all edges are divided into half (and triangles and quadrilaterals are subdivided into four subelements, hexahedra into eight, etc.). In the case of p-adaptive schemes one increases the order of approximation in all elements by one with no spatial refinement. For hp-adaptive methods one performs an $(h,p) \to (h/2, p+1)$ refinement. The reference solution $u_{ref} = u_{h/2}$, $u_{ref} = u_{p+1}$ or $u_{ref} = u_{h/2,p+1}$, respectively, is then the approximate finite element solution on the fine mesh.

The fine mesh problems are usually several times larger than the original ones. Their size rises most significantly on hp-meshes that already contain a number of higher order elements. However, the point is that we do not need to start from scratch to resolve the fine mesh problem. In fact, the coarse mesh solution already represents a lot of valuable information on lower frequencies that we can (and *have to*) use. This is a textbook situation for the application of a multigrid solver. Specifically in our case, as we deal only with one coarse and one fine grid, we need a *two-grid solver* (see Paragraph 5.2.6). In this way the time needed for obtaining the fine mesh solution becomes only a fraction of the time that would be needed to compute it from scratch.

Moreover, since we use the fine grid solution as an error indicator for the adaptive algorithm only, we do not need to resolve it extremely accurately. Recent experiments show that most of the time just a few smoothing iterations on the fine mesh can drive the adaptive scheme reliably.

6.2.2 A strategy based on automatic mesh optimization

Enhancing the automatic mesh optimization Algorithm 6.1 by the error indicator obtained as the difference between the reference and coarse mesh solution, we arrive at the following automatic adaptive algorithm:

ALGORITHM 6.2 (Automatic adaptivity in 1D)

1. *Put* $k := 0$ *and consider an initial finite element mesh* $\mathcal{T}^k_{coarse} = \mathcal{T}^k_{h,p}$.
2. *Compute approximation* $u^k_{coarse} = u_{h,p}$ *on the current mesh* $\mathcal{T}^k_{coarse}$.
3. *Construct a uniformly refined grid* $\mathcal{T}^k_{fine}$:
   (a) $\mathcal{T}^k_{fine} = \mathcal{T}^k_{h/2,p}$ *(h-adaptivity),*
   (b) $\mathcal{T}^k_{fine} = \mathcal{T}^k_{h,p+1}$ *(p-adaptivity),*
   (c) $\mathcal{T}^k_{fine} = \mathcal{T}^k_{h/2,p+1}$ *(hp-adaptivity).*
4. *Compute approximation* u^k_{fine} *on the fine mesh* $\mathcal{T}^k_{fine}$, *optimally using a two-grid solver starting from the coarse mesh solution* u^k_{coarse}.
5. *If the approximate discretization error* $err^k = \|u^k_{fine} - u^k_{coarse}\|_e$ *lies within a given tolerance* TOL, *stop.*
6. *Replace the original (global) finite element problem of minimizing the discretization error,*
$$\|u - u^k_{coarse}\|_e, \tag{6.13}$$
by (elementwise local) problems of minimizing projection-based interpolation error,
$$\|u - \Pi^k_{coarse}u\|_{e,K_i}, \quad 1 \le i \le N^{(k)}, \tag{6.14}$$

by neglecting the difference between the coarse mesh solution u^k_{coarse} *and coarse mesh interpolant of the exact solution* $\Pi^k_{coarse}u$,

$$\|u^k_{coarse} - \Pi^k_{coarse}u\|_{e,K_i} \approx 0, \quad 1 \le i \le N^{(k)}. \tag{6.15}$$

Notice that here the locality property of the projection-based interpolation operator turns out to be essential.

7. *Replace the exact solution* u *by the reference solution* u_{ref}, $u := u_{ref} = u^k_{fine}$. *Minimize elementwise contributions to the projection-based interpolation error*

   $$\|u^k_{fine} - \Pi^k_{optimal}u^k_{fine}\|_{e,K_i}, \quad 1 \le i \le N^{(k)},$$

   as explained in Step 1 of Paragraph 6.1.1:

   - *Select optimal refinement type for all elements* K_i, $1 \le i \le N^{(k)}$.
   - *Compute corresponding projection-based interpolation error decrease rates,*

     $$\triangle err_i = \|u^k_{fine} - \Pi^k_{coarse,i}u^k_{fine}\|_{e,K_i} - \|u^k_{fine} - \Pi^k_{optimal,i}u^k_{fine}\|_{e,K_i}, \tag{6.16}$$

     for all elements K_i, $1 \le i \le N^{(k)}$. *Analogously as in Section 6.1 it is* $\Pi_{optimal,i} = \Pi_{fine,i}$ *for pure h- and p-adaptivity, and the previous relation therefore reduces to*

     $$\triangle err_i = \|\Pi^k_{fine,i}u^k_{fine} - \Pi^k_{coarse,i}u^k_{fine}\|_{e,K_i}$$

     in these cases.

8. *Determine which elements will be refined, using criteria (6.5), (6.6).*

9. *Refine selected elements, by* $\mathcal{T}^{k+1}_{coarse} = \mathcal{T}^{k+1}_{h,p}$ *denote the new optimal (coarse) mesh. Put* $k := k + 1$.

10. *Go to 2.*

6.2.3 Model problem

Let us have a look at how the automatic adaptive strategies introduced above perform in practice. In order to preserve the possibility of comparison with results of the automatic mesh optimization procedures from Paragraphs 6.1.3 – 6.1.5, we will choose a model equation in such a way that the function u from (6.8),

$$u(x) = \frac{\exp(-500(x - 0.4)^2)}{2},$$

which was depicted in Figure 6.4, is its exact solution. For simplicity let us consider a positive definite elliptic problem

$$-\triangle u + k^2 u = f \quad \text{in } \Omega = (0,1) \tag{6.17}$$

with, for instance, $k = \sqrt{5}$. At Ω endpoints we prescribe Dirichlet conditions that coincide with the exact solution u.

6.2.4 Automatic h-adaptivity

Figures 6.27 – 6.31 show a few first steps of the h-version of the adaptive Algorithm 6.2, starting from three equally long linear elements as in the previous cases (compare with Figures 6.5 – 6.10). The value err^2 means now the total projection-based interpolation error of the fine mesh solution u^k_{fine} with respect to the coarse mesh interpolant (squared),

$$err^2 = \|u^k_{fine} - \Pi^k_{coarse} u^k_{fine}\|^2_e.$$

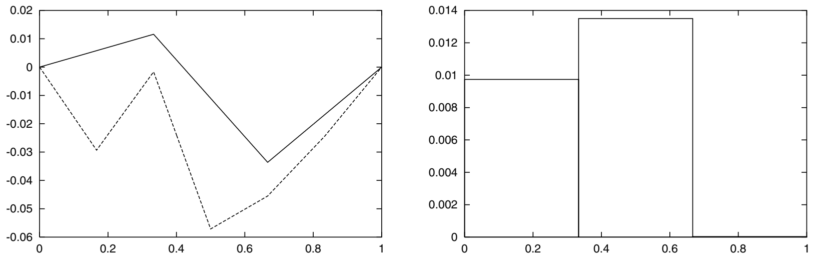

FIGURE 6.27: Step 1: initial coarse and fine mesh solutions, projection error decrease rates; number of DOF = 2, err^2 = 0.0232704. Elements K_1 and K_2 selected for refinement.

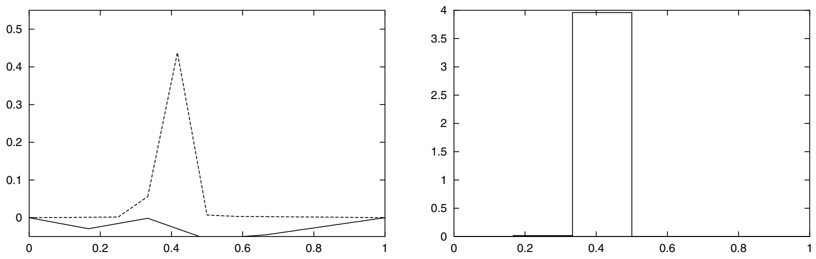

FIGURE 6.28: Step 2: after one h-refinement; number of DOF = 4, err^2 = 3.97743. Element K_3 selected for refinement.

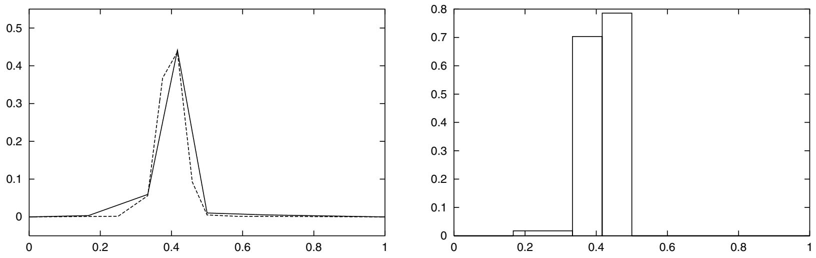

FIGURE 6.29: Step 3: number of DOF = 5, err^2 = 1.50713. Elements K_3, K_4 selected for refinement.

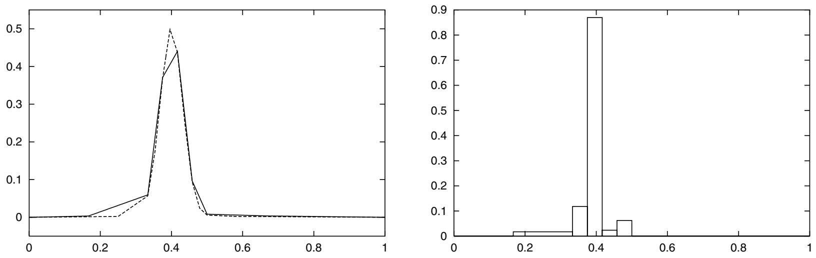

FIGURE 6.30: Step 4: number of DOF = 7, err^2 = 1.09153.

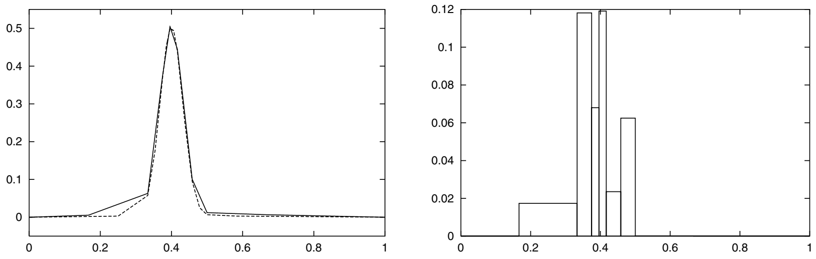

FIGURE 6.31: Step 5: number of DOF = 8, err^2 = 0.408867. Observe in the last four figures the growth of the error decrease rate of element K_2 – it will cause its refinement in the next step.

6.2.5 Automatic p-adaptivity

Next let us observe the performance of the p-version of the automatic adaptive Algorithm 6.2 applied to the problem (6.17). Figures 6.32 – 6.35 again start from three equally long linear elements (compare with Figures 6.16 – 6.18).

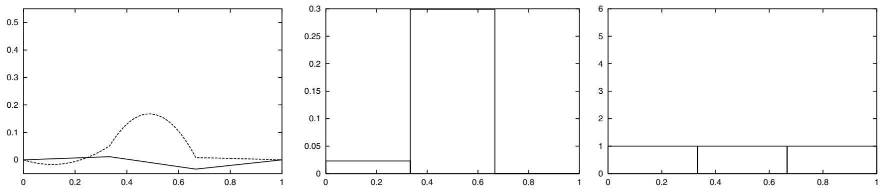

FIGURE 6.32: Step 1: coarse and fine mesh solutions corresponding to the initial mesh, projection error decrease rates and distribution of the polynomial order in the coarse mesh; number of DOF = 2, $err^2 = 0.322116$. Element K_2 selected for refinement.

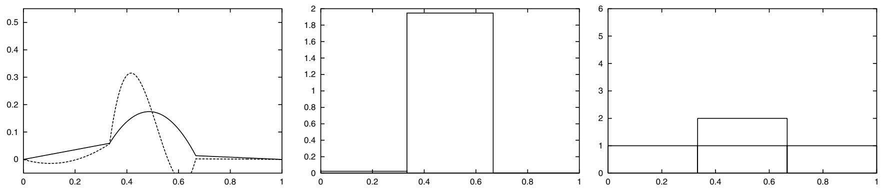

FIGURE 6.33: Step 2: number of DOF = 3, $err^2 = 1.97128$. Element K_2 selected for refinement.

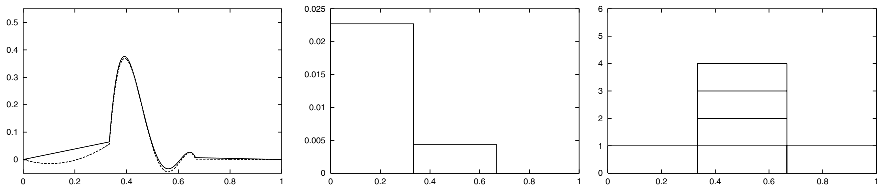

FIGURE 6.34: Step 4: number of DOF = 5, $err^2 = 0.028536$. Element K_1 selected for refinement.

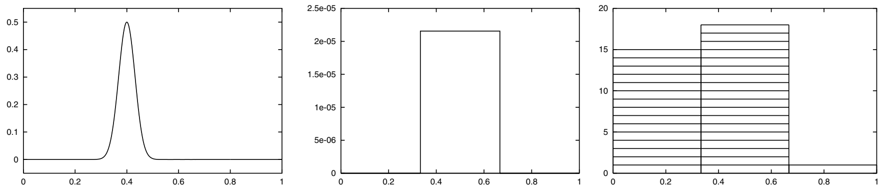

FIGURE 6.35: Step 29: number of DOF = 33, $err^2 = 2.16267 \cdot 10^{-5}$. Element K_2 selected for refinement.

6.2.6 Automatic *hp*-adaptivity

At last we come to the *hp*-version of the automatic adaptive Algorithm 6.2. The algorithm starts from the same initial mesh as in the previous cases. Compare Figures 6.36 – 6.43 with Figures 6.19 – 6.25.

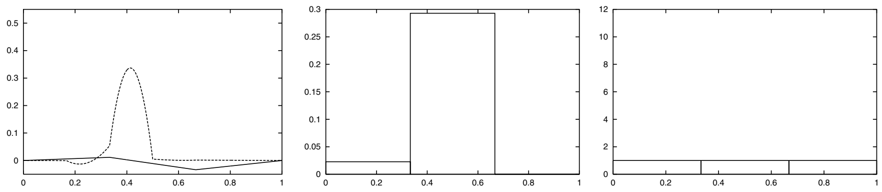

FIGURE 6.36: Step 1: number of DOF = 2, err^2 = 3.06323. Element K_2 selected for pure p-refinement.

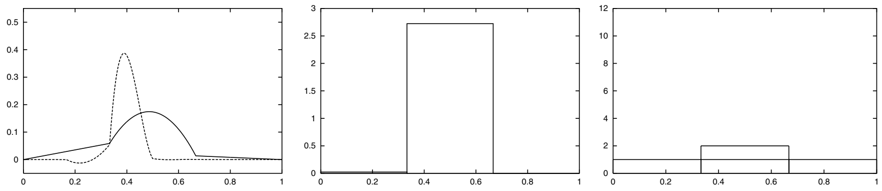

FIGURE 6.37: Step 2: number of DOF = 3, err^2 = 4.05918. Element K_2 selected for genuine hp-refinement with $p_L = 2, p_R = 1$.

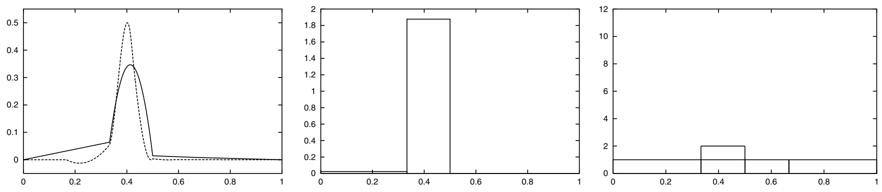

FIGURE 6.38: Step 3: number of DOF = 4, err^2 = 3.82452. Element K_2 selected for genuine hp-refinement with $p_L = 2, p_R = 1$.

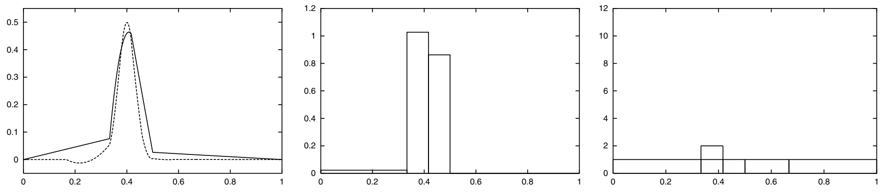

FIGURE 6.39: Step 4: number of DOF = 5, err^2 = 1.9912. Elements K_2, K_3 selected for pure p-refinement.

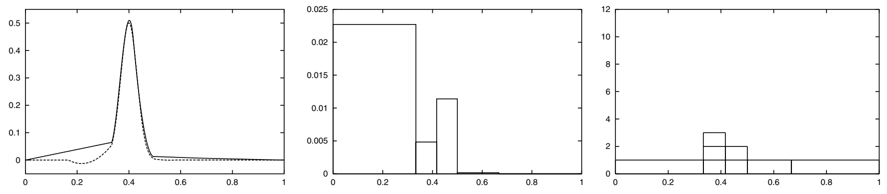

FIGURE 6.40: Step 5: number of DOF = 7, $err^2 = 0.145694$. Elements K_1, K_3 selected for pure p-refinement.

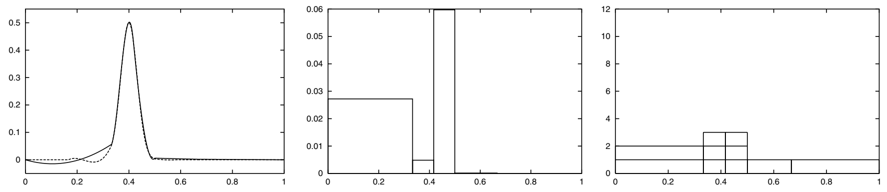

FIGURE 6.41: Step 6: number of DOF = 9, $err^2 = 0.144764$. Elements K_1, K_3 selected for pure p-refinement.

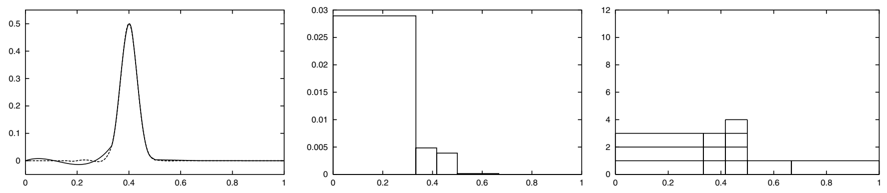

FIGURE 6.42: Step 7: number of DOF = 11, $err^2 = 0.0733345$. Element K_1 selected for hp-refinement.

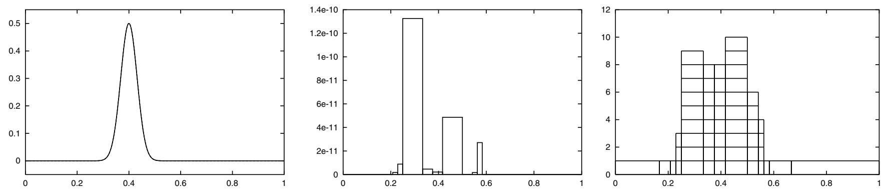

FIGURE 6.43: Step 27: number of DOF = 53, $err^2 = 2.62835 \cdot 10^{-10}$.

Comparison of performance of h-, p- and hp-adaptive strategies

Convergence curves of the approximate discretization error $err^2 = \|u^k_{fine} - u^k_{coarse}\|^2_e$ for the h-, p- and hp-versions of the automatic adaptive Algorithm 6.2 are shown in Figure 6.44. The h-version is considered with both linear and quadratic elements.

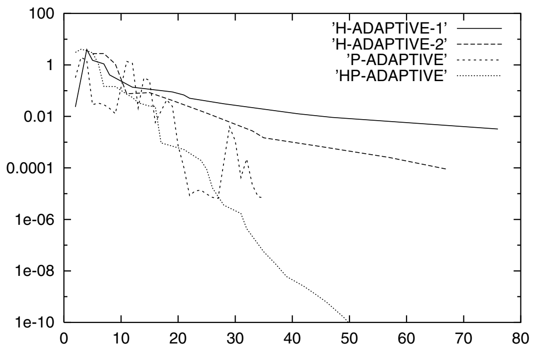

FIGURE 6.44: Convergence of the approximate discretization error err^2 with respect to the number of degrees of freedom in decimal logarithmic scale.

Notice that the convergence curves are not monotone – this is not in contradiction to the theory because we depict *approximate* discretization error err instead of the true error $e_{h,p} = \|u - u_{h,p}\|_e$. Convergence curves of $e_{h,p}$ would have to be monotone for the h- and p-adaptive schemes, since finite element subspaces in the resulting sequence are nested. Due to the nonnestedness of genuine hp-refinements this is not the case for hp-adaptivity.

It is interesting to compare the convergence curves of the approximate discretization error err^2 shown in Figure 6.44 with the convergence curves from Figure 6.26, related to mesh optimization schemes that "knew" the function u in advance.

6.3 Goal-oriented adaptivity

During the last two decades, goal-oriented adaptivity for partial differential equations has been subject to ongoing scientific and engineering effort, and several basic methodologies have been proposed (see, e.g., [29, 30, 164, 48, 152, 153, 23, 148]). In comparison with adaptivity in energy norm, which is designed to minimize the energy of the residual of the approximate solution, the goal-oriented approach attempts to control concrete features of the solved problem – *quantities of interest*. Goal-oriented adaptive techniques achieve precise resolution in quantities of interest with qualitatively fewer degrees of freedom than energy-driven adaptive schemes.

6.3.1 Quantities of interest

Quantities of interest are specific properties of the solution, in the precise resolution of which we are interested. Often they can be represented by bounded linear functionals of the (generally vector-valued) solution u. In finite element computations it is convenient to define the quantities of interest in the form of an integral over the domain Ω, since integration lies at the heart of all finite element codes. Boundedness of the interest functional is obviously crucial for the success of the goal-oriented adaptive procedure.

We may be interested, e.g., in the precise resolution of the average of the solution u in a selected subdomain $\Omega_s \subset \Omega$. Then, the linear functional of interest can be written in the form

$$L(u) = \frac{1}{|\Omega_s|} \int_\Omega \chi_{\Omega_s}(x) u(x) \mathrm{d}x, \tag{6.18}$$

where χ_{Ω_s} stands for the characteristic function of the domain Ω_s, i.e.,

$$\chi_{\Omega_s}(x) = \begin{cases} 1 \text{ if } x \in \Omega_s, \\ \\ 0 \text{ otherwise.} \end{cases}$$

PROPOSITION 6.1
Let $u \in H_0^1(\Omega)$. Then, the linear functional defined in (6.18) is bounded in $H_0^1(\Omega)$.

PROOF Since $u \in H_0^1(\Omega)$ and $\chi_{\Omega_s} \in L^2(\Omega)$, using the Minkowski inequality we obtain

$$|L(u)| = \left| \frac{1}{|\Omega_s|} \int_\Omega \chi_{\Omega_s}(x) u(x) \mathrm{d}x \right| \le \|\chi_{\Omega_s}\|_0 \|u\|_0 \le C_0 |\Omega_s| |u|_1,$$

where C_0 is the Poincaré constant and $|u|_1$ denotes the H^1-seminorm. $\square$

For vector-valued solutions, the goal of the computation may be the flux through the boundary of a subdomain Ω_s of Ω, in which case the linear functional of interest reads

$$L(u) = \int_{\partial\Omega_s} u(x) \cdot n(x) \mathrm{d}S = \int_{\Omega_s} \nabla \cdot u(x) \mathrm{d}x = \int_{\Omega} \chi_{\Omega_s}(x) \nabla \cdot u(x) \mathrm{d}x. \quad (6.19)$$

It is easy to see that this linear functional is bounded in $(H_0^1)^d$.

In the case of vector-valued problems, our goal may be the precise resolution of a selected solution component u_i in a subdomain Ω_s. Then, the linear functional of interest will have the form

$$L(u) = \frac{1}{|\Omega_s|} \int_{\Omega} \chi_{\Omega_s}(x) u_i(x) \mathrm{d}x. \quad (6.20)$$

6.3.2 Formulation of the dual problem

Let us recall the basic ideas of the goal-oriented adaptivity, leading to the formulation of the dual problem.

1. Consider a problem to find a solution u lying in a Hilbert space V and satisfying the weak formulation (6.10),

$$b(u, v) = f(v) \quad \text{for all } v \in V, \quad (6.21)$$

   b being an elliptic bilinear form defined on $V \times V$ and $f \in V'$.

2. Consider the discrete problem

$$b(u_{h,p}, v_{h,p}) = f(v_{h,p}), \quad \text{for all } v_{h,p} \in V_{h,p}, \quad (6.22)$$

   where $V_{h,p} \subset V$ is a piecewise-polynomial subspace of the space V.

3. Define the discretization error $e_{h,p} = u - u_{h,p}$ and consider the residual $r_{h,p} \in V'$,

$$r_{h,p}(v) = f(v) - b(u_{h,p}, v), \quad v \in V. \quad (6.23)$$

4. It is our aim to relate the residual $r_{h,p}$ to the error in the quantity of interest, i.e., to find $G \in V''$ such that

$$G(r_{h,p}) = L(e_{h,p}).$$

5. By reflexivity, G can be identified with an element v in the original space V such that

$$G(r_{h,p}) = r_{h,p}(v).$$

Sometimes the function v is called the *influence function.*

6. Using now (6.21), (6.22) and (6.23), we obtain

$$\begin{aligned} L(e_{h,p}) &= G(r_{h,p}) = r_{h,p}(v) = f(v) - b(u_{h,p}, v) = b(u, v) - b(u_{h,p}, v) \\ &= b(e_{h,p}, v). \end{aligned}$$

Hence, the influence function v is obtained as a solution of the following *dual problem.*

7. **Dual problem**: Find $v \in V$ such that

$$b(u, v) = L(u), \tag{6.24}$$

for all $u \in V$.

Notice that the primal and dual problems differ *only in the right-hand side.* The same stiffness matrix can be used for both the primal and dual problems.

6.3.3 Error control in quantity of interest

In goal-oriented adaptivity, it is our aim to control the error in the quantity of interest,

$$e_{h,p} = |L(u) - L(u_{h,p})|. \tag{6.25}$$

Optimally we would like to control it by means of the discretization errors *in energy norm* for both the primal and dual problem, as these error quantities are quite easily accessible. Hence, consider the discrete dual problem

$$b(u_{h,p}, v_{h,p}) = L(u_{h,p}) \quad \text{for all } u_{h,p} \in V_{h,p}. \tag{6.26}$$

It follows from the linearity of the functional L and (bi)linearity of the form b that

$$|L(u) - L(u_{h,p})| = |L(u - u_{h,p})| = |b(u - u_{h,p}, v)|.$$

Standard orthogonality property for the error of the solution (subtract (6.22) from (6.21)) yields

$$|b(u - u_{h,p}, v)| = |b(u - u_{h,p}, v - v_{h,p})|.$$

In the language of element contributions we can write

$$|b(u - u_{h,p}, v - v_{h,p})| \leq C \sum_{K \in \tau_{h,p}} ||u - u_{h,p}||_{e,K} ||v - v_{h,p}||_{e,K},$$

and thus finally we obtain

$$|L(u) - L(u_{h,p})| \leq C \sum_{K \in \tau_{h,p}} ||u - u_{h,p}||_{e,K} ||v - v_{h,p}||_{e,K}. \tag{6.27}$$

If $b(.,.)$ is symmetric, $C = 1$. In practice, if the finite element code is sufficiently modular, it should not be very difficult to incorporate the solution of the dual problem into the finite element algorithm since we can use the same stiffness matrix. We only have to adjust the procedure for the discretization of the right-hand side functional L. Dramatic changes in the code are usually not necessary for the implementation of the calculation of the error in energy norm for the dual problem.

When using the Gauss elimination, it is advantageous to eliminate for both the primal and dual right-hand sides simultaneously. When using iterative solvers, the difference between the consequent and simultaneous solution of the discrete primal and dual problem is less significant, depending on the implementation of the iterative solver.

6.3.4 Selected nonlinear and unbounded functionals

Sometimes we come across quantities of interest that either are nonlinear or that cannot be directly expressed in terms of bounded linear functionals of the solution u. In such cases one usually attempts to linearize the nonlinear ones by some standard technique, and to regularize the unbounded functionals.

Nonlinear quantities of interest

To give a simple example, let us choose as our goal the integral over the domain Ω of squared magnitude of the (generally vector-valued) solution u,

$$N(u) = \int_{\Omega_s} |u|^2 \, \mathrm{d}x.$$

Error in the nonlinear quantity $N(u)$ can be written as

$$N(u) - N(u_{h,p}) = \int_{\Omega_s} |u|^2 - |u_{h,p}|^2 \, \mathrm{d}x = 2 \int_{\Omega_s} u_{h,p} \cdot e_{h,p} \, \mathrm{d}x + \int_{\Omega_s} |e_{h,p}|^2 \, \mathrm{d}x,$$

where $e_{h,p} = u - u_{h,p}$ is the discretization error. $N(u)$ can be linearized by neglecting the higher-order term in $e_{h,p}$. This yields a bounded linear functional

$$L(e) = 2\int_{\Omega_s} u_{h,p} \cdot e \,\mathrm{d}x = \int_{\Omega} \chi_{\Omega_s}(x) u_{h,p}(x) \cdot e(x) \,\mathrm{d}x.$$

REMARK 6.2 (Starting the goal-oriented adaptivity) When using a linearization technique based on neglecting of higher-order error terms, one should not forget this fact in the computation, and start the goal-oriented adaptive procedure after first making sure that one is sufficiently close to the solution u in some global (e.g., *energy*) norm. This may be done by a switch in the code that allows us to start the goal-oriented procedure only when the approximate discretization error $e_{h,p} \approx \|u_{ref} - u_{h,p}\|_e$ is smaller than a given tolerance. The reference solution can be computed, for example, as described in Paragraph 6.2.1. $\square$

Sometimes we may deal with nonlinear quantities of interest containing the natural or decimal logarithm (this can be the case, for example, when working with the decibel scale – see also, e.g., [185]). The nonlinear functional

$$N(u) = \log_{10}\left(\left|\int_{\Omega_s} u \,\mathrm{d}x\right|\right),$$

defined if $\int_{\Omega_s} u \,\mathrm{d}x \neq 0$, yields an error quantity of the form

$$N(u) - N(u_{h,p}) = \log_{10}\left(\left|\frac{\int_{\Omega_s} u \,\mathrm{d}x}{\int_{\Omega_s} u_{h,p} \,\mathrm{d}x}\right|\right). \tag{6.28}$$

When the discretization error $e_{h,p} = u - u_{h,p}$ is sufficiently small, it is

$$\frac{\int_{\Omega_s} u \,\mathrm{d}x}{\int_{\Omega_s} u_{h,p} \,\mathrm{d}x} \approx 1,$$

and we can approximate

$$N(u) - N(u_{h,p}) = \log_{10}\left(1 + \frac{\int_{\Omega_s} e_{h,p} \,\mathrm{d}x}{\int_{\Omega_s} u_{h,p} \,\mathrm{d}x}\right) \approx \frac{\int_{\Omega_s} e_{h,p} \,\mathrm{d}x}{\ln 10 \int_{\Omega_s} u_{h,p} \,\mathrm{d}x}.$$

Hence the linearization of the logarithm $\log_{10} y$ at $y = 1$ leads to a bounded linear functional

$$L(e) = \frac{\int_{\Omega_s} e \,\mathrm{d}x}{\int_{\Omega_s} u_{h,p} \,\mathrm{d}x}.$$

In this case the approximation $u_{h,p}$ works as a scaling parameter in the dual problem. Before we start the goal-oriented adaptivity, it is necessary to verify that $\int_{\Omega_s} u_{h,p} \,\mathrm{d}x$ is nonzero. Moreover, according to Remark 6.2, we should verify that $e_{h,p} \approx \|u_{ref} - u_{h,p}\|_e$ is sufficiently small.

Pointwise values of the solution u

A particularly important example of a computational goal is the value $u(x_0)$ of the solution u at a selected point $x_0 \in \Omega \subset \mathbb{R}^d$. In one spatial dimension, the corresponding operator

$$L(u) = u(x_0) \tag{6.29}$$

is linear and bounded (solution of the dual problem is the corresponding Green's function). Unfortunately, this is no longer true in higher spatial dimensions.

To overcome this obstacle we may, for example, replace the pointwise value of the solution $u(x_0)$ with an average over a small ball $B(x_0, r) \subset \Omega$, $r > 0$, surrounding the point x_0,

$$L(u) = \frac{1}{|B(x_0, r)|} \int_{B(x_0,r)} u(x)\,\mathrm{d}x. \tag{6.30}$$

This functional of interest is bounded.

Another standard approach to the treatment of pointwise values of interest is to apply regularizing *mollifiers* (see, e.g., [148]). For $\epsilon > 0$ we define a linear bounded operator of the form

$$L(u) = \int_\Omega k_\epsilon(x - x_0) u(x)\,\mathrm{d}x,$$

where

$$k_\epsilon(x) = \begin{cases} C_{\epsilon,x_0} \exp\left(\dfrac{\epsilon^2}{|x|^2 - \epsilon^2}\right) & \text{if } |x| < \epsilon, \\ 0 \quad \text{otherwise.} & \end{cases}$$

The constant C_{ϵ,x_0} is chosen to satisfy

$$\int_\Omega k_\epsilon(x - x_0)\,\mathrm{d}x = 1.$$

Notice that k_ϵ is infinitely smooth in Ω. In general, the mollifier can be less smooth to obtain a bounded functional in $H^1(\Omega)$. In practical implementation it is convenient to choose the value of ϵ in such a way that the support of the mollifier k_ϵ lies in one element only (supposed that x_0 does not lie on an element interface or at a grid vertex).

6.4 Automatic goal-oriented h-, p- and hp-adaptivity

Let us see how the goal-oriented adaptivity can be incorporated into the automatic h-, p- and hp-adaptive schemes from Section 6.2. We shall begin with one-dimensional algorithms again, since the basic principles are here free of the unavoidable technical difficulty of higher-dimensional implementations. The reader should indeed be aware that from the practical point of view, goal-oriented adaptivity makes good sense mainly in higher spatial dimensions – we will address two-dimensional schemes in more detail in Section 6.5.

6.4.1 Automatic goal-oriented adaptive strategies

The basic difference between the automatic *h*-, *p*- and *hp*-adaptive strategies from Section 6.2 and the goal-oriented approach is that instead of minimizing the error (6.12) in a global (energy) norm we minimize the error in the quantity of interest,

$$|L(u) - L(u_{h,p})|. \tag{6.31}$$

Returning to the estimate (6.27),

$$|L(u) - L(u_{h,p})| \le C \sum_{K \in \tau_{h,p}} ||u - u_{h,p}||_{e,K} ||v - v_{h,p}||_{e,K},$$

we see we will have to simultaneously minimize the discretization error in the solution to both the primal and dual problem.

ALGORITHM 6.3 (Automatic goal-oriented adaptivity in 1D)

1. *Put $k := 0$ and consider an initial finite element mesh $\mathcal{T}^k_{coarse} = \mathcal{T}^k_{h,p}$.*
2. *Compute approximation $u^k_{coarse} = u_{h,p}$ to the primal problem on the current mesh $\mathcal{T}^k_{coarse}$.*
3. *Compute approximation $v^k_{coarse} = v_{h,p}$ to the dual problem on the current mesh $\mathcal{T}^k_{coarse}$.*
4. *Construct a uniformly refined grid $\mathcal{T}^k_{fine}$:*
   - *(a) $\mathcal{T}^k_{fine} = \mathcal{T}^0_{h/2,p}$ (h-adaptivity),*
   - *(b) $\mathcal{T}^k_{fine} = \mathcal{T}^0_{h,p+1}$ (p-adaptivity),*
   - *(c) $\mathcal{T}^k_{fine} = \mathcal{T}^0_{h/2,p+1}$ (hp-adaptivity).*
5. *Compute approximation u^k_{fine} to the primal problem on the fine mesh $\mathcal{T}^k_{fine}$, optimally using a two-grid solver and the coarse mesh solution u^k_{coarse}.*
6. *In the same way compute approximation v^k_{fine} to the dual problem on the fine mesh $\mathcal{T}^k_{fine}$.*

7. *If the approximate error in goal* $|L(u^k_{fine}) - L(u^k_{coarse})|$ *satisfies a given tolerance* TOL*, stop.*

8. *Minimize elementwise contributions to the* product *of projection-based interpolation errors for both the primal and dual problem,*

$$\|u^k_{fine} - \Pi^k_{optimal} u^k_{fine}\|_e \|v^k_{fine} - \Pi^k_{optimal} v^k_{fine}\|_e,$$

analogously to how we minimized the first term $\|u^k_{fine} - \Pi^k_{optimal} u^k_{fine}\|_e$ *in Algorithm 6.2. Now, according to (6.27) we select optimal refinement type for all elements* $K_i \in \mathcal{T}^k_{coarse}$ *by maximizing the* product *of projection-based interpolation error decrease rates*

$$\triangle err_i = \triangle err_{primal,i} \triangle err_{dual,i}, \tag{6.32}$$

where

$$\triangle err_{primal,i} = \|u^k_{fine} - \Pi_{coarse,i} u^k_{fine}\|_e - \|u^k_{fine} - \Pi_{optimal,i} u^k_{fine}\|_e,$$

and

$$\triangle err_{dual,i} = \|v^k_{fine} - \Pi_{coarse,i} v^k_{fine}\|_e - \|v^k_{fine} - \Pi_{optimal,i} v^k_{fine}\|_e.$$

Analogously to Algorithm 6.2 the quantities $\triangle err_{primal,i}$*,* $\triangle err_{dual,i}$ *simplify for pure h- and p-adaptivity due to the nestedness of polynomial spaces corresponding to the coarse and refined elements.*

9. *Determine which elements will be refined using the criteria (6.5) and (6.6).*

10. *Refine selected elements, by* $\mathcal{T}^{k+1}_{coarse} = \mathcal{T}^{k+1}_{h,p}$ *denote the new coarse mesh. Put* $k := k + 1$*.*

11. *Go to 2.*

Let us look at some examples in the next paragraph.

6.4.2 Example: average of solution over a subdomain

To illustrate the advantages of automatic goal-oriented adaptivity, let us return to the model equation (6.17),

$$-\triangle u + k^2 u = f \quad \text{in } \Omega = (0, 1). \tag{6.33}$$

We use the same value of $k = \sqrt{5}$ and the right-hand side function f and Dirichlet boundary conditions are chosen in such a way that the function (6.8),

$$u(x) = \frac{\exp(-500(x - 0.4)^2)}{2},$$

that we already know from Figure 6.4, is the exact solution of (6.33).

This time, however, the goal of the computation will be the *average of the solution* u in the subdomain $\Omega_s = (0.5, 0.6)$ of the domain $\Omega = (0, 1)$, as shown in Figure 6.45.

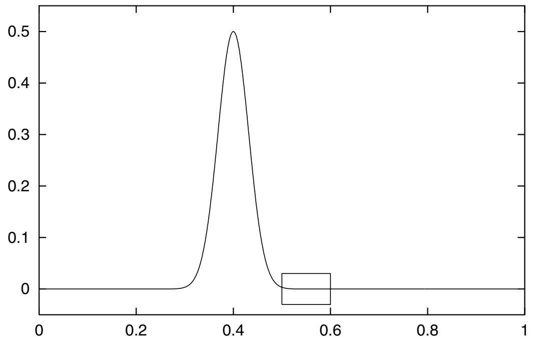

FIGURE 6.45: Goal of computation: average of the solution u over the subdomain $\Omega_s = (0.5, 0.6)$.

The appropriate functional of interest has the form (6.18) from Paragraph 6.3.1,

$$L(u) = \frac{1}{|\Omega_s|} \int_\Omega \chi_{\Omega_s}(x) u(x) \mathrm{d}x. \tag{6.34}$$

6.4.3 Goal-oriented and energy-driven h-adaptivity

The left part of Figures 6.46 – 6.52 shows coarse and fine mesh solutions to the primal problem (solid and dashed lines, respectively). The midpart shows the same quantities for the dual problem and on the right are the corresponding projection-based interpolation error decrease rates $\triangle err_i$,

$$\triangle err_i = ||u^k_{fine} - \Pi_{coarse} u^k_{fine}||_{e,K_i} ||v^k_{fine} - \Pi_{coarse} v^k_{fine}||_{e,K_i}, \tag{6.35}$$

obtained as a product of contributions from both the primal and dual problems. The symbol Π_{coarse} stands for the projection-based interpolation operator on the coarse mesh.

It is interesting to compare these first steps with Figures 6.27 – 6.31 (corresponding to energy-driven h-adaptivity). Convergence rates in the quantity of interest will be shown in Figure 6.60.

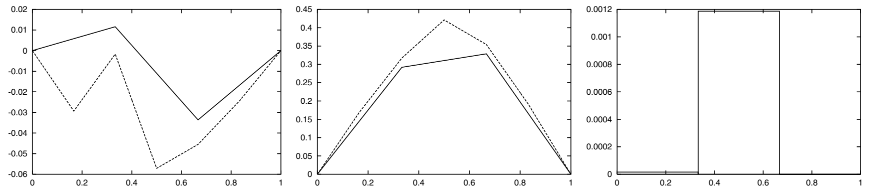
FIGURE 6.46: Step 1: initial coarse and fine mesh solutions for both the primal and dual problems (left and middle, respectively), and corresponding error decrease rates (on the right). Only element K_2 selected for refinement. In contrast to the energy-driven h-adaptive Algorithm 6.2, element K_1 is left unchanged.

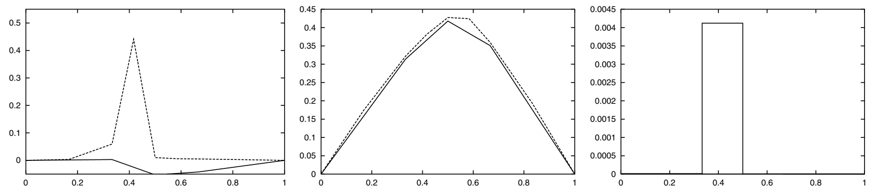
FIGURE 6.47: Step 2: number of DOF = 3. The product of the projection-based interpolation error decrease rates (6.35) is negligible except for a single element at $x = 0.4$. Due to this fact, element K_1 is skipped again.

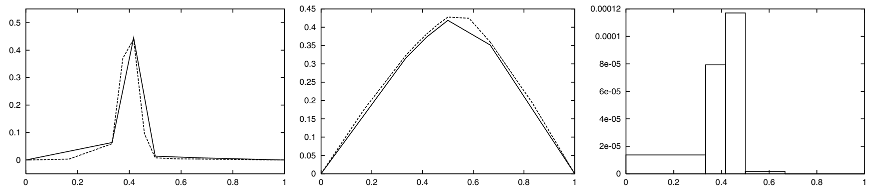
FIGURE 6.48: Step 3: number of DOF = 4. Element K_1 starts to be seen; however, the rate $\triangle err_1$ does not yet exceed the critical level $\triangle err_{max}/3$.

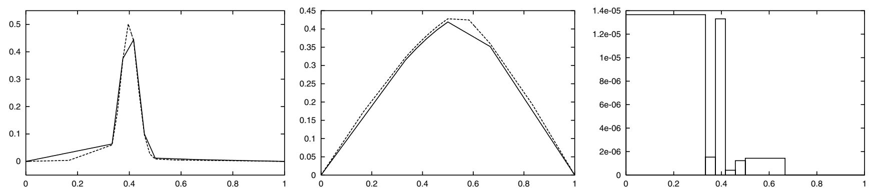

FIGURE 6.49: Step 4: number of DOF = 6. The reduction of error in both the primal and dual problems, resulting from the previous refinements, finally allows us to refine element K_1.

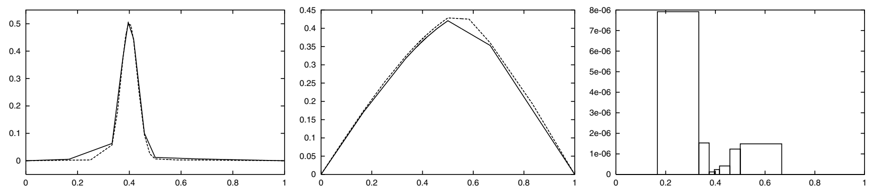

FIGURE 6.50: Step 5: number of DOF = 8. Initially slowed down by skipping the element K_1, now the goal-oriented adaptive algorithm achieves the same resolution in goal as the energy-driven h-adaptive procedure, and it starts to win (compare h-convergence curves in Figure 6.60).

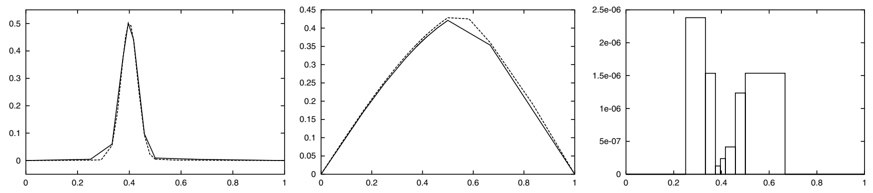

FIGURE 6.51: Step 6: number of DOF = 9. Only elements corresponding to the foot of the exponential peak and the top of the solution of the dual problem are selected for refinement.

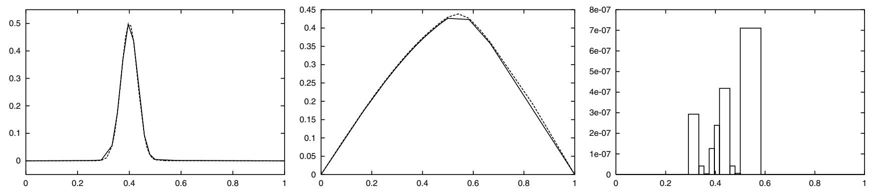

FIGURE 6.52: Step 7: number of DOF = 13. From now on, the goal-oriented algorithm invests new degrees of freedom mainly to the resolution of the region on the right of the peak – in contrast to the standard h-adaptive scheme, which improves the resolution equally on both its sides.

6.4.4 Goal-oriented and energy-driven *hp*-adaptivity

The next series of Figures 6.53 – 6.59 documents the decisions of the *hp*-version of the goal-oriented adaptive Algorithm 6.3 (again we solve problem (6.33) from Paragraph 6.4.2). The initial mesh consists of three equally long linear elements. Again it is interesting to compare these figures with Figures 6.36 – 6.40. Convergence rates in the quantity of interest will be presented later in Figure 6.60.

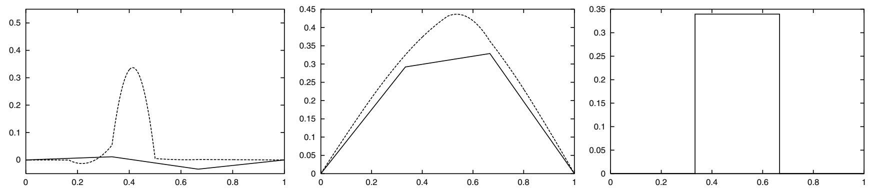

FIGURE 6.53: Step 1: number of DOF = 2. Initial coarse and fine mesh solutions for both the primal and dual problems (left and middle, respectively) and the corresponding error decrease rates (right). Element K_2 selected for pure p-refinement.

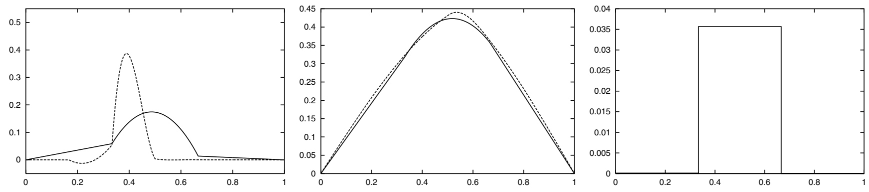

FIGURE 6.54: Step 2: number of DOF = 3. Element K_2 selected for pure p-refinement.

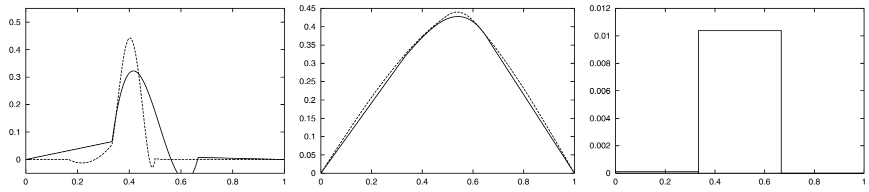

FIGURE 6.55: Step 3: number of DOF = 4. Element K_2 selected for genuine hp-refinement. The reader may notice that in the first two steps, error in the element K_2 was reduced by pure p-refinements only (compare with Figure 6.37).

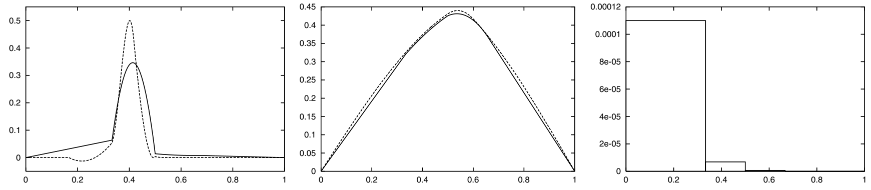

FIGURE 6.56: Step 4: number of DOF = 5. Element K_1 selected for pure p-refinement.

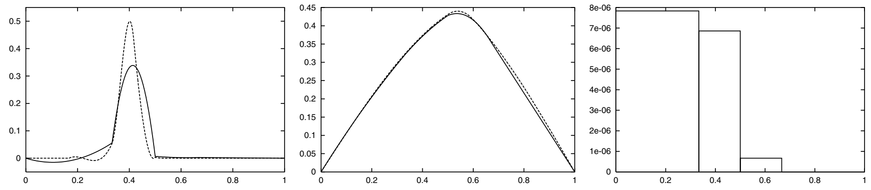

FIGURE 6.57: Step 5: number of DOF = 6. Elements K_1, K_2 selected for pure p-refinement.

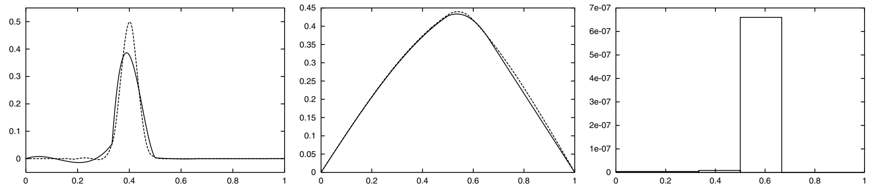

FIGURE 6.58: Step 6: number of DOF = 8. Element K_3 selected for pure p-refinement. From now on, similarly as in the goal-oriented h-adaptive case, degrees of freedom are invested mainly to the region lying on the right from the peak.

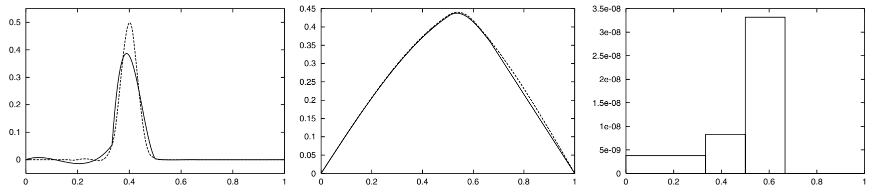

FIGURE 6.59: Step 7: number of DOF = 9. Element K_3 selected for pure p-refinement. At this number of DOF, the resolution in goal is already almost by two orders of magnitude better compared to the energy-driven hp-adaptive scheme and four orders of magnitude better than in energy-driven h-adaptivity.

Comparison of performance of energy-driven and goal-oriented strategies

Convergence in the quantity of interest of the standard (energy-driven) and goal-oriented h- and hp-adaptive algorithms is shown in Figure 6.60. According to the theory, convergence in goal is not necessarily monotone.

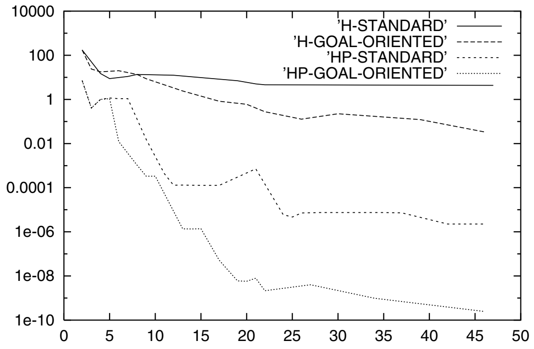

FIGURE 6.60: Convergence in quantity of interest (average of solution u over interval $\Omega_s = (0.5, 0.6)$) with respect to the number of degrees of freedom for standard (energy-driven) and goal-oriented h- and hp-adaptivity in decimal logarithmic scale.

6.5 Automatic goal-oriented hp-adaptivity in two dimensions

The family of automatic goal-oriented h-, p- and hp-adaptive Algorithms 6.3 can be naturally extended into two spatial dimensions. At this point let us say again that this approach to automatic adaptivity is, of course, not the only one possible. The hp-adaptive finite element methods are subject to ongoing active research and various additional strategies are currently investigated. Some of them are based on a posteriori error estimates in the standard sense (i.e., without a finite element computation of reference solutions). To provide a few references to the ongoing research, we mention at least [8, 122, 136, 149, 172]. Many additional references can be found therein.

The computation of reference solutions can be viewed as a special a posteriori error estimation technique that, as with every other approach, has its merits and its disadvantages. The two basic advantages, from our point of view, are that this type of a posteriori error estimate works for a significantly wider class of problems than standard error estimates do. But more significantly, the reference solutions provide an approximation *of the error itself*, i.e., of its concrete *shape* as a function, while standard error estimates can mostly offer information about its *magnitude* only. From the point of view of automatic *hp*-adaptivity this difference is essential. An automatic *hp*-algorithm has to choose the best candidate from among several competitive refinements, and the shape of the function $e_{h,p}$ in element interiors does matter. The price that we may pay for the luxury of having such an amount of information about the error function is in many cases longer computational times.

The results presented in the following are based on the work [185] that is a continuation of a long-term research devoted to automatic *hp*-adaptivity for elliptic problems by Demkowicz et al. [62, 64, 65, 66, 151, 162, 63]. We refer to [63] for a 3D implementation, the technicality of which exceeds the scope of this chapter.

The basic principles of automatic goal-oriented *hp*-adaptivity in two spatial dimensions are analogous to those in one dimension, and so is the goal-oriented adaptive Algorithm 6.3. However, the mesh optimization step in 2D makes the theory as well as the implementation much more difficult. For goal-oriented adaptivity we will perform simultaneous minimization of the projection-based interpolation error of the reference solution to both the primal and dual problems. The reader may find it useful to recall the projection-based interpolation procedure introduced in Chapter 3. For the sake of simplicity, we will work with H^1-conforming approximations only.

6.5.1 Mesh optimization step in two dimensions

The mesh optimization procedure is based on elementwise minimization of the error in goal by means of the approximate inequality (6.27),

$$|L(u_{h/2,p+1}) - L(u_{h,p})| \le C \sum_{K \in \mathcal{T}_{h,p}} \|u_{ref} - \Pi_{h,p} u_{ref}\|_{e,K} \|v_{ref} - \Pi_{h,p} v_{ref}\|_{e,K}. \tag{6.36}$$

One step of the procedure looks as follows (see [185] for more details):

Step 0. Obtain solutions to both the primal and the dual problems on the coarse mesh $\mathcal{T}_{h,p}$ and globally refined mesh $\mathcal{T}_{h/2,p+1}$ (simplifications analogous to those in the one dimensional case (Algorithm 6.3) apply to pure h- and p-adaptivity).

Step 1. Compute element contributions to estimate (6.36),

$$\|u_{h/2,p+1} - \Pi_{h,p} u_{h/2,p+1}\|_{e,K} \|v_{h/2,p+1} - \Pi_{h,p} v_{h/2,p+1}\|_{e,K}, \quad \text{for all } K \in \mathcal{T}_{h,p}. \tag{6.37}$$

Step 2. Determine the element isotropy flags. This step is relevant only for quadrilateral elements. An element is declared to be a candidate for anisotropic refinement if *both the differences* $u_{h/2,p+1} - u_{h,p}$ *and* $v_{h/2,p+1} - v_{h,p}$ represent the same anisotropic behavior. First, the approximate error function to the primal problem,

$$e_K(\boldsymbol{x}) = u_{h/2,p+1}(\boldsymbol{x}) - u_{h,p}(\boldsymbol{x}),$$

is transformed from a physical mesh quadrilateral K to the reference domain K_q by means of an appropriate reference map $\boldsymbol{x}_K(\boldsymbol{\xi}) = (x_{K,1}(\boldsymbol{\xi}), x_{K,2}(\boldsymbol{\xi})) : K_q \to K$. On the reference domain the error function has the form

$$\hat{e}(\boldsymbol{\xi}) = e(\boldsymbol{x}_K(\boldsymbol{\xi})).$$

The H_0^1 error on the element K,

$$e_K^2 = \int_K |\boldsymbol{\nabla}(u_{h/2,p+1} - u_{h,p})|^2 \,\mathrm{d}\boldsymbol{x},$$

can be expressed in reference coordinates $\boldsymbol{\xi}$ as

$$\hat{e}_K^2 = \int_{K_q} \left[a_{11} \left(\frac{\partial \hat{e}}{\partial \xi_1} \right)^2 + 2a_{12} \frac{\partial \hat{e}}{\partial \xi_1} \frac{\partial \hat{e}}{\partial \xi_1} + a_{22} \left(\frac{\partial \hat{e}}{\partial \xi_1} \right)^2 \right] J(\boldsymbol{\xi}) \,\mathrm{d}\boldsymbol{\xi} \qquad (6.38)$$

where

$$a_{ij} = \sum_{k=1}^{2} \frac{\partial x_{K,k}}{\partial \xi_{K,i}} \frac{\partial x_k}{\partial \xi_j}.$$

The Jacobian J is computed in the standard way,

$$J(\boldsymbol{\xi}) = \det \left(\frac{\partial x_{K,i}}{\partial \xi_k} \right) (\boldsymbol{\xi}).$$

Consider the direction ξ_1 first. The solution to the primal problem is declared to behave anisotropically in ξ_1 if two conditions are met:

1. The error function $\hat{e}(\boldsymbol{\xi})$ varies significantly only in a single axial direction,

   $$|a_{12}| \leq C_1(|a_{11}| + |a_{22}|).$$

   By the choice of the constant $C_1 > 0$ we specify how significant the anisotropic behavior must be to invoke anisotropic refinement. In the code we set $C_1 := 0.01$, so that anisotropic refinements occur only if the anisotropy of $\hat{e}(\boldsymbol{\xi})$ is relatively strong. Small values of the constant C_1 prefer isotropic refinements.

2. The anisotropic behavior occurs in the ξ_1-direction,

$$|a_{22}| \le C_2 |a_{11}|,$$

where the constant $C_2 := 0.01$ for similar reasons as above.

Analogously we check the anisotropic behavior of the function $\hat{e}(\boldsymbol{\xi})$ in the direction ξ_2.

For each reference direction ξ_1, ξ_2 the same test is also performed for the error function $v_{h/2,p+1}(\boldsymbol{x}) - v_{h,p}(\boldsymbol{x})$ corresponding to the dual problem. As mentioned above, an anisotropic refinement occurs only if both the tests are successful.

REMARK 6.3 Notice that anisotropic behavior of the error function is investigated *on the reference domain* only. Thus the algorithm will not be able to identify anisotropy when the physical mesh is not aligned with boundary (or internal) layers. This is one of the few points where the user still can improve the performance of the otherwise fully automatic adaptive algorithm. $\square$

Step 3. Determine optimal refinement for each edge e in the current mesh $\mathcal{T}_{h,p}$. Loop through the edges, and on each of them select the most competitive refinement (either pure p-refinement, or h-refinement with an optimal redistribution of polynomial orders on the subintervals). The theory suggests using the $H^{1/2}_{00}$-norm, which we again approximate by means of a weighted H^1_0-norm (see Section 3.1).

Step 4. Following the 1D strategy further, we determine the maximum edge error decrease rate and identify all edges with an error decrease rate greater than or equal to (for example) one third of the maximal one; those edges are going to be refined.

Step 5. Use the information about edge h-refinements (and the element isotropy flags for quadrilateral elements) to make decisions about h-refinements for all elements. A triangular element is broken into four whenever at least one of its edges has been marked for h-refinement. Similarly, quadrilaterals are broken into four in the isotropic case if at least one of their edges is h-refined. In the anisotropic case, the h-refined edges display anisotropic behavior, and the element is going to be refined into two. In the hypothetical case, when the anisotropic behavior differs from the anticipated refinement of edges (this does not often happen in practice), we perform isotropic 4-refinement. At this point, the topology of the new mesh is determined.

Step 6. Determine optimal orders of approximation for all element interiors, monitoring the decrease rate of the product of the element interpolation errors

(6.37) for the primal and the dual problem. Solve locally, one element at a time, the discrete projection problems

$$\sqrt{\int_K \left(\nabla(u_{h/2,p+1} - \tilde{u} - u_{opt})\right)^2 \,\mathrm{d}\boldsymbol{x}} \sqrt{\int_K \left(\nabla(v_{h/2,p+1} - \tilde{v} - v_{opt})\right)^2 \,\mathrm{d}\boldsymbol{x}} \to \min, \tag{6.39}$$

where $\tilde{u}, \tilde{v}$ are lifts of already known interpolants along the optimally refined edges, and the unknown bubble-parts of the projections on the optimally refined element, u_{opt} and v_{opt}, vanish on the element boundary.

Begin the optimization process with values that are suggested by already known orders on the element boundary and the minimum rule. Monitor the product error decrease rate

$$\triangle err = \frac{||u_{h/2,p+1} - \tilde{u} - u_{old}|| - ||u_{h/2,p+1} - \tilde{u} - u_{new}||}{ndof_{new} - ndof_{old}} \frac{||v_{h/2,p+1} - \tilde{v} - v_{old}|| - ||v_{h/2,p+1} - \tilde{v} - v_{new}||}{ndof_{new} - ndof_{old}}, \tag{6.40}$$

where u_{old}, u_{new} and v_{old}, v_{new} stand for the previous and current solutions to projection problem (6.39), and $ndof_{old}, ndof_{new}$ denote the number of degrees of freedom in the interior of the refined element. The order of approximation cannot exceed the order corresponding to the globally *hp*-refined mesh $\mathcal{T}_{h/2,p+1}$ used for the computation of the reference solutions $u_{ref} = u_{h/2,p+1}$ and $v_{ref} = v_{h/2,p+1}$.

The local optimization problem is viewed as resolved when two conditions are satisfied:

1. The product projection error (6.39) is smaller than the projection error corresponding to the unrefined element and the current order of approximation.

2. The error decrease rate (6.40) is greater than or equal to (for example) one third of the initial rate $\triangle err_0$.

Step 7. It may happen that after the new element orders have been specified, the orders on the edges do not conform to the *minimum rule* (i.e., are not equal to the minimum of the orders in the interior of the adjacent elements). In this case, we increase the orders on the edges appropriately.

6.5.2 Example: singular solution in the L-shape domain

We consider the standard L-shape domain problem (see, e.g., [64, 185]) in the domain Ω shown in Figure 6.61.

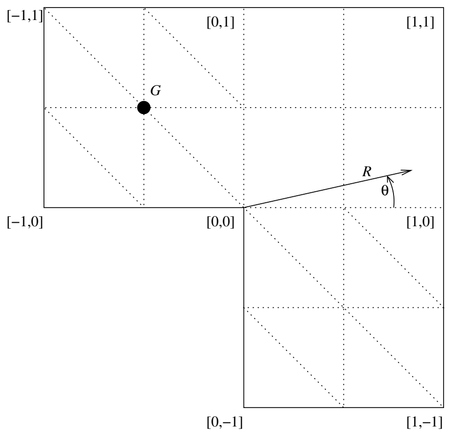

FIGURE 6.61: Geometry and initial hybrid quadrilateral/triangular mesh.

The Laplace equation

$$-\triangle u = 0 \tag{6.41}$$

is solved in Ω, with Dirichlet boundary conditions $u(\boldsymbol{x}) = \phi(\boldsymbol{x})$ for all $\boldsymbol{x} \in \partial\Omega$. Function ϕ is chosen to be compatible with the harmonic function

$$u(x_1, x_2) = R^{2/3} \sin(2\theta/3 + \pi/3) \tag{6.42}$$

where (R, θ) are standard polar coordinates. The reason why this example is difficult for finite element schemes is that the solution has singular derivatives at the re-entrant corner. The exact solution is shown in Figure 6.62.

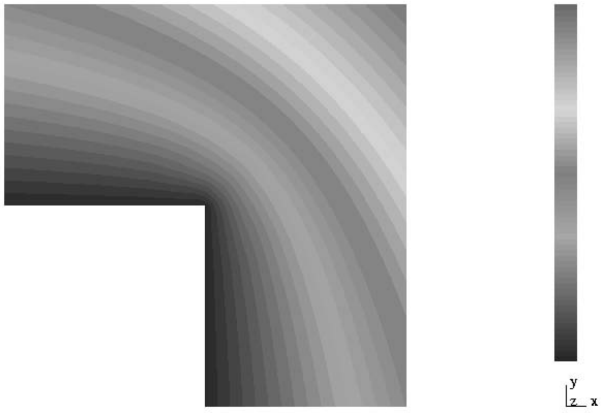

FIGURE 6.62: Exact solution u to the problem (6.41). The values of the function u lie in the interval $[0, 1.26]$.

The goal of the computation will be, for example, the average value of the solution u over a small neighborhood Ω_s of the point $G = [-0.5, 0.5]$. The linear functional of interest has the form (6.18) from Paragraph 6.3.1,

$$L(u) = \frac{1}{|\Omega_s|} \int_\Omega \chi_{\Omega_s}(\boldsymbol{x}) u(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}. \tag{6.43}$$

The neighborhood Ω_s of point G is defined as follows: let us consider the mesh edges e_j, $j = 1, 2, \ldots, 6$, starting at G. For each of them we consider a point Q_j, which lies at e_j and whose distance from G is

$$|Q_j - G| = \frac{|e_j|}{2^D}, \tag{6.44}$$

$D = 6$. The subdomain Ω_s is defined as the convex envelope of points $Q_1, Q_2, \ldots, Q_6$. This choice of the shape of neighborhood Ω_s has been motivated by the ease of numerical integration over Ω_s only. Exact solution to the dual problem is depicted in Figure 6.63.

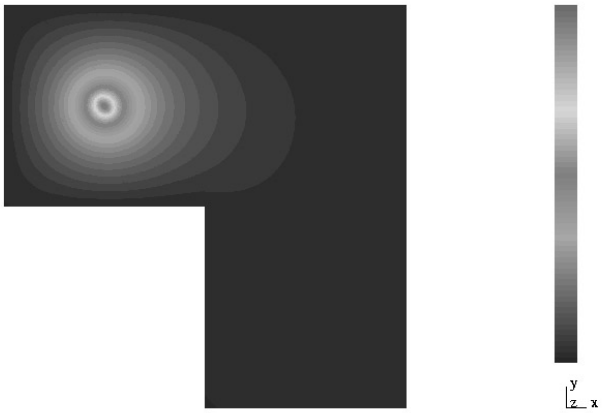

FIGURE 6.63: Exact solution v to the dual problem corresponding to the linear functional of interest (6.43). Its values lie in the interval $[0, 0.4066]$.

6.5.3 Goal-oriented and energy-driven h-adaptivity

First let us have a look at the performance of the h-version of the adaptive Algorithm 6.2 enhanced with the two-dimensional mesh optimization step 6.5.1. Numerical results presented in this and the following paragraphs are obtained by means of the goal-oriented *hp*-adaptive code [185], which is based on the original energy-driven *hp*-adaptive code [64].

The computation begins from the initial mesh depicted in Figure 6.61. All elements are quadratic ($p = 2$). Magnitude of the gradient $\nabla(u_{h/2,p} - u_{h,p})$ of the error function $u_{h/2,p} - u_{h,p}$, which guides the refinement strategy, and a few first optimal meshes are shown in Figures 6.64 – 6.67. Convergence in the quantity of interest will be shown later in Figure 6.86.

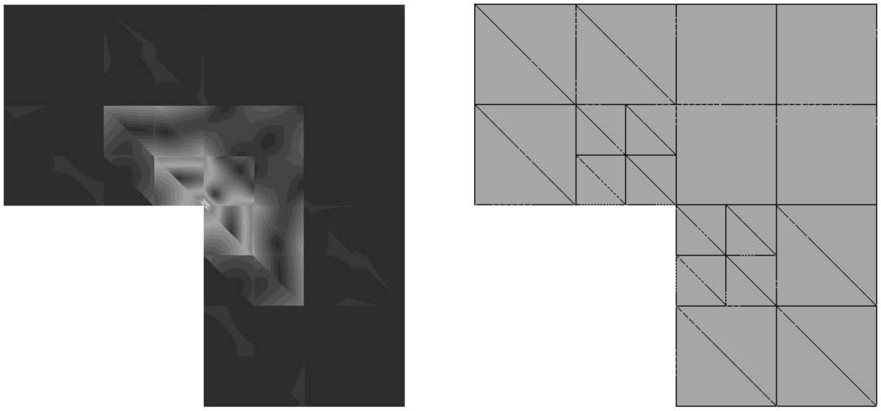

FIGURE 6.64: Energy-based h-adaptivity. Step 1: magnitude of gradient $\nabla(u_{h/2,p} - u_{h,p})$ (with values in the interval $[2.03 \cdot 10^{-4}, 0.5639]$) corresponding to the initial mesh (left), and the optimal mesh after the first mesh optimization step (right).

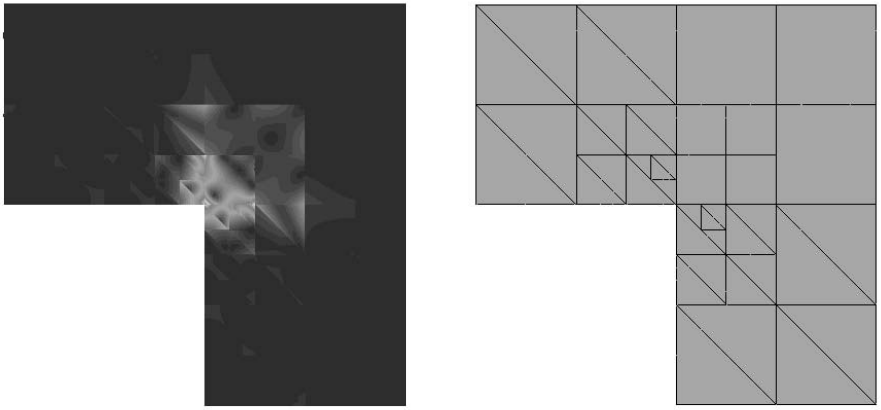

FIGURE 6.65: Energy-based h-adaptivity. Step 2: magnitude of gradient $\nabla(u_{h/2,p} - u_{h,p})$ (with values in the interval $[1.81 \cdot 10^{-4}, 0.6000]$) corresponding to the first optimal mesh (left), and the next optimal mesh (right).

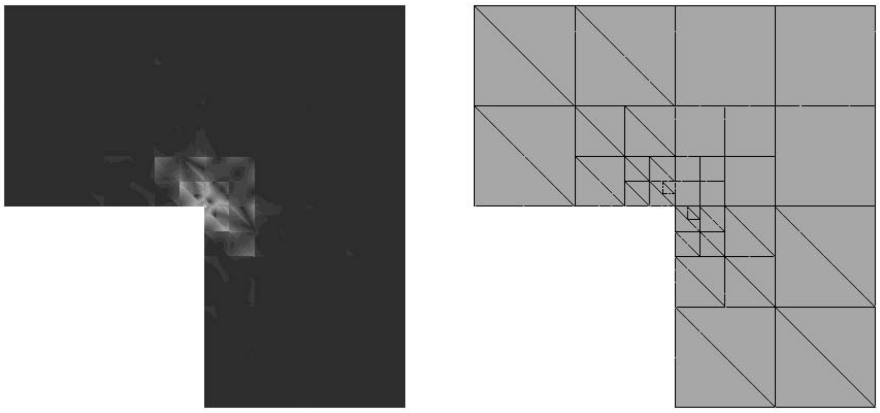

FIGURE 6.66: Energy-based h-adaptivity. Step 3: magnitude of gradient $\nabla(u_{h/2,p}-u_{h,p})$ (with values in the interval $[5.48\cdot 10^{-5}, 0.7400]$) corresponding to the second optimal mesh (left), and the next optimal mesh (right).

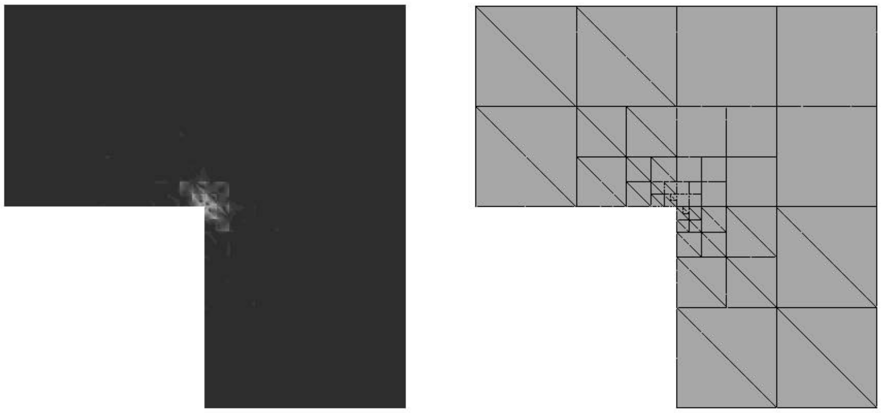

FIGURE 6.67: Energy-based h-adaptivity. Step 4: magnitude of gradient $\nabla(u_{h/2,p}-u_{h,p})$ (with values in the interval $[1.56\cdot 10^{-5}, 0.9340]$) corresponding to the third optimal mesh (left), and the next optimal mesh (right).

REMARK 6.4 The reader may notice that the optimal meshes are symmetric with respect to the line $x_1 = x_2$. This is due to the fact that the exact solution u to the original problem (6.42) and consequently the gradient of the error functions $|u_{h/2,p} - u_{h,p}|$ are symmetric. Such symmetry is, of course, an extremely helpful feature for debugging of hp-adaptive codes. $\square$

Goal-oriented h-adaptivity

In Figures 6.68 – 6.71 we show results obtained using the h-version of the goal-oriented Algorithm 6.3, combined with the mesh optimization step 6.5.1. It starts from the same initial mesh as in the previous case. This time we visualize the product $|\nabla(u_{h/2,p} - u_{h,p})||\nabla(v_{h/2,p} - v_{h,p})|$. Convergence in the quantity of interest will be shown in Figure 6.86.

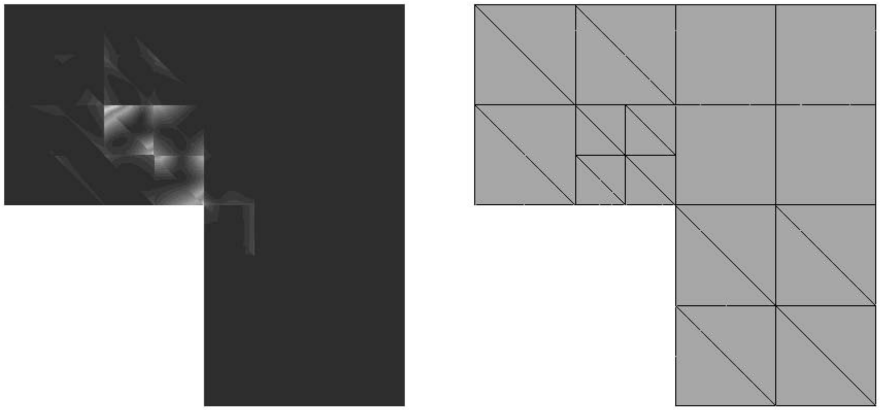

FIGURE 6.68: Goal-oriented h-adaptivity. Step 1: product $|\nabla(u_{h/2,p} - u_{h,p})|\ |\nabla(v_{h/2,p} - v_{h,p})|$ (with values in the interval $[0, 0.4389]$) on the initial mesh (left), and the optimal mesh after the first mesh optimization step (right).

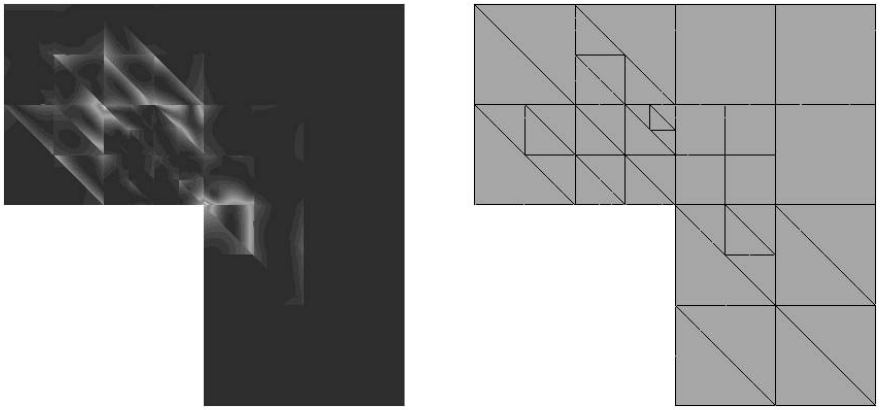

FIGURE 6.69: Goal-oriented h-adaptivity. Step 2: product $|\nabla(u_{h/2,p} - u_{h,p})|\ |\nabla(v_{h/2,p} - v_{h,p})|$ (with values in the interval $[0, 0.0883]$) on the first optimal mesh (left), and the next optimal mesh (right).

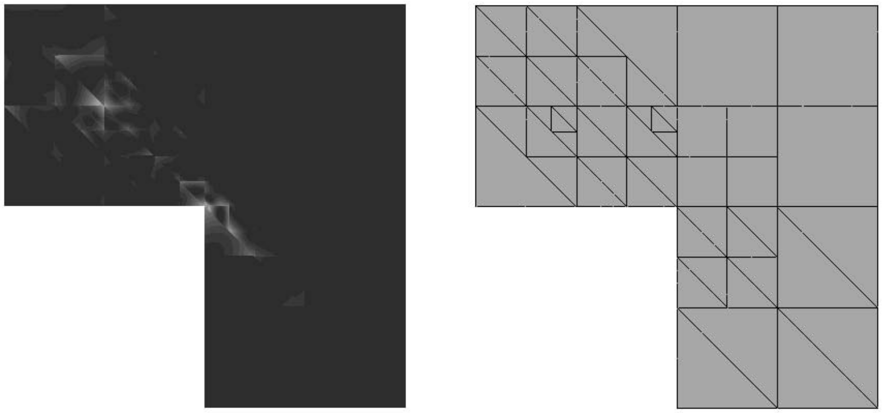

FIGURE 6.70: Goal-oriented h-adaptivity. Step 3: product $|\nabla(u_{h/2,p} - u_{h,p})|\ |\nabla(v_{h/2,p} - v_{h,p})|$ (with values in the interval $[0, 0.1294]$) on the second optimal mesh (left), and the next optimal mesh (right).

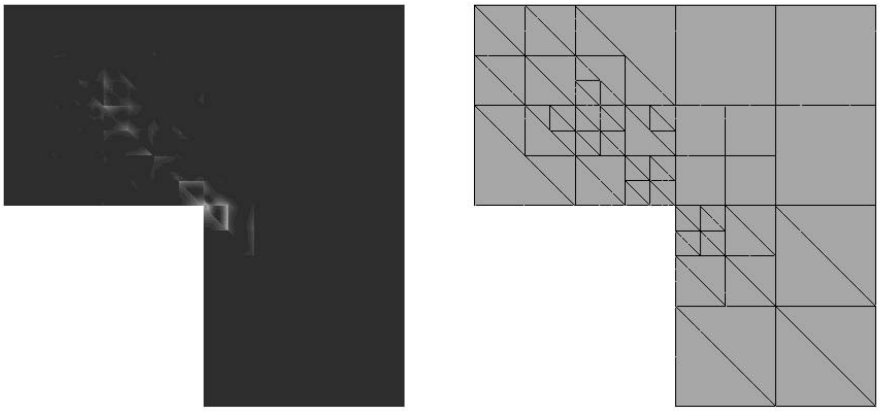

FIGURE 6.71: Goal-oriented h-adaptivity. Step 4: product $|\nabla(u_{h/2,p} - u_{h,p})|\ |\nabla(v_{h/2,p} - v_{h,p})|$ (with values in the interval $[0, 0.1381]$) on the third optimal mesh (left), and the next optimal mesh (right).

6.5.4 Goal-oriented and energy-driven *hp*-adaptivity

Finally, we arrive at the *hp*-versions of the energy-based and goal-oriented adaptive Algorithms 6.2 and 6.3 in 2D.

An interesting new issue in 2D is the visualization of *hp*-meshes, for which we will use a comprehensive color code [64], illustrated in Figure 6.72.

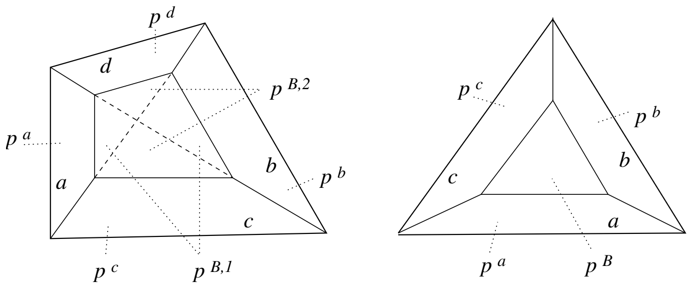

FIGURE 6.72: Color code for the visualization *hp*-meshes in 2D. Areas associated with the symbols p^a, p^b, p^c and p^d show the polynomial order of corresponding edge functions. The middle section is reserved for bubble functions. For quadrilateral elements the middle part is further split to visualize directional orders of approximation.

Again we begin with the energy-driven scheme and let it start from the initial mesh depicted in Figure 6.61 with all second-order elements. The magnitude of the gradient $\nabla(u_{h/2,p+1} - u_{h,p})$ of the error function and a few first optimal meshes are shown in Figures 6.74 – 6.77.

FIGURE 6.73: Color scale for the order of polynomial approximation, starting at $p = 1$ (left) and ending at $p = 8$.

Convergence in the quantity of interest is shown in Figure 6.86.

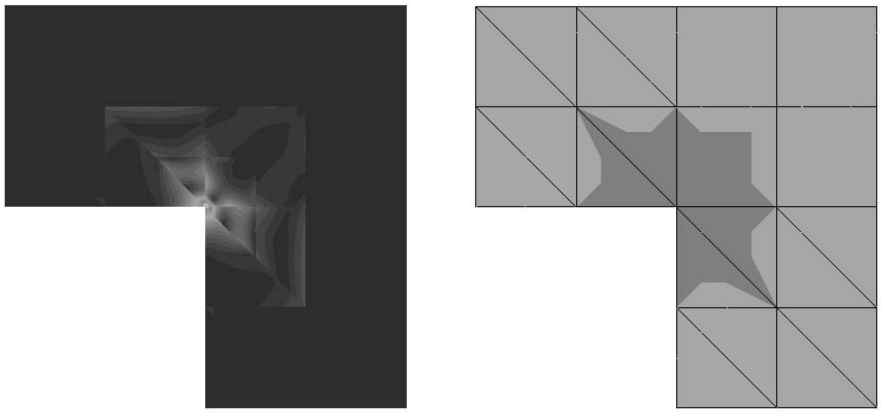

FIGURE 6.74: Energy-based *hp*-adaptivity. Step 1: product $|\nabla(u_{h/2,p+1} - u_{h,p})|\,|\nabla(v_{h/2,p+1} - v_{h,p})|$ (with values in the interval $[1.56 \cdot 10^{-4}, 1.3116]$) corresponding to the initial mesh (left), and the optimal mesh after the first mesh optimization step (right).

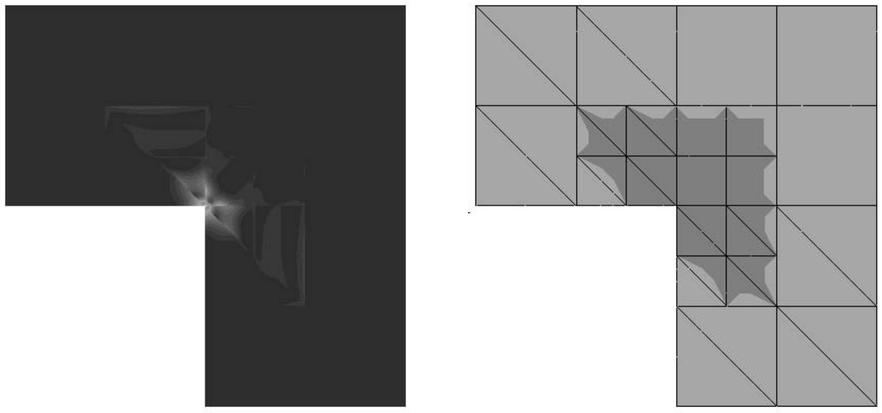

FIGURE 6.75: Energy-based *hp*-adaptivity. Step 2: product $|\nabla(u_{h/2,p+1} - u_{h,p})|\,|\nabla(v_{h/2,p+1} - v_{h,p})|$ (with values in the interval $[4.86 \cdot 10^{-5}, 1.387]$) corresponding to the first optimal mesh (left), and the next optimal mesh (right).

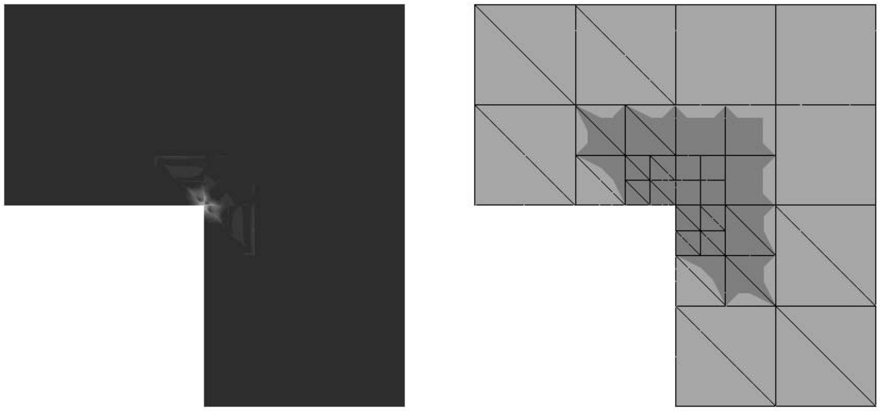

FIGURE 6.76: Energy-based *hp*-adaptivity. Step 3: product $|\nabla(u_{h/2,p+1} - u_{h,p})|\ |\nabla(v_{h/2,p+1} - v_{h,p})|$ (with values in the interval $[4.10 \cdot 10^{-5}, 1.7475]$) corresponding to the second optimal mesh (left), and the next optimal mesh (right).

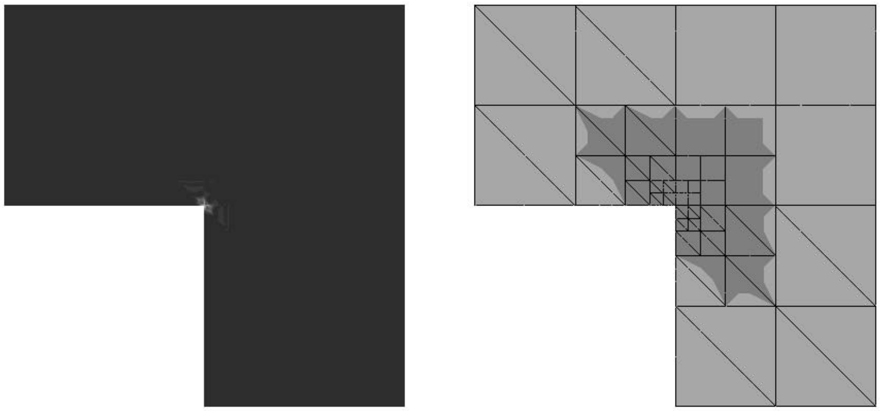

FIGURE 6.77: Energy-based *hp*-adaptivity. Step 4: product $|\nabla(u_{h/2,p+1} - u_{h,p})|\ |\nabla(v_{h/2,p+1} - v_{h,p})|$ (with values in the interval $[2.08 \cdot 10^{-5}, 2.2029]$) corresponding to the third optimal mesh (left), and the next optimal mesh (right).

Goal-oriented *hp*-adaptivity

In this paragraph we finally have an opportunity to see the performance of the goal-oriented *hp*-adaptive Algorithm 6.3 combined with the mesh optimization step 6.5.1. Similarly as in the goal-oriented *h*-adaptive case we depict the product $|\nabla(u_{h/2,p+1} - u_{h,p})||\nabla(v_{h/2,p+1} - v_{h,p})|$. We show this quantity for the first two optimization steps only because it quickly vanishes from the visible scale. Several optimal meshes are shown in Figures 6.78 – 6.83.

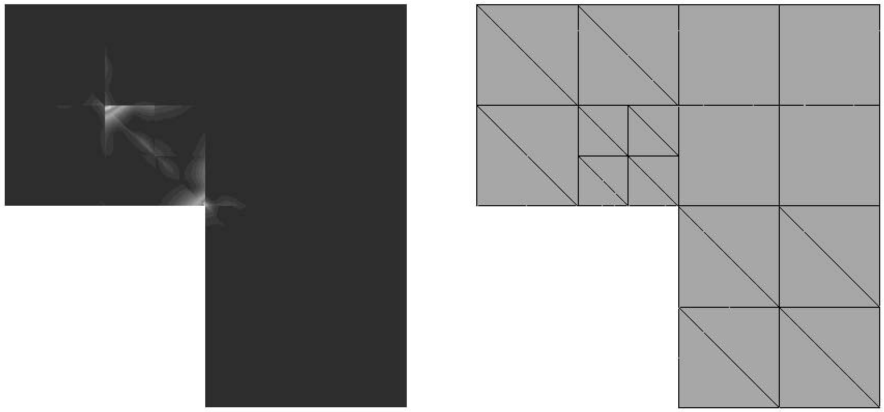

FIGURE 6.78: Goal-oriented *hp*-adaptivity. Step 1: product $|\nabla(u_{h/2,p+1} - u_{h,p})|\,|\nabla(v_{h/2,p+1} - v_{h,p})|$ (with values in the interval $[0, 1.064]$) corresponding to the initial mesh (left), and the optimal mesh after the first mesh optimization step (right).

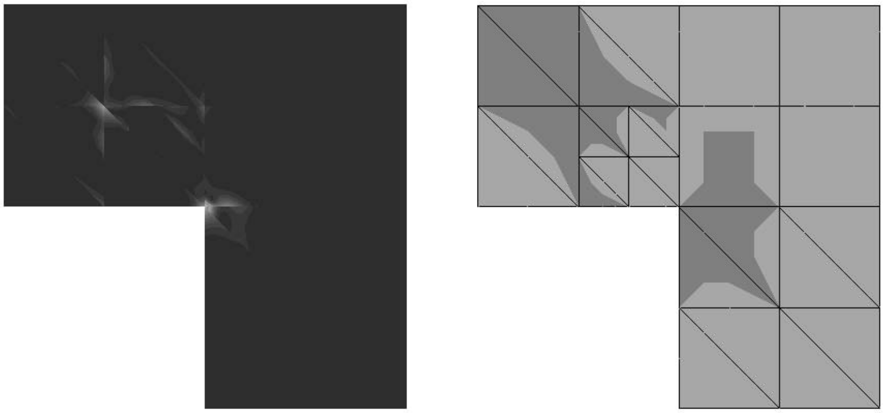

FIGURE 6.79: Goal-oriented *hp*-adaptivity. Step 2: product $|\nabla(u_{h/2,p+1} - u_{h,p})|\,|\nabla(v_{h/2,p+1} - v_{h,p})|$ (with values in the interval $[0, 0.4545]$) corresponding to the first optimal mesh (left), and the next optimal mesh (right).

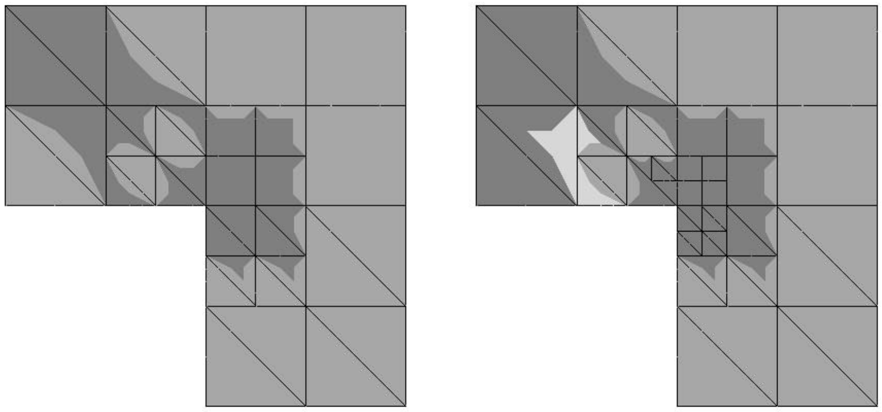

FIGURE 6.80: Goal-oriented *hp*-adaptivity. Meshes after third and fourth optimization steps.

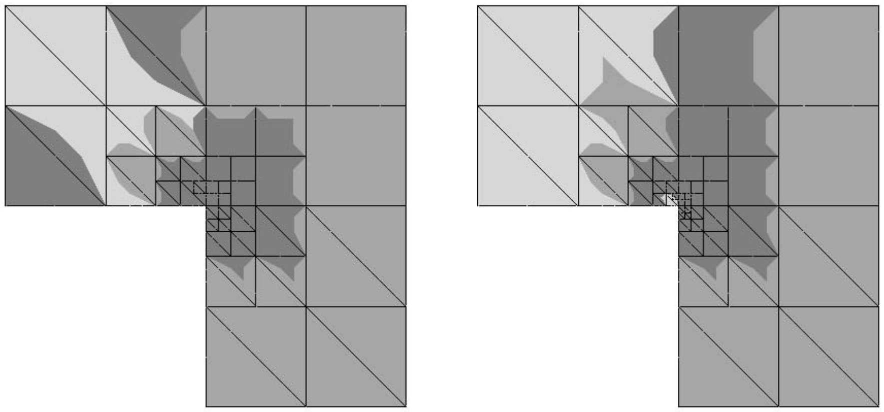

FIGURE 6.81: Goal-oriented *hp*-adaptivity. Meshes after fifth and sixth optimization steps.

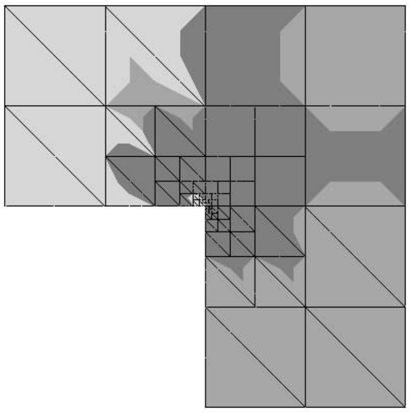
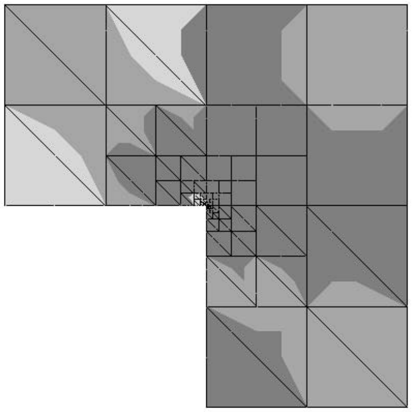
FIGURE 6.82: Goal-oriented *hp*-adaptivity. Meshes after seventh and eighth optimization steps.

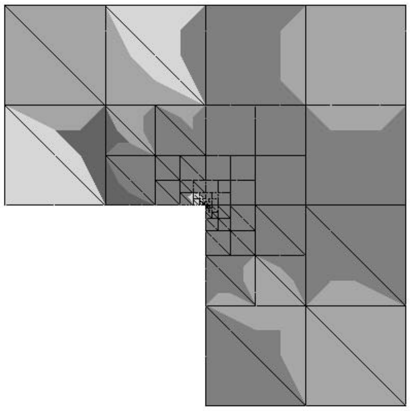
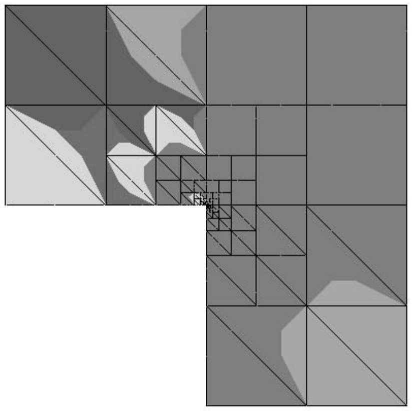
FIGURE 6.83: Goal-oriented *hp*-adaptivity. Meshes after ninth and tenth optimization steps.

6.5.5 Comparison of convergence in the quantity of interest

Finally, let us compare the performance of all the adaptive approaches presented in terms of the convergence in the quantity of interest (recall that our goal is the average of the solution u over a small subdomain Ω_s of the point $G = [-0.5, 0.5]$. It is interesting to select (for example) a relative error level of 10^{-6} and observe how many degrees of freedom each scheme needs to achieve it. This is shown in Figures 6.84 and 6.85.

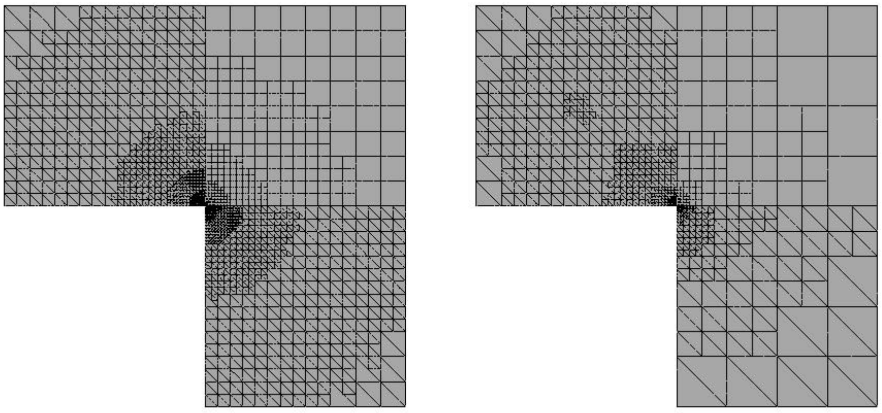

FIGURE 6.84: Resulting optimal meshes, *h*-adaptivity with quadratic elements. The energy-driven scheme (left) needed **4909 DOF** and 20 levels of refinement. The goal-oriented algorithm (right) achieved the prescribed precision in goal with **2652 DOF** and 17 refinement levels. Notice that *20 levels of refinement mean a reduction of h by the factor of one million.*

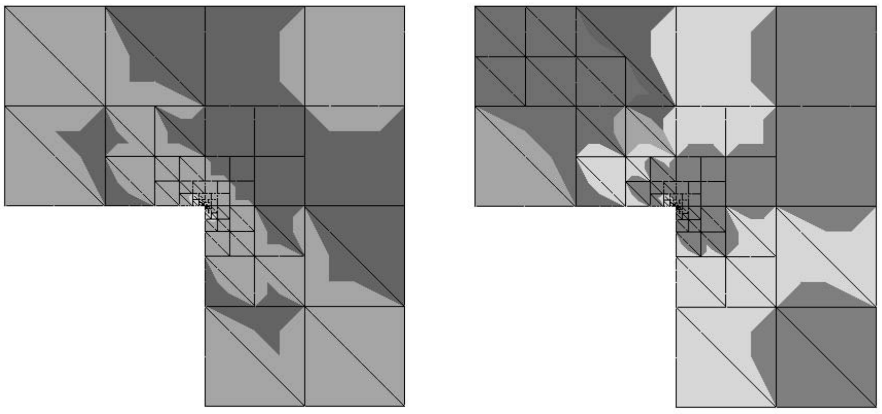

FIGURE 6.85: Resulting optimal meshes delivered by the *hp*-adaptive algorithms. The energy-driven scheme (left) needed **2310 DOF** and 18 refinement levels, the goal-oriented strategy (right) achieved the same precision *with only* **1273 DOF** *and 13 levels of refinement.*

Let us mention that the energy-driven *hp*-adaptive scheme (on the left of Figure 6.85) fully automatically reduces both the element size and the order of polynomial approximation toward the singularity in a way that exactly corresponds to the theory.

Figure 6.86 shows the history of the relative error in goal for all four tested approaches. The x-axis represents the number of degrees of freedom.

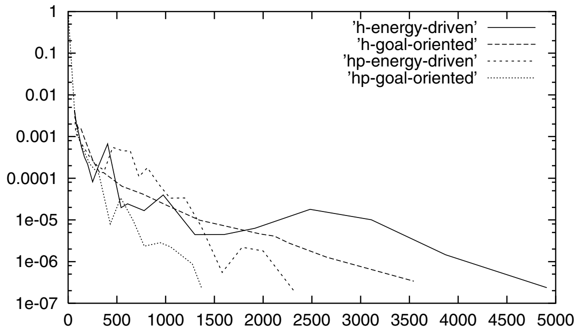

FIGURE 6.86: Relative error in the quantity of interest with respect to the exact solution.

A more difficult example related to a complex-valued elliptic problem rooted in axisymmetric Maxwell's equations, where goal-oriented *hp*-adaptivity turned out to be the only reasonable way to resolve the problem, can be found in [185].

REMARK 6.5 The reader may ask what role the choice of the parameter D plays in the definition (6.43) of the subdomain Ω_s. According to our experience, the influence is not extremely significant. We document this observation by a series of computations. The final optimal meshes corresponding to the choices of $D = 6, 10, 15$ and 20 are presented in Figures 6.87 and 6.88, and the convergence curves in Figure 6.89.

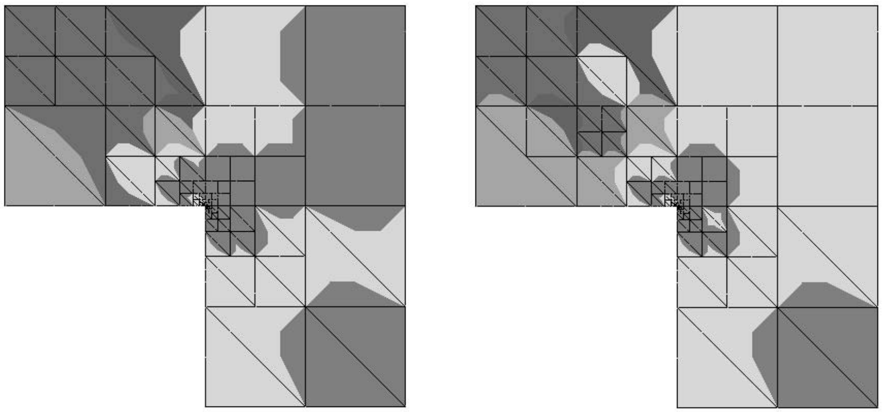

FIGURE 6.87: Final optimal meshes needed to obtain relative error in goal of 10^{-6} for various parameters D from (6.43): $D = 6$ (left, 1273 DOF) and $D = 10$ (right, 1444 DOF).

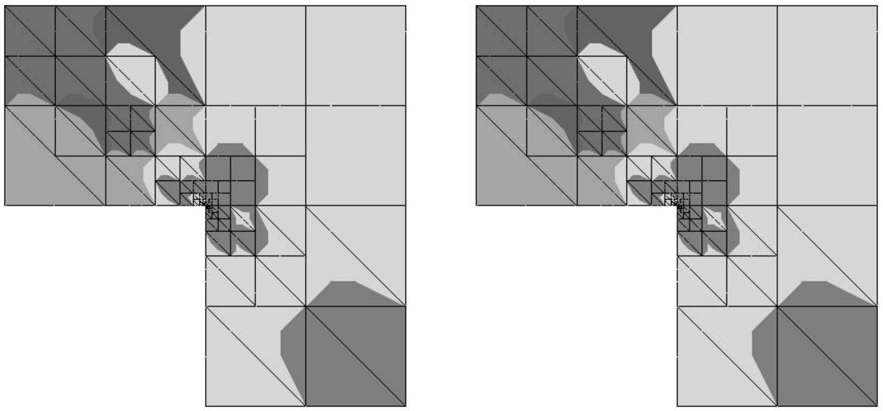

FIGURE 6.88: Final optimal *hp*-meshes needed to obtain relative error in goal of 10^{-6} for various parameters D from (6.43): $D = 15$ (left, 1444 DOF) and $D = 20$ (right, 1435 DOF).

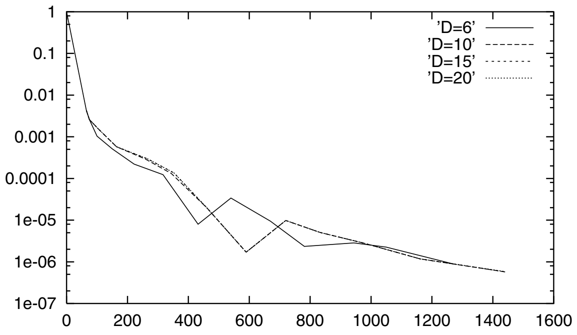

FIGURE 6.89: Relative error with respect to the exact solution in goal (goal-oriented *hp*-adaptive scheme) for various values of the parameter D in (6.43).

The conclusion, that the choice of the parameter D does not influence the adaptive procedure significantly, extends to various other quantities of interest that are similar to (6.43). $\square$

References

[1] M. Abramowitz, I.A. Stegun, *Handbook of Mathematical Functions*, Applied Mathematics Series 55, National Bureau of Standards, Washington, CD, 1964.

[2] S. Adjerid, J.E. Flaherty, A moving finite element method with error estimation and refinement for one-dimensional time dependent partial differential equations, SIAM J. Numer. Anal. 23 (1986), 778–796.

[3] S. Adjerid, J.E. Flaherty, Y.J. Wang, A posteriori error estimation with finite element methods of lines for one-dimensional parabolic systems, Tech. Report 91-1, Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, 1991.

[4] A. Ahagon, K. Fujiwara, T. Nakata, Comparison of various kinds of edge elements for electromagnetic field analysis, IEEE Trans. Magn. 32 (1996), 898–901.

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

[6] M. Ainsworth, J. Coyle, Hierarchic finite element bases on unstructured tetrahedral meshes, May 14, 2002, in review.

[7] M. Ainsworth, J. Coyle, Conditioning of hierarchic p-version Nédélec elements on meshes of curvilinear quadrilaterals and hexahedra, September 28, 2002, in review.

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

[9] P. Alfeld, R.E. Barnhill, A transfinite C^2 interpolant over triangles, Rocky Mountain J. Math. 14 (1984), 17–39.

[10] E. Anderson et al., *LAPACK Users' Guide*, 2nd ed., SIAM, Philadelphia, PA, 1995.

[11] O. Axelsson, V.A. Barker, *Finite Element Solution of Boundary Value Problems*, Classics in Applied Mathematics 35, SIAM, Philadelphia, PA, 2001.

[12] I. Babuška, R. Durán, R. Rodríguez, Analysis of the efficiency of an a posteriori error estimator for linear triangular finite elements, SIAM J. Numer. Anal. 29 (1992), 947–964.

[13] I. Babuška, M. Feistauer, P. Šolín, On one approach to a posteriori error estimates for evolution problems solved by the method of lines, Numer. Math. 89 (2001), 225–256, online DOI 10.1007/s002110100228.

[14] I. Babuška, W. Gui, Basic principles of feedback and adaptive approaches in the finite element method, Comput. Methods Appl. Mech. Engrg. 55 (1986), 27–42.

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

[16] I. Babuška, S. Ohnimus, A posteriori error estimation for the semidiscrete finite element method of parabolic differential equations, Comput. Methods Appl. Mech. Engrg. 190 (2001), 4691–4712.

[17] I. Babuška, W.C. Rheinboldt, A-posteriori error estimates for the finite element method, Internat. J. Numer. Methods Engrg. 12 (1978), 1597–1615.

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

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

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

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

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

[23] I. Babuška et al., A-posteriori error estimation for finite element and generalized finite element method, TICAM Report 98-01, The University of Texas at Austin, 1998.

[24] R.E. Bank, *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations. Users' Guide 7.0*, Frontiers in Applied Mathematics 15, SIAM, Philadelphia, PA, 1994.

[25] R.E. Bank, D.J. Rose, Parameter selection for Newton-like methods applicable to nonlinear partial differential equations, SIAM J. Numer. Anal. 17 (1980), 806–822.

[26] R.E. Bank, A. Weiser, Some a posteriori error estimators for elliptic partial differential equations, Math. Comp. 44 (1985), 283–301.

[27] R.E. Barnhill, W.J. Gordon, D.H. Thomas, Decomposition methods for the approximate integration of multivariate functions, Research Report, General Motors, Warren, MI.

[28] R.E. Barnhill, F.F. Little, Three- and four-dimensional surfaces, Rocky Mountain J. Math. 14 (1984), 77–102.

[29] R. Becker, R. Rannacher, Weighted a posteriori error control in FE method, in: Proc. ENUMATH-95, Paris, September 1995.

[30] R. Becker, R. Rannacher, A feedback approach to error control in finite elements methods: Basic analysis and examples, East-West J. Numer. Math. 4 (1996), 237–264.

[31] J. Berntsen, R. Cools, T.O. Espelid, Algorithm 720. An algorithm for adaptive cubature over a collection of 3-dimensional simplices, ACM Trans. Math. Software 19 (1993), 320–332.

[32] M. Bieterman, I. Babuška, An adaptive method of lines with error control for parabolic equations of the reaction-diffusion type, J. Comput. Phys. 63 (1986), 33–66.

[33] A. Bossavit, Un nouveau point de vue sur les éléments finis mixtes, Bull. Soc. Math. Appl. Industr. (1989), 25–35.

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

[35] A. Bossavit, J. Verite, A mixed FEM-BEM method to solve 3D eddy current problems, IEEE Trans. Magn. MAG-18 (1982), 431–435.

[36] M. Brdička, *Continuum Mechanics*, ČSAV, Prague, 1959.

[37] K.E. Brenan, S.L. Campbell, L.R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, Classics in Applied Mathematics 14, SIAM, Philadelphia, PA, 1996.

[38] K.E. Brenner, L.R. Scott, *The Mathematical Theory of Finite Element Methods*, Springer-Verlag, New York, 1994.

[39] F. Brezzi, M. Fortin, *Mixed and Hybrid Finite Element Methods*, Springer-Verlag, Berlin, 1991.

[40] P.N. Brown, G.D. Byrne, A.C. Hindmarsh, VODE, a variable-coefficient ODE solver, SIAM J. Sci. Statist. Comput. 10 (1989), 1038–1051.

[41] P.N. Brown, A.C. Hindmarsh, Reduced storage matrix methods in stiff ODE systems, J. Appl. Math. Comput. 31 (1989), 40–91.

[42] J.C. Butcher, *The Numerical Analysis of Ordinary Differential Equations*, John Wiley & Sons, New York, 1987.

[43] C. Carstensen, S.A. Funken, Constants in Clement-interpolation error and residual based a posteriori estimates in finite element methods, Bericht 97-11, Christian-Albrechts-Universität zu Kiel, August 1997.

[44] F. Chalot, T.J.R. Hughes, Analysis of hypersonic flows in thermomechanical equilibrium by application of the Galerkin least square formulation, In: Proc. 2nd International Conference for Industrial and Applied Mathematics (R.E. O'Malley, ed.), SIAM, Philadelphia, PA, 1992, 146–159.

[45] H.S. Chen, B. Li, Superconvergence analysis and error expansion for the Wilson non-conforming finite element, Numer. Math. 69 (1994), 125–140.

[46] Y.K. Cheung, S.H. Lo, A.Y.T. Leung, *Finite Element Implementation*, Blackwell Science, Oxford, 1996.

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

[48] F. Cirak, E. Ramm, A posteriori error estimation and adaptivity for linear elasticity using the reciprocal theorem, Comput. Methods Appl. Mech. Engrg. 156 (1998), 351–362.

[49] P. Clément, Approximation by finite element functions using local regularization, RAIRO Anal. Numér. 9 (1975), 77–84.

[50] A. Coddington, N. Levinson, *Ordinary Differential Equations*, McGraw-Hill, New York, 1955.

[51] J.W. Cooley, J.W. Tukey, An algorithm for the machine calculation of complex Fourier series, Math. Comp. 19 (1965), 297–301.

[52] R. Cools, Constructing cubature formulae: the science behind the art, Acta Numer. 8 (1997), 1–54.

[53] R. Cools, A. Haegemans, Construction of fully symmetric cubature formulae of degree $4k-3$ for fully symmetric planar region, J. Comput. Appl. Math. 17 (1987), 173–180.

[54] R. Cools, A. Haegemans, Algorithm 8xx. CUBPACK: a package for automatic cubature; framework description, ACM Trans. Math. Software (2003), in press.

[55] S.A. Coons, Surfaces for computer aided design of space forms, Project MAC, Design Division, Department of Mechanical Engineering, MIT, Cambridge, MA, 1964, revised to MAC-TR-41, 1967.

[56] M. Crouzeix, P.A. Raviart, Conforming and non-conforming finite element methods for solving the stationary Stokes equations, RAIRO R-3 (1973), 77–104.

[57] C.W. Crowley, P.P. Silvester, H. Hurwitz, Jr., Covariant projection elements for 3D vector field problems, IEEE Trans. Magn. 24 (1998), 397–400.

[58] P.J. Davis, *Interpolation and Approximation*, Blaisdell, London, 1963.

[59] L. Demkowicz, Edge finite elements of variable order for Maxwell's equations, in: Proc. International Workshop on Scientific Computing in Electrical Engineering (SCEE 2000), Warnemünde, August 20-23, 2000.

[60] L. Demkowicz, 2D *hp*-adaptive finite element package (2Dhp90), Version 2.1, TICAM Report 02-06, The University of Texas at Austin, 2002.

[61] L. Demkowicz, I. Babuška, Optimal *p*-interpolation error estimates for edge finite elements of variable order in 2D, TICAM Report 01-11, The University of Texas at Austin, April 2001.

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

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

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

[65] L. Demkowicz, L. Vardapetyan, Modelling of electromagnetic absorption/scattering problems using *hp*-adaptive finite elements, Comput. Methods Appl. Mech. Engrg. 152 (1998), 103–124.

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

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

[68] L. Demkowicz et al., De Rham diagram in 3D. Quasi-optimal *p*-interpolation estimates, manuscript of TICAM Report, The University of Texas at Austin, 2002.

[69] J.W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, PA, 1997.

[70] P. Deuflhard, U. Nowak, Extrapolation integrators for quasilinear implicit ODEs, In: *Large Scale Scientific Computing* (P. Deuflhard, B. Engquist, eds.), Progress in Scientific Computing 7, Birkhäuser, Basel, 1987, 37–50.

[71] F. Dubois, Discrete vector potential representation of a divergence-free vector-field in three-dimensional domains. Numerical analysis of a model problem, SIAM J. Numer. Anal. 27 (1990), 1103–1141.

[72] I.S. Duff, A.M. Erisman, J.K. Reid, *Direct Methods for Sparse Matrices*, Clarendon Press, Oxford, 1997.

[73] D.A. Dunavant, Economical symmetrical quadrature rules for complete polynomials over a square domain, Internat. J. Numer. Methods Engrg. 21 (1985), 1777–1784.

[74] D.A. Dunavant, High degree efficient symmetrical Gaussian quadrature rules for the triangle, Internat. J. Numer. Methods Engrg. 21 (1985), 1129–1148.

[75] D.A. Dunavant, Efficient symmetrical cubature rules for complete polynomials of high degree over the unit cube, Internat. J. Numer. Methods Engrg. 23 (1986), 397–407.

[76] H. Engels, *Numerical Quadrature and Cubature*, Academic Press, London, 1980.

[77] I. Ergatoudis, B. Irons, O. Zienkiewicz, Curved isoparametric "quadrilateral" elements for finite element analysis, Internat. J. Solids Structures 4 (1968), 31–42.

[78] K. Eriksson, C. Johnson, Adaptive finite element methods for linear elliptic problems, Math. Comp. 50 (1988), 361–383.

[79] K. Eriksson, C. Johnson, Adaptive finite element methods for parabolic problems I: A linear model problem, SIAM J. Numer. Anal. 28 (1991), 43–77.

[80] K. Eriksson, C. Johnson, Adaptive finite element methods for parabolic problems IV: Nonlinear problems, SIAM J. Numer. Anal. 32 (1995), 1729–1749.

[81] G. Evans, *Practical Numerical Integration*, J. Wiley & Sons, Chichester, 1993.

[82] M. Fiedler, *Special Matrices and Their Applications in Numerical Mathematics*, Nijhoff, Dordrecht, 1986.

[83] A.R. Forrest, Curves and surfaces for computer aided design, Ph.D. Thesis, The University Mathematical Laboratory, Cambridge University, Cambridge, England, 1968.

[84] A.R. Forrest, Coons surfaces and multivariable curve interpolation, J. Assoc. Comput. Mach. (1971).

[85] R.M. Furzeland, J.G. Verwer, P.A. Zegeling, A numerical study of three moving grid methods for one-dimensional partial differential equations which are based on the method of lines, J. Comput. Phys. 89 (1990), 349–388.

[86] H. Gajewski, K. Gröger, K. Zacharias, *Nichtlineare Operatorgleichungen und Operatordifferentialgleichungen*, Akademie-Verlag, Berlin, 1994.

[87] W.C. Gear, The simultaneous numerical solution of differential-algebraic equations, IEEE Trans. Circuit Theory CT-18 (1971), 89–95.

[88] A. George, J.W.H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[89] C. Geuzaine et al., Convergence of high-order curl-conforming finite elements, IEEE Trans. Magn. 35 (1999), 1442–1444.

[90] V. Girault, P.A. Raviart, *Finite Element Methods for Navier-Stokes Equations*, Springer-Verlag, Berlin–Heidelberg, 1986.

[91] G.H. Golub, J.M. Ortega, *Scientific Computing – An Introduction with Parallel Computing*, Academic Press, New York, 1993.

[92] G.H. Golub, C.F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, 1996.

[93] W.J. Gordon, Blending function methods of bivariate and multivariate interpolation and approximation, Research Report GMR-834, General Motors, Warren, MI, 1968.

[94] W.J. Gordon, Distributive lattices and the approximation of multivariate functions, In: *Approximation with Special Emphasis on Spline Functions* (I.J. Schoenberg, ed.), Academic Press, New York, 1969, 223 – 277.

[95] W.J. Gordon, Free-form surface interpolation through curve networks, Research Report GMR-921, General Motors, Warren, MI, 1969.

[96] W.J. Gordon, Spline-blended surface interpolation through curve networks, J. Math. Mech. 18 (1969), 931–952.

[97] W.J. Gordon, Blending function methods of bivariate and multivariate interpolation and approximation, SIAM J. Numer. Anal. 8 (1971), 158–177.

[98] W.J. Gordon, C.A. Hall, Geometric aspects of the finite element method: Construction of curvilinear coordinate systems and their application to mesh generation, Int. J. Numer. Methods Engrg. (1972).

[99] W.J. Gordon, C.A. Hall, Transfinite element methods: Blending functions interpolation over arbitrary element domains, Numer. Math. 21 (1973), 109–129.

[100] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, Frontiers in Applied Mathematics 17, SIAM, Philadelphia, PA, 1997.

[101] A. Grundmann, H.M. Moeller, Invariant integration formulas for the n-simplex by combinatorial methods, SIAM J. Numer. Anal. 15 (1978), 282–290.

[102] M.H. Gutknecht, Lanczos-type solvers for non-symmetric linear systems of equations, Technical Report, Swiss Center for Scientific Computing, ETH, Zürich, 1997.

[103] G. Haase, M. Kuhn, U. Langer, Parallel multigrid 3D Maxwell solvers, Parallel Comput. 27 (2001), 761–775.

[104] W. Hackbush, *Multigrid Methods and Applications*, Springer-Verlag, Berlin, 1985.

[105] L.A. Hageman, D.M. Young, *Applied Iterative Methods*, Academic Press, New York, 1981.

[106] V. Han, L. Wang, Nonconforming finite element analysis for a plate contact problem, SIAM J. Industr. Appl. Math. 40 (2002), 1683–1697.

[107] M. Hano, Finite element analysis of dielectric-loaded waveguides, IEEE Trans. Microwave Theory Tech. MTT-32 (1984), 1275–1279.

[108] M.R. Hestenes, E. Stiefel, Methods of conjugate gradients for solving linear algebraic systems, J. Res. Nat. Bur. Standards 49 (1952), 409–435.

[109] J.S. Hesthaven, From electrostatics to almost optimal nodal sets for polynomial interpolation in a simplex, SIAM J. Numer. Anal. 35 (1998), 655–676.

[110] J.S. Hesthaven, D. Gottlieb, Stable spectral methods for conservation laws on triangles with unstructured grids, Comput. Methods Appl. Mech. Engrg. 175 (1999), 361–381.

[111] J.S. Hesthaven, C.H. Teng, Stable spectral methods on tetrahedral elements, SIAM J. Sci. Comput. 21 (2000), 2352–2380.

[112] J.S. Hesthaven, T. Warburton, High-order nodal methods on unstructured grids. I. Time-domain solution of Maxwell's equations, J. Comput. Phys. 181 (2002), 186–221.

[113] A.C. Hindmarsh, LSODE and LSODI, two new initial value ordinary differential equation solvers, ACM SIGNUM Newslett. 15 (1980), 10–11.

[114] A.C. Hindmarsh, ODEPACK. A systematized collection of ODE solvers, In: *Scientific Computing* (R.S. Stapleman et al., eds.), IMACS Transactions on Scientific Computation 1, North-Holland, Amsterdam, 1983, 55–64.

[115] R. Hiptmair, Canonical construction of finite elements, Math. Comput. 68 (1999), 1325–1346.

[116] R. Hiptmair, Higher-order Whitney forms, Technical Report 156, Sonderforschungsbereich 382, University of Tübingen, 2000.

[117] P. Houston, E. Süli, *hp*-adaptive discontinuous Galerkin finite element methods for first order hyperbolic problems, SIAM J. Numer. Anal., to appear.

[118] D. Ingerman, V. Druskin, L. Knizhnerman, Optimal finite difference grids and rational approximations of the square root. I. Elliptic Problems, Commun. Pure Appl. Math. 8 (2000), 1039–1066.

[119] J. Jin, *The Finite Element Method in Electromagnetics*, John Wiley & Sons, New York, 1993.

[120] T. Kailath, A.H. Sayed (eds.), *Fast Reliable Algorithms for Matrices with Structure*, SIAM, Philadelphia, PA, 1999.

[121] O.A. Karakashian, W.N. Jureidini, A nonconforming finite element method for the stationary Navier-Stokes equations, SIAM J. Numer. Anal. 35 (1998), 93–120.

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

[123] P. Keast, Moderate-degree tetrahedral quadrature formulas, Comput. Methods Appl. Mech. Engrg. 55 (1986), 339–348.

[124] P. Klouček, B. Li, M. Luskin, Analysis of a class of non-conforming finite elements for crystalline microstructures, Math. Comp. 65 (1996), 1111–1135.

[125] P. Klouček, F. Toffoletto, The three-dimensional non-conforming finite element solution of the Chapman-Ferraro problem, J. Comput. Phys. 150 (1999), 549–560.

[126] Z. Kopal, *Numerical Analysis*, John Wiley & Sons, New York, 1961.

[127] M. Křížek, P. Neittaanmäki, *Finite Element Approximations of Variational Problems and Applications*, Pitman Monographs and Surveys in Pure and Applied Mathematics 50, Longman Scientific & Technical, Harlow, 1990.

[128] A. Kufner, O. John, S. Fučík, *Function Spaces*, Academia, Prague, 1977.

[129] J. Kurzweil, *Ordinary Differential Equations*, SNTL, Prague, 1978.

[130] J.D. Lambert, *Computational Methods in Ordinary Differential Equations*, J. Wiley & Sons, London–New York, 1973.

[131] J.L. Lions, *Quelques Méthodes de Résolution des Problèmes aux Limites non Linéaires*, Dunod, Paris, 1969.

[132] J.N. Lyness, R. Cools, A survey of numerical cubature over triangles, In: *Mathematics of Computation 1943–1993: A Half-Century of Computational Mathematics* (W. Gautschi, ed.), Proc. Symposia in Applied Mathematics 48, American Mathematical Society, Providence, RI, 1994, 127–150.

[133] J.N. Lyness, D. Jespersen, Moderate degree symmetric quadrature rules for the triangle, J. Inst. Math. Appl. 15 (1975), 15–32.

[134] S.F. McCormick, *Multilevel Adaptive Methods for Partial Differential Equations*, Frontiers in Applied Mathematics 6, SIAM, Philadelphia, PA, 1989.

[135] J.M. Melenk, On the robust exponential convergence of *hp* finite element methods for problems with boundary layers, IMA J. Numer. Anal. 17 (1997), 577–601.

[136] J.M. Melenk, *hp-Finite Element Methods for Singular Perturbations*, Lecture Notes in Math. 1796, Springer-Verlag, Berlin, 2002.

[137] G. Meurant, *Computer Solution of Large Linear Systems*, Studies in Mathematics and Its Applications 28, North-Holland, Amsterdam, 1999.

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

[139] P. Monk, On the p- and hp-extension of Nédélec's $H(\mathrm{curl})$-conforming elements, J. Comput. Appl. Math. (1994), 117–137.

[140] P.K. Moore, A posteriori error estimation with finite element semi- and fully discrete methods for nonlinear parabolic equations in one space dimension, SIAM J. Numer. Anal. 31 (1994), 149–169.

[141] G. Mur, Edge elements, their advantages and their disadvantages, IEEE Trans. Magn. 30 (1994), 3552–3557.

[142] J.C. Nédélec, Mixed finite elements in $\mathbb{R}^3$, Numer. Math. 93 (1980), 315–341.

[143] J.C. Nédélec, A new family of mixed finite elements in $\mathbb{R}^3$, Numer. Math. 50 (1986), 57–81.

[144] R.H. Nochetto, G. Savaré, C. Verdi, A-posteriori error estimates for variable time-step discretizations of nonlinear evolution equations, Commun. Pure Appl. Math. 53 (2003), 525–589.

[145] J.T. Oden, Theory and implementation of high-order adaptive *hp* methods for the analysis of incompressible viscous flow, TICAM Report, The University of Texas at Austin, 1992.

[146] J.T. Oden, L.F. Demkowicz, *Applied Functional Analysis*, CRC Press, Boca Raton, 1996.

[147] J.T. Oden, A. Patra, Y. Feng, An *hp*-adaptive strategy, In: *Adaptive Multilevel and Hierarchic Computational Strategies* (A.K. Noor, ed.), ASME Publication 157, New York, 1992, 23–46.

[148] J.T. Oden, S. Prudhomme, Goal-oriented error estimation and adaptivity for the finite element method, Comput. Math. Appl. 41 (2001), 735–756.

[149] J.T. Oden, W. Wu, M. Ainsworth, Three step *hp* adaptive strategy for the incompressible Navier-Stokes equations, In: *Modeling, Mesh Generation and Adaptive Numerical Methods for Partial Differential Equations* (I. Babuška, J.E. Flaherty, eds.), IMA Vol. Math. Appl. 75, Springer, New York, 1995, 347–366.

[150] J.T. Oden et al., Towards a universal *hp* adaptive finite element strategy. Part 2. A posteriori error estimation, Comput. Methods Appl. Mech. Engrg. 77 (1989), 113–180.

[151] J.T. Oden et al., A-posteriori error analysis in finite elements. The element residual methods for symmetrizable problems with applications to compressible Euler and Navier-Stokes equations, Comput. Methods Appl. Mech. Engrg. 82 (1990), 183–203.

[152] M. Paraschivoiu, A.T. Patera, A hierarchical duality approach to bounds for the outputs of partial differential equations, Comput. Methods Appl. Mech. Engrg. 158 (1998), 389–407.

[153] J. Peraire, A.T. Patera, Bounds for linear-functional outputs of coercive partial differential equations: local indicators and adaptive refinement, In: *Advances in Adaptive Computational Methods in Mechanics* (P. Ladevéze, J.T. Oden, eds.), Elsevier, Amsterdam, 1998, 199–215.

[154] L.R. Petzold, A description of DDASSL: A differential/algebraic system solver, Sandia Report Sand 82-8637, Sandia National Laboratory, Livermore, CA, 1982.

[155] M. Picasso, Adaptive finite elements for a linear parabolic problem, Comput. Methods Appl. Mech. Engrg. 167 (1998), 223–237.

[156] S. Pissanetzky, *Sparse Matrix Technology*, Academic Press, London, 1984.

[157] M. Práger, J. Taufer, E. Vitásek, Overimplicit multistep methods, Appl. Math. 18 (1973), 399–421.

[158] W. H. Press et al., *Numerical Recipes. The Art of Numerical Computing*, Cambridge University Press, Cambridge, 1988.

[159] A. Quarteroni, A. Valli, *Numerical Approximation of Partial Differential Equations*, 2nd ed., Springer-Verlag, Berlin, 1997.

[160] W. Rachowicz, L. Demkowicz, A two-dimensional *hp*-adaptive finite package for electromagnetics, Comput. Methods Appl. Mech. Engrg. 93 (1999), 315–341.

[161] W. Rachowicz, L. Demkowicz, An *hp*-adaptive finite element method for electromagnetics. Part II. A 3D implementation, Internat. J. Numer. Methods Engrg. 53 (2002), 147–180.

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

[163] A. Ralston, P. Rabinowitz, *A First Course in Numerical Analysis*, 2nd ed., Dover Publications, Mineola, NY, 2001.

[164] R. Rannacher, F.T. Stuttmeier, A posteriori error control in finite element methods via duality techniques: Application to perfect plasticity, Comput. Mech. 21 (1998), 123–133.

[165] R. Rannacher, S. Turek, Simple non-conforming quadrilateral Stokes element, Numer. Methods Partial Differential Equations 8 (1992), 97–111.

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

[167] K. Rektorys, *The Method of Discretization in Time and Partial Differential Equations*, Reidel Publishing Company, Dordrecht, 1982.

[168] H.-G. Roos, M. Stynes, L. Tobiska, *Numerical Methods for Singularly Pertubed Differential Equations (Convection-Diffusion and Flow Problems)*, Springer-Verlag, Berlin, 1996.

[169] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Series in Computer Science, PWS Publishing Company, Boston, MA, 1996.

[170] H.E. Salzer, Tables for facilitating of Chebyshev's quadrature formulae, J. Math. Phys. 26 (1947), 191–194.

[171] A.A. Samarskii, E.S. Nikolaev, *Methods of Solution of Finite Difference Equations* (Russian), Nauka, Moscow, 1978.

[172] Ch. Schwab, *p- and hp-Finite Element Methods*, Clarendon Press, Oxford, 1998.

[173] K. Segeth, Grid adjustment based on a posteriori error estimates, Appl. Math. 38 (1993), 488–504.

[174] K. Segeth, A posteriori error estimates for parabolic differential systems solved by the finite element method of lines, Appl. Math. 39 (1994), 415–443.

[175] K. Segeth, A posteriori error estimation with the finite element method of lines for a nonlinear parabolic equation in one space dimension, Numer. Math. 83 (1999), 455–475.

[176] G. Sewell, *The Numerical Solution of Ordinary and Partial Differential Equations*, Academic Press, New York, 1988.

[177] J.R. Shewchuk, An introduction to the conjugate gradient method without the agonizing pain, www.cs.cmu.edu/∼quake-papers/painless-conjugate-gradient.ps.gz, 1994.

[178] P. Silvester, R. Ferrari, *Finite Elements for Electrical Engineers*, Cambridge University Press, Cambridge, 1996.

[179] V.J. Smirnov, *A Course of Higher Mathematics*, Vol. 3, Pergamon Press, Oxford, 1964.

[180] I.M. Smith, D.V. Griffiths, *Programming the Finite Element Method*, 3rd ed., John Wiley & Sons, Chichester, 1997.

[181] S.L. Sobolev, V.L. Vaskevich, *The Theory of Cubature Formulas*, Kluwer Academic Publishers, Dordrecht, 1996.

[182] P. Šolín, On the method of lines (Applications in the computational fluid dynamics and a-posteriori error estimation), PhD Thesis, Faculty of Mathematics and Physics, Charles University, Prague, 1999.

[183] P. Šolín, Scalar and vector-valued shape functions of variable order, TICAM Report 02-36, The University of Texas at Austin, August 2002.

[184] P. Šolín, Scalar and vector-valued finite elements of variable order. Part 1: Orthogonal polynomials and H^1-conforming elements, Part 2: $\boldsymbol{H}$(curl)-conforming elements, Part 3: $\boldsymbol{H}$(div)- and L^2-conforming elements, Appl. Math., submitted.

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

[186] T. Strouboulis, J.T. Oden, A posteriori error estimation of finite element approximations in fluid mechanics, Comput. Methods Appl. Mech. Engrg. 78 (1990), 201–242.

[187] A. Stroud, *Approximate Calculation of Multiple Integrals*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

[188] A. Stroud, D. Secrest, *Gaussian Quadrature Formulas*, Prentice-Hall, Englewood Cliffs, NJ, 1966.

[189] D. Sun et al., Spurious modes in finite element methods, IEEE Trans. Antennas and Propagation 37 (1995), 12–24.

[190] P.N. Swarztrauber, R.A. Sweet, Algorithm 541. Efficient Fortran subprograms for the solution of separable elliptic partial differential equations, ACM Trans. Math. Software 5 (1979), 352–371.

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

[192] G. Szegö, *Orthogonal Polynomials*, AMS Colloquium Publications 23, Providence, RI, 1939.

[193] R.P. Tewarson, *Sparse Matrices*, Academic Press, New York, 1973.

[194] V. Thomée, *Galerkin Finite Element Methods for Parabolic Problems*, Springer-Verlag, Berlin, 1997.

[195] R. Todd, The condition of certain matrices, Quart. J. Mech. Appl. Math. 2 (1949), 469–472.

[196] L.N. Trefethen, D. Bau, *Numerical Linear Algebra*, SIAM, Philadelphia, PA, 1997.

[197] C.W. Ueberhuber, *Numerical Computation*, Springer, Berlin, 1997.

[198] A. Vande Wouwer, P. Saucez, W.E. Schiesser (eds.), *Adaptive Method of Lines*, Chapman & Hall/CRC, Boca Raton, FL, 2001.

[199] R.S. Varga, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1962.

[200] R. Verfürth, *A Review of A Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques*, Wiley–Teubner, Chichester, 1996.

[201] R. Verfürth, A posteriori error estimates for nonlinear problems: $L^r(0,T;W^{1,\rho}(\Omega))$-error estimates for finite element discretizations of parabolic equations, Numer. Methods Partial Differential Equations 14 (1998), 487–518.

[202] J. Wang, N. Ida, Curvilinear and higher-order edge elements in electromagnetic field computation, IEEE Trans. Magn. 29 (1993), 1491–1494.

[203] J. Webb, Hierarchical vector based functions of arbitrary order for triangular and tetrahedral finite elements, IEEE Trans. Antennas and Propagation 47 (1999), 1244–1253.

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

[205] J.H. Wilkinson, *Rounding Errors in Algebraic Processes*, HMSO, London, 1963.

[206] J.H. Wilkinson, C. Reinsch, *Linear Algebra, Handbook for Automatic Computation 2*, Springer-Verlag, Berlin, 1971.

[207] D.M. Young, Iterative methods for solving partial difference equations of elliptic type, Trans. Amer. Math. Soc. 76 (1954), 92–111.

[208] J.Z. Zhu, O.C. Zienkiewicz, Adaptive techniques in the finite element method, Commun. Appl. Numer. Methods 4 (1988), 197–204.

[209] O. Zienkiewicz, *The Finite Element Method*, McGraw-Hill, London, 1971.

[210] O. Zienkiewicz et al., Isoparametric and associate finite element families for two- and three-dimensional analysis, In: *Finite Element Methods in Stress Analysis*, TAPIR, Technical University of Norway, Trondheim, 1970.

[211] D. Zwillinger, *Handbook of Integration*, Jones and Bartlett, Boston, 1992.