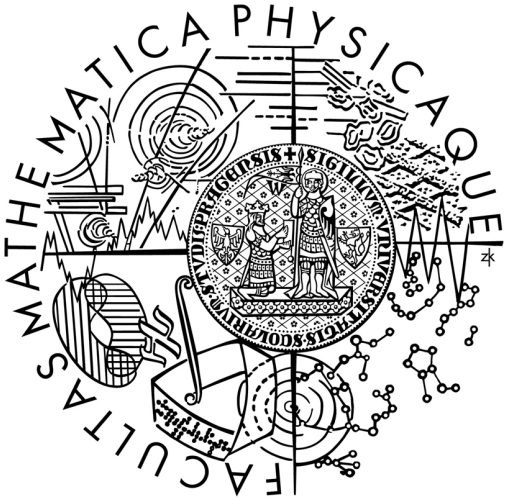
Charles University in Prague
Faculty of Mathematics and Physics

DOCTORAL THESIS




Lenka Dubcová

# *hp*-FEM FOR COUPLED PROBLEMS IN FLUID DYNAMICS

Department of Numerical Mathematics

Supervisor: *Prof. RNDr. Miloslav Feistauer, DrSc., Dr.h.c.*
Consultants: *RNDr. Pavel Šolín, PhD., RNDr. Veronika Sobotíková, CSc.*

Study program: *M6 - Scientific and technical computing*

2010

## Acknowledgments

I would like to thank here all people who helped me and supported me during my studies. First, I want to express my thanks to my advisor Prof. Miloslav Feistauer, who aroused my interest in numerical mathematics many years ago. I am obliged to him for his valuable advice and comments during consultations and for his support and help over the past four years.

My deep gratitude goes also to Prof. Ivo Doležel, who supported me during my stay at the Institute of Thermomechanics of the Academy of Sciences in Prague. He supplied me with many interesting problems and helped me with engineering aspects of the thesis. A wonderful person, he was always ready to spend time with me on my problems despite his own full schedule.

Next, I would like to thank Dr. Pavel Šolín for his leadership, inspiration and support during my studies. He gave me the opportunity to study and work at University of Texas at El Paso for two years and to do research at University of Nevada at Reno as a short-term scholar. I am grateful to be a part of his *hp*-FEM group and to work on so many interesting research topics he provided us with. Thanks to him I also met many interesting people in the finite element community all over the world (Glen Hansen, Dmitri Kuzmin or Granville Sewell), whom I thank as well for many valuable remarks.

I would like to thank my consultant Dr. Veronika Sobotíková for her detailed proof-reading of my thesis which improved it a lot. I am also thankful to many of my colleagues working in our team, Jakub Červený, Pavel Kůs and David Andrš, without whom the Hermes2d code would never have been written in today's form and without whom many of the results in this thesis would not be possible.

Finally, I wish to express my deepest thanks to my family for their support during my studies and mainly to Jakub Červený for his encouragement, valuable advice, programming guidance, English improvements, patience and many more.

This work was supported by grant No. 12810 of the Charles University in Prague and by grant GA AVČR IAA100760702.

I declare that this PhD. thesis is my own work and that I cited all the used references. The document can be freely reproduced for educational purposes.

Prague, 25 June 2010 Lenka Dubcová

## Abstract

The thesis is concerned with the solution of multiphysics problems described by partial differential equations using higher-order finite element method ($hp$-FEM). Basics of $hp$-FEM are described, together with some practical details and challenges. The $hp$-adaptive strategy, based on the reference solution and meshes with arbitrary level hanging nodes, is discussed. The thesis is mainly concerned with the extension of this strategy to monolithical solution of coupled multiphysics problems, where each physical field exhibits different qualitative behavior. In such problems, each physical field is discretized on an individual mesh automatically obtained by the adaptive algorithm to suit the best the corresponding solution component. Moreover, the meshes can change in time, following the needs of the solution components. All described methods and technologies are demonstrated on several examples throughout the thesis, where comparisons with traditionally used approaches are shown.

## Abstrakt

Disertační práce se zabývá řešením multifyzikálních problemů popsaných parciálními diferenciálními rovnicemi metodou konečných prvků vyšších řádů ($hp$-FEM). Základy této metody jsou popsány společně s praktickými detaily a problémy. Dále je popsána nová $hp$-adaptivní strategie založená na tzv. referenčním řešení a sítích s libovolným stupněm visících uzlů. Práce se především zabývá rozšířením této metody pro monolitické řešení multifyzikálních problémů, kde každá fyzikální složka vykazuje jiné kvalitativní chování a je tedy diskretizována na vlastní adaptivně získané síti vyhovující chování příslušné složky řešení. Tyto sítě se navíc mohou měnit v čase podle potřeb jednotlivých složek řešení. Všechny popsané metody jsou v práci demonstrovány na několika příkladech společně se srovnáním s tradičně používanými metodami.

# Contents

Chapter 1

# Introduction

Analytical solution of many physical and engineering processes described by partial differential equations (PDEs) is usually hard to obtain, and thus the only way to solve such problems is to apply numerical methods. In the last decades, computer and numerical modeling became an important part of most engineering and scientific fields. The most popular and the most widely used numerical method for the solution of partial differential equations is the finite element method (FEM). With the improvement of computer hardware the complexity of engineering simulations increases as well as the necessity for more accurate numerical approximations. This resulted in rapidly increasing popularity of adaptive higher-order finite element methods. Their main advantages lie in better approximation properties and the capability of exponential convergence [2, 3, 4, 10, 11, 38, 48]. Automatic $hp$-adaptivity, based on optimal combination of spatial refinements ($h$-adaptivity) and different polynomial degrees on elements ($p$-adaptivity), belongs to the most advanced topics in $hp$-FEM studied by many researchers [10, 11, 31, 48, 44, 41]. The reason why this method is not widely used among practitioners and in the engineering community is its algorithmic and implementation complexity.

Numerical modeling of multiphysics problems is a challenging task because several physical fields interact, usually in nonlinear manner. Examples of such nonlinearly coupled problems can be found in fluid dynamics, fluid-structure interaction, nuclear reactor analysis, magnetohydrodynamics and other areas. Furthermore, each component of the solution may have different spatial discretization requirements since the qualitative behavior of various physics components can differ significantly and may evolve in time [26]. As examples of such behavior we can mention singularities in the elastic displacement or in the electromagnetic field, boundary and internal layers in fluid flow or various sizes of vortices in the velocity field. Therefore, in order to capture individual behaviors of the solution components more efficiently, different physics should be discretized on individual meshes, and an adaptive method should be used to refine each mesh based on the needs of the physics it supports.

In the last decades, researchers solved multiphysics problems by dividing them into several distinct problems, one for each physical field, and solving them by existing monodisciplinary codes [19]. But since the components of the solution are usually strongly coupled, interaction between models (and codes) is necessary. Traditional coupling paradigms rely on solving the different physics in a loosely coupled way, a technique mathematically described as operator-splitting (OS). This is usually done in a simple black-box fashion, when results of one code serve as data for another code and vice-versa, utilizing various methods of data transfer between non-matching meshes. The major drawback of this approach is reduced accuracy and stability of the simulations [35], originating in the fact that coupling terms between the various physics components are handled inconsistently. Smaller time step is usually necessary to overcome problems arising from the fact that nonlinearities are not treated accurately, which results in increase of the computational time. Moreover, if different physics are discretized on independent meshes, OS methods are biased by the error caused by data transfers between non-matching meshes. Despite these drawbacks, operator-splitting methods are still one of the most popular methods for solving coupled multiphysics systems. This popularity dwells in the possibility of reusing existing monodisciplinary codes whose development took many years.

An alternative to solving multiphysics problems in decoupled form is to solve them monolithically, as one large nonlinear problem [35, 29, 21]. This involves an application of competent numerical methods for nonlinear systems of algebraic equations, such as standard Newton's method or Jacobian-free Newton's method together with efficient iterative linear solvers and suitable preconditioning [29, 30]. All nonlinearities in the coupled problem are then resolved up to a prescribed tolerance. To use the monolithic approach in practice one has to overcome several problems. One of them may be the necessity to use different types of finite elements in one computation, e.g., continuous elements for temperature and edge elements for electric field in microwave heating problems. If individual meshes for solution components are required, the problem of assembling stiffness matrix over geometrically different meshes appears and needs to be resolved. These problems usually discourage practitioners from using the monolithic approach.

Most multiphysics problems are time-dependent and their qualitative behavior may change significantly in time. In order to capture transient phenomena, sufficiently fine meshes have to be used. On the other hand the size of the problem should stay reasonably small since it has to be solved repeatedly on each time level. In engineering applications a very fine uniform mesh usually has to be employed for all time levels, which results in excessive computational times. Since the solution changes its qualitative behavior from one time level to another (e.g., moving fronts, vortices, etc.), most of the refinements in uniform meshes may not be necessary at a particular time level. This leads to a need for dynamical meshes for such problems [37, 47, 45].

The rest of the thesis is organized as follows: In Chapter 2, a brief overview of the higher-order finite element method is given together with some practical details and challenges. Chapter 3 describes an automatic $hp$-adaptive algorithm based on meshes with arbitrary-level hanging nodes and using the reference solution as an error estimator [11]. Solution of multiphysics problems is addressed in Chapter 4, where traditional methods such as operator-splitting and data transfer methods between non-matching meshes are described and compared to our approach, in which multiple meshes can be used for coupled physics [43, 14, 47]. Chapter 5 extends the $hp$-adaptivity and the multimesh technology to time-dependent problems, where automatically obtained $hp$-meshes dynamically change with the evolving solution with no need for data transfers between meshes. Finally, Chapter 6 is devoted to the numerical solution of an inductively heated flow of molten metals, where most of the described algorithms are employed.

CHAPTER 2

# Higher-order finite element method

This chapter is devoted to a brief introduction to the hierarchical higher-order finite element method (*hp*-FEM). We start with the definition of a model problem, which for simplicity is a linear second order elliptic partial differential equation. In next chapters, the concept of *hp*-FEM will be extended to nonlinear coupled problems involving several physical fields. We state the weak formulation of the model problem and describe its discretization by hierarchical *hp*-FEM in detail. The affine concept of finite elements is explained and the construction of global basis functions is described together with explicit formulas for shape functions on triangular and quadrilateral elements. Main theoretical and practical challenges and drawbacks of the method are mentioned. We demonstrate qualities and advantages of higher-order elements on an example with a known solution.

## 2.1 Model problem

Let us assume $\Omega \subset \mathbb{R}^2$ to be a bounded, simply-connected computational domain with Lipschitz continuous boundary $\partial\Omega = \overline{\Gamma}_D \cup \overline{\Gamma}_N$, where $\Gamma_D$ and $\Gamma_N$ are disjoint. Let $a_1, a_0, f, g_D$ and $g_N$ be given functions satisfying the regularity requirements.

$$a_1 \in C^1(\overline{\Omega}), \quad a_0 \in C(\overline{\Omega}), \quad f \in C(\overline{\Omega}), \quad g_D \in C(\Gamma_D), \quad g_N \in C(\Gamma_N).$$

These regularity requirements will be reduced after the partial differential equation (PDE) is formulated in a weak (variational) sense. We are concerned with the following problem: Find $u : \Omega \to \mathbb{R}$ such that

$$-\nabla(a_1 \nabla u) + a_0 u = f \qquad \text{in } \Omega, \tag{2.1}$$

$$u = g_D \qquad \text{on } \Gamma_D, \tag{2.2}$$

$$a_1 \frac{\partial u}{\partial \boldsymbol{n}} = g_N \qquad \text{on } \Gamma_N, \tag{2.3}$$

where $a_1$ and $a_0$ are space-dependent coefficients, $f$ is so-called load function, and $\boldsymbol{n}$ represents the unit normal vector to the boundary $\partial\Omega$. We say that a function $u$ is a classical solution of the above problem if $u \in C^2(\Omega) \cap C^1(\overline{\Omega})$ and $u$ satisfies (2.1) - (2.3).

For the purpose of the weak formulation of problem (2.1) - (2.3) we consider a function $u^* \in C^2(\Omega) \cap C(\overline{\Omega})$ such that $u^* = g_D$ on $\Gamma_D$ (the so-called Dirichlet lift of $g_D$). Notice that $u^*$ is not unique, but we will show later that the solution is invariant under its choice. Writing $u = U + u^*$, the problem (2.1) - (2.3) can be reformulated to the form: Find $U \in C^2(\Omega)$ such that

$$-\nabla(a_1 \nabla U) + a_0 U = f + \nabla(a_1 \nabla u^*) - a_0 u^* \qquad \text{in } \Omega, \tag{2.4}$$

$$U = 0 \qquad \text{on } \Gamma_D, \tag{2.5}$$

$$\frac{\partial(U + u^*)}{\partial \boldsymbol{n}} = g_N \qquad \text{on } \Gamma_N. \tag{2.6}$$

The weak formulation is then derived in a standard way. Equation (2.4) is multiplied by a smooth test function $v \in \{C^\infty(\Omega),\ v|_{\Gamma_D} = 0\}$ and integrated over the domain $\Omega$

$$\int_\Omega \left(-\nabla(a_1 \nabla U)\, v + a_0 U\, v\right)\, \mathrm{d}\boldsymbol{x} = \int_\Omega f\, v\, \mathrm{d}\boldsymbol{x} + \int_\Omega (\nabla(a_1 \nabla u^*)\ v - a_0\, u^*\, v)\, \mathrm{d}\boldsymbol{x}.$$

Then the Green's theorem is used to reduce the order of differentiation and boundary condition (2.6) is applied

$$\int_\Omega (a_1 \nabla U \cdot \nabla v + a_0\, U\, v)\, \mathrm{d}\boldsymbol{x} = \int_\Omega f\, v\, \mathrm{d}\boldsymbol{x} - \int_\Omega (a_1 \nabla u^* \cdot \nabla v - a_0\, u^*\, v)\, \mathrm{d}\boldsymbol{x} + \int_{\Gamma_N} g_N\, v\, \mathrm{d}S.$$

The identity above was derived under very strong regularity assumptions, but notice that all integrals are still well defined if these assumptions are weakened to

$$U,\ v \in V = \{\varphi \in H^1(\Omega),\ \varphi|_{\Gamma_D} = 0\}, \qquad f \in L^2(\Omega), \tag{2.7}$$

where $H^1(\Omega)$ is a Sobolev space. Similarly the regularity assumptions for the coefficients $a_1$, $a_0$ and $g_N$ can be reduced to $a_1, a_0 \in L^\infty(\Omega)$ and $g_N \in L^2(\Gamma_N)$.

Let us define a bilinear form $a(\cdot,\cdot) : V \times V \to \mathbb{R}$ and a linear form $l(\cdot) \in V'$ by

$$a(U, v) = \int_\Omega (a_1 \nabla U \cdot \nabla v + a_0\, U\, v)\ \mathrm{d}\boldsymbol{x}, \tag{2.8}$$

$$l(v) = \int_\Omega f\, v\, \mathrm{d}\boldsymbol{x} + \int_{\Gamma_N} g_N\, v\, \mathrm{d}S. \tag{2.9}$$

Then the weak formulation of problem (2.1) - (2.3) reads:

Find $u \in H^1(\Omega)$ such that

$$u = U + u^*, \quad \text{where } U \in V,\ u^* \in H^1(\Omega),\ u^* = g_D \text{ on } \Gamma_D,$$
$$a(U, v) = l(v) - a(u^*, v) \quad \text{for all } v \in V. \tag{2.10}$$

Assume that $a_1(x) \geq C_{min} > 0$ and $a_0(x) \geq 0$ a.e. in $\Omega$, then it can be easily shown (see [39]) that the bilinear form $a(\cdot, \cdot)$ is $V$-elliptic and bounded and that the linear form $l(\cdot)$ is bounded. Hence, these forms satisfy assumptions of the Lax-Milgram lemma and the solution $U$ exists and is unique. It remains to prove that also function $u = U + u^*$ is unique, in other words show that it is independent of the choice of the Dirichlet lift $u^*$. Suppose that there are two weak solutions $u_1 = U_1 + u_1^*$ and $u_2 = U_2 + u_2^*$, then

$$\int_\Omega a_1 \nabla(u_1 - u_2) \cdot \nabla v + a_0 (u_1 - u_2)\, v \, \mathrm{d}\boldsymbol{x} = 0,$$

for all $v \in V$. Since $u_1 - u_2 \in V$, we can take $v = u_1 - u_2$ and thus,

$$\int_\Omega a_1 (\nabla(u_1 - u_2))^2 + a_0 (u_1 - u_2)^2 \mathrm{d}\boldsymbol{x} = 0,$$

which implies $||u_1 - u_2||_e^2 = 0$, hence, $u_1 = u_2$ almost everywhere in $\Omega$.

## 2.2 $hp$-FEM discretization

We assume that the bounded domain $\Omega$ with a Lipschitz-continuous boundary is approximated by a computational domain $\Omega_h$ whose boundary is piece-wise polynomial and is more convenient for meshing. Then we cover the domain $\Omega_h$ with a finite element mesh.

**Definition 2.1 (Finite element mesh)** *Finite element mesh $\mathcal{T}_{h,p} = \{K_1, K_2, \ldots, K_M\}$ over a domain $\Omega_h \subset \mathbb{R}^2$ is a geometrical division of $\Omega_h$ into a finite number of nonoverlapping open cells (elements) $K_i$ such that*

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

*Each element $K_i$, $1 \leq i \leq M$ is equipped with a polynomial degree $0 \leq p(K_i) = p_i$. By $\mathcal{E}$ we denote the set of all edges in the mesh $\mathcal{T}_{hp}$.*

**Remark 2.1** *In this thesis we consider the finite element mesh to contain triangles or quadrilaterals with straight or curved edges.*

**Definition 2.2 (Regular mesh)** *The mesh is called regular if for any two elements $K_i$ and $K_j$, $i \neq j$, just one of the following alternatives holds:*

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

**Remark 2.2** *By assuming that the mesh is regular we avoid the so-called hanging nodes, which simplifies the discretization procedure. Hanging node in 2D is a mesh vertex, lying in the interior of an mesh edge. Finite element meshes with hanging nodes will be discused in the Chapter 3. Examples of irregular meshes with hanging nodes are illustrated in Fig. 2.1.*

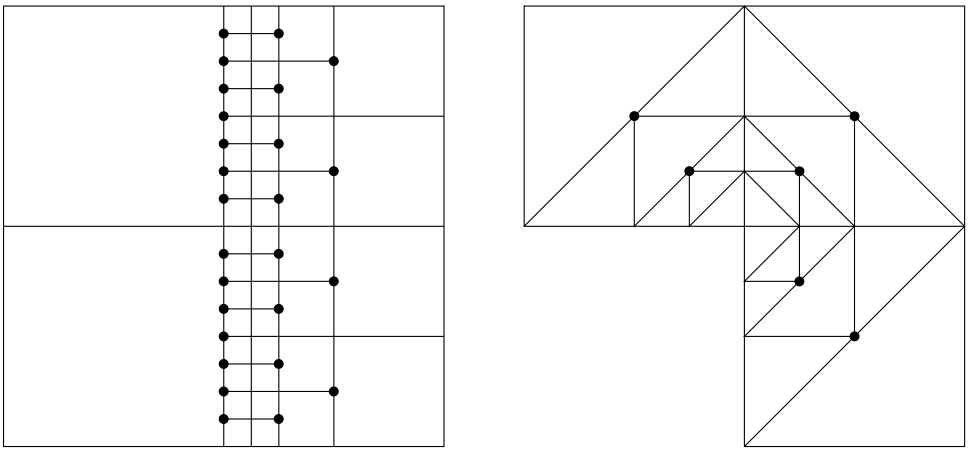


Figure 2.1: Finite element meshes with hanging nodes (black dots).

Consider an element $K_i$ equipped with polynomial degree $p_i$. Then we define local polynomial space $P^{p_i}(K_i)$ of dimension $N_i$ which consists of polynomials of degree at most $p_i$ on the element $K_i$. In order to define finite element space with nonuniform distribution of polynomial degrees in elements, we need to assign polynomial degrees also to edges. They obey so-called *minimum rule* – polynomial degree assigned to the edge $e \in \mathcal{E}$ is equal to the minimum of polynomial degrees in the interiors of adjacent elements

$$e = \overline{K_i} \cap \overline{K_j} \quad \rightarrow \quad p_e = \min(p_i, p_j). \tag{2.11}$$

The finite element space $V_{hp}$ (finite-dimensional approximation of the space $V$ from (2.7)) is then constructed as follows

$$V_{hp} = \{v \in V;\ v|_{K_i} \in P^{p_i}(K_i),\ K_i \in \mathcal{T}_{hp};\ v|_{e_j} \in P^{p_{e_j}}(e_j),\ e_j \in \mathcal{E}\}. \tag{2.12}$$

Then the variational formulation (2.10) can be approximated by the *discrete problem:* Find $u_{hp}$ such that

$$u_{hp} = U_{hp} + u^*_{hp}\,,$$
$$a(u_{hp},\, v_{hp}) = l(v_{hp}) \quad \text{for all} \;\; v_{hp} \in V_{hp}\,, \tag{2.13}$$

where $U_{hp} \in V_{hp}$ and $u^*_{hp}$ is a suitable piecewise polynomial approximation of the Dirichlet lift $u^*$.

Let $\{v_1, \ldots, v_N\}$, $N = \dim V_{hp}$, be a basis of the finite-dimensional space $V_{hp}$. As $\operatorname{span}\{v_1, \ldots, v_N\} = V_{hp}$, the solution $U_{hp}$ can be written as a linear combination of these basis functions with unknown coefficients

$$u_{hp} = u^*_{hp} + U_{hp} = u^*_{hp} + \sum_{i=1}^{N} y_i v_i.$$

Substituting the basis functions $v_1, \ldots, v_N$ for $v_{hp}$ and taking advantage of linearity of the bilinear form $a(\cdot,\cdot)$ we obtain the system of linear algebraic equations

$$\sum_{i=1}^{N} \underbrace{y_i}_{\boldsymbol{Y}_i}\, \underbrace{a(v_i,\, v_j)}_{\boldsymbol{S}_{ji}} = \underbrace{a(u^*_{hp},\, v_j) + l(v_j)}_{\boldsymbol{F}_j} \quad \text{for} \;\; j = 1, \ldots, N, \tag{2.14}$$

which can be written in matrix form as

$$\boldsymbol{S}\,\boldsymbol{Y} = \boldsymbol{F}.$$

Here $\boldsymbol{S}$ is called the stiffness matrix, $\boldsymbol{Y}$ is the vector of unknown coefficients and $\boldsymbol{F}$ the right-hand side (load) vector.

**Remark 2.3** *What remains to be defined is a suitable basis of the finite-dimensional space $V_{hp}$. While the Galerkin method assumes an arbitrary basis of the space $V_{hp}$, the finite element method prefers basis functions with supports as small as possible, so that as many of them as possible are disjoint. This results in sparse structure of the stiffness matrix $\boldsymbol{S}$.*

Let us define three types of global basis functions (see Fig. 2.2 for examples) that are in agreement with the Remark 2.3:

1. **Vertex functions** are associated with mesh vertices. Their value is one at one vertex and zero at all others. Hence, their support is formed by a patch of elements with one common vertex. The global basis of $V_{hp}$ contains vertex functions associated with all vertices that do not lie on the Dirichlet boundary $\Gamma_D$.

2. **Edge functions** are associated with mesh edges. They are nonzero on one edge and they vanish on all others. Their support consists of two adjacent elements. The global basis of $V_{hp}$ contains edge functions associated with edges that do no lie on the $\Gamma_D$. Number of edge functions on each edge corresponds to the polynomial degree of the edge $p_e$.

3. **Bubble functions** are associated with element interiors. Their support is only one element and they vanish on its boundary. Number of bubble functions on each element interion corresponds to the polynomial degree of the element $p_i$.

Such a description of global basis is very vague, we will illustrate the process of constructing the global basis of the space $V_{hp}$ in detail in the following sections.

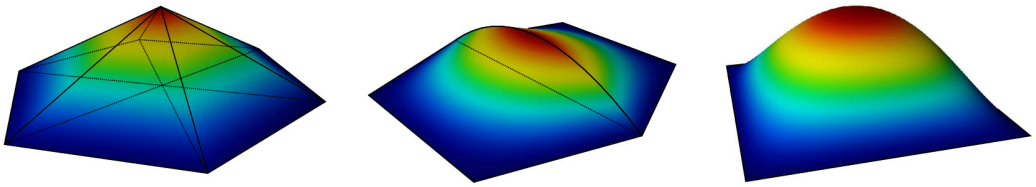


Figure 2.2: Example of a vertex, edge and bubble function.

### 2.2.1 Affine concept of FEM

For piecewise linear approximation, all integrals in weak formulation can be easily evaluated directly in the mesh using mesh vertices as quadrature points. For higher-order finite elements situation is more complex, many integration points are required per one element and storage of these points (coordinates, weights, function values) would be extremely inefficient. Therefore we follow the so called *affine concept*, in which a reference domain $\hat{K}$ is mapped onto mesh elements $K_i \in \mathcal{T}_{hp}$ by smooth bijective reference mappings

$$\boldsymbol{x}_{K_i} : \hat{K} \rightarrow K_i.$$

The mapping $\boldsymbol{x}_{K_i}$ is affine in case $K_i$ is either a triangle with straight edges or a parallelogram with straight edges. Main advantage of the *affine concept* is that discrete formulation (2.13) can be transformed from each mesh element $K_i$ to $\hat{K}$ by mappings $\boldsymbol{x}_{K_i}$, and all computational work (stiffness matrix assembly) can be done on the reference domain $\hat{K}$. Thus, numerical quadrature (coordinates of integration points and weights) can be defined only on the reference domain and values of functions and their derivatives at integration points can be stored only for the reference domain(s) resulting in distinctively smaller memory requirements.

As follows from the affine concept, everything is done locally on the reference domain(s) and transferred to the physical mesh elements via reference mappings. The global basis functions follow the same approach. Let us start with the definition of a finite element.

**Definition 2.3 (Finite element)** *A finite element is a triad $\mathcal{K} = (K, W, \Sigma)$, where*

- *$K$ is a domain in $\mathbb{R}^2$ - we restrict ourselves to triangles and quadrilaterals*
- *$W = \langle \phi_1, \phi_2, \ldots, \phi_N \rangle$ is a space of polynomials (**shape functions**) on $K$ of dimension $dim(W) = N$.*
- *$\Sigma = \{L_1, L_2, \ldots, L_N\}$ is a set of linear forms*

$$L_i : W \to \mathbb{R}, \quad i = 1, 2, \ldots, N.$$

*The elements of $\Sigma$ are called degrees of freedom (DOF).*

Following the affine concept, we define master finite element $\hat{\mathcal{K}} = (\hat{K}, \hat{W}, \hat{\Sigma})$. For computational purposes we consider two types of reference domains $\hat{K}$ - quadrilateral $K_q$ and triangular $K_t$.

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

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

Both reference domains with examples of reference maps are illustrated in Figs. 2.3 and 2.4.

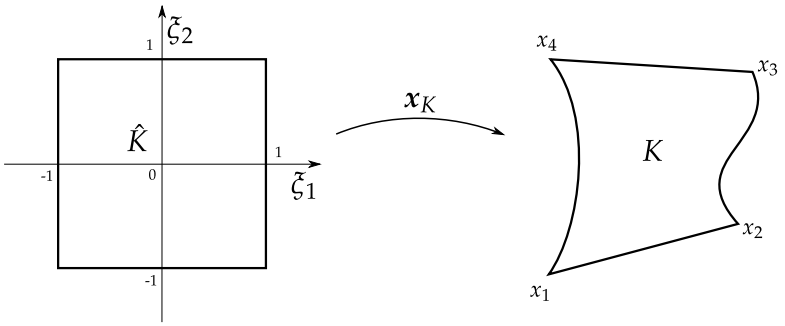



Figure 2.3: Examples of reference mappings from $\hat{K}$ onto $K$ (quadrilaterals).

Let the local order of approximation be $p^b$ for element interior and $p^{e_j}$ for edges, and let these orders satisfy the minimum rule $p^{e_j} \leq p^b$. To define the triad $\hat{\mathcal{K}} = (\hat{K}, \hat{W}, \hat{\Sigma})$, the reference domain $\hat{K}$ is equipped with the local polynomial space

$$\hat{W} = \{w \in P^{p^b}(\hat{K});\ w|_{e_j} \in P^{p^{e_j}}(e_j),\ j = 1, \ldots, m_{\hat{K}}\},$$

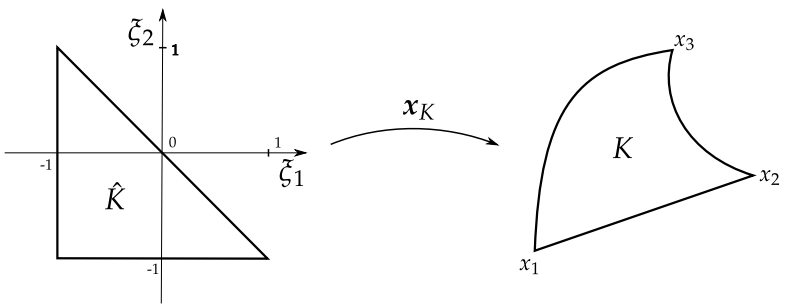



Figure 2.4: Examples of reference mappings from $\hat{K}$ onto $K$ (triangles).

where $m_{\hat{K}}$ denotes the number of edges of the reference element $\hat{K}$ ($m_{\hat{K}=3,4}$ for triangles and quadrilaterals, respectively). The set of degrees of freedom $\hat{\Sigma}$ is uniquely defined by the choice of a basis in $\hat{W}$. The finite element is said to be unisolvent if it satisfies *δ-property*

$$L_i(\phi_j) = \delta_{ij}, \quad \text{for all } 1 \leq i, j \leq N.$$

Here symbol $\delta_{ij}$ is standard Kronecker delta. In this thesis we consider hierarchic approach to finite elements since hierarchic basis allows (unlike nodal approach) nonuniform distribution of the polynomial degree in the mesh, which makes it suitable for *hp*-adaptivity. Consider a hierarchic basis $\mathcal{B}^p = \{\phi_1, \phi_2, \ldots, \phi_{N^p}\}$ of the local polynomial space $W$. By hierarchic we mean that

$$\mathcal{B}^p \subset \mathcal{B}^{p+1} \text{ for every } p.$$

Degrees of freedom are related to the basis functions, not to specific points in the mesh as in the nodal finite elements. Let $g \in P^{p_i}(K_i)$ be any polynomial and $\mathcal{B}^{p_i}(K_i) = \{\phi_1, \phi_2, \ldots, \phi_{N_i}\}$ be a basis of $P^{p_i}(K_i)$, then

$$g = \sum_{j=1}^{N_i} \beta_j \phi_j = \sum_{j=1}^{N_i} L_j(g)\, \phi_j,$$

where $\beta_i$ are real coefficients and $L_i(g) = \beta_i$ linear forms from $\Sigma$. Such a choice of degrees of freedom satisfies the $\delta$-property and yields an unisolvent finite element [48].

**Remark 2.4** *Since the space $H^1$ imposes global continuity of approximation, their values are constraint at vertices and on edges. Thus, the hierarchic local basis of space $\hat{W}$ have the same structure as the global basis. Shape functions are associated with vertex, edge and bubble nodes. Global basis of the space $V_{hp}$ is then constructed using these shape functions since all basis functions in the physical finite element mesh can be defined by specifying their restriction to each element by means of shape functions and reference maps.*

**Quadrilateral master element $K_q$**

In this paragraph we design hierarchic master element shape functions of arbitrary polynomial degree on quadrilateral reference domain $K_q$, depicted in Fig. 2.5

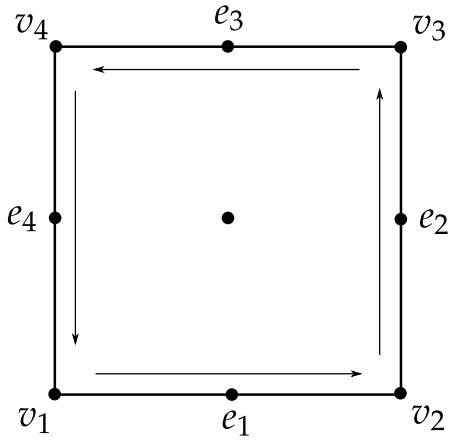



Figure 2.5: Reference quadrilateral element $K_q$ with 4 vertex nodes, 4 edge nodes and interior (bubble) node.

Optimal design of shape functions is an extremely difficult and important question. It influences the resulting stiffness matrix, its properties, and hence numerical solution of the resulting algebraic system. We describe the shape functions based on Lobatto polynomials, most commonly used shape functions in consequence of their orthonormality in $H_0^1$-seminorm.

To allow anisotropic polynomial order of approximation inside quadrilateral elements (horizontal and vertical), we consider two local directional polynomial degrees $p^{b,1}$ and $p^{b,2}$. Element edges are connected with polynomial degrees $p^{e_1}, p^{e_2}, p^{e_3}, p^{e_4}$ that originate in the physical mesh, where they satisfy the minimum rule (2.11)

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

Thus, the local polynomial space $W_q$ connected to $K_q$ is defined as follows

$$W_q = \{w \in Q_{p^{b,1},p^{b,1}};\ w|_{e_j} \in P_{p^{e_j}}(e_j),\ j = 1,\dots,4\}, \tag{2.15}$$

where

$$Q_{p,r} = \operatorname{span}\left\{\xi_1^i \xi_2^j;\ (\xi_1,\xi_2) \in K_q,\ i = 0,\dots,p,\ j = 0,\dots,r\right\}.$$

Before we state formulas for basis functions of the space $W_q$, let us define Legendre and Lobatto polynomials and mention some of their important properties.

**Definition 2.4 (Legendre polynomials)** *Legendre polynomials $L_n(x)$, $n = 0, 1, \ldots$ are eigenvectors of the Legendre operator,*

$$-\frac{d}{dx}\left[(1-x^2)\frac{dL_n}{dx}\right] = n(n+1)L_n, \quad x \in (-1,1).$$

*Legendre polynomials can be evaluated using the recursive formula*

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

Legendre polynomials form an orthogonal basis of the space $L_2(-1,1)$

$$\int_{-1}^{1} L_k(x)L_m(x)\, \mathrm{d}x = \begin{cases} \dfrac{2}{2k+1} & \text{for } k = m, \\ 0 & \text{otherwise.} \end{cases} \tag{2.16}$$

**Definition 2.5 (Lobatto functions)** *Let us define functions*

$$\begin{aligned} l_0(x) &= \frac{1-x}{2}, \\ l_1(x) &= \frac{x+1}{2}, \\ l_k(x) &= \frac{1}{||L_{k-1}||_2}\int_{-1}^{x} L_{k-1}(\xi)\, d\xi, \quad k = 2, 3, \ldots \end{aligned} \tag{2.17}$$

Here $||L_{k-1}||_2 = \sqrt{2/(2k-1)}$, which follows from (2.16). Obviously $l_k(-1) = 0$ and also $l_k(1) = 0$ since $L_{k-1}$, $k \geq 2$ are orthogonal to $L_0 \equiv 1$,

$$\int_{-1}^{1} L_{k-1}(x)\, \mathrm{d}x = \int_{-1}^{1} L_{k-1}(x)L_0(x)\, \mathrm{d}x = 0, \quad k = 2, 3, \ldots$$

Lobatto functions $l_0, l_1, \ldots, l_p$ form a complete basis of the space $P_p(-1,1)$.

Two dimensional shape functions on quadrilateral reference domain $(-1,1)^2$ are constructed as tensor products of the one dimensional Lobatto functions. They are grouped into 3 subsets according to which node they belong.

*Lobatto vertex functions* $\varphi_q^{v_1}, \ldots, \varphi_q^{v_4}$ assigned to vertices $v_1, \ldots, v_4$ are equal to one at $v_j$ and vanishes at all remaining vertices. They are chosen bilinear, defined via products

of Lobatto functions as follows:

$$\begin{aligned} \varphi_q^{v_1}(\xi_1,\xi_2) &= l_0(\xi_1)l_0(\xi_2), \\ \varphi_q^{v_2}(\xi_1,\xi_2) &= l_1(\xi_1)l_0(\xi_2), \\ \varphi_q^{v_3}(\xi_1,\xi_2) &= l_1(\xi_1)l_1(\xi_2), \\ \varphi_q^{v_4}(\xi_1,\xi_2) &= l_0(\xi_1)l_1(\xi_2). \end{aligned} \tag{2.18}$$

*Lobatto edge functions* $\varphi_{k,q}^{e_j}$, $k = 2,\ldots,p^{e_j}$, $j = 1,\ldots,4$ are associated with the corresponding edge $e_j$. Their trace on $e_j$ coincides with the Lobatto functions $l_k$ and their trace vanishes on all remaining edges.

$$\begin{aligned} \varphi_{k,q}^{e_1}(\xi_1,\xi_2) &= l_k(\xi_1)l_0(\xi_2), \qquad 2 \le k \le p^{e_1}, \\ \varphi_{k,q}^{e_2}(\xi_1,\xi_2) &= l_1(\xi_1)l_k(\xi_2), \qquad 2 \le k \le p^{e_1}, \\ \varphi_{k,q}^{e_3}(\xi_1,\xi_2) &= l_k(\xi_1)l_1(\xi_2), \qquad 2 \le k \le p^{e_1}, \\ \varphi_{k,q}^{e_4}(\xi_1,\xi_2) &= l_0(\xi_1)l_k(\xi_2), \qquad 2 \le k \le p^{e_1}. \end{aligned} \tag{2.19}$$

The hierarchic basis of the space $W_q$ is completed by *Lobatto bubble functions* $\varphi_{n_1,n_2,q}^{b}$ which vanish everywhere on the boundary of the reference domain $K_q$

$$\varphi_{n_1,n_2,q}^{b}(\xi_1,\xi_2) = l_{n_1}(\xi_1)l_{n_2}(\xi_2), \qquad 2 \le n_1 \le p^{b,1},\ 2 \le n_2 \le p^{b,2}. \tag{2.20}$$

The total number of hierarchic shape functions on quadrilateral master element is summarized in Tab. 2.1.

Table 2.1: Hierarchic shape functions on $K_q$.

| **Node type** | **Pol. degree** | **# of shape functions** | **# of nodes** |
|---|---|---|---|
| Vertex | always | 1 | 4 |
| Edge | $p^{e_j} \ge 2$ | $p^{e_j} - 1$ | 4 |
| Bubble | $p^{b,1}, p^{b,2} \ge 2$ | $(p^{b,1} - 1)(p^{b,2} - 1)$ | 1 |

**Triangular master element** $K_t$

In this paragraph we design hierarchic master element shape functions of arbitrary polynomial degree on triangular reference domain $K_t$, depicted in Fig. 2.6

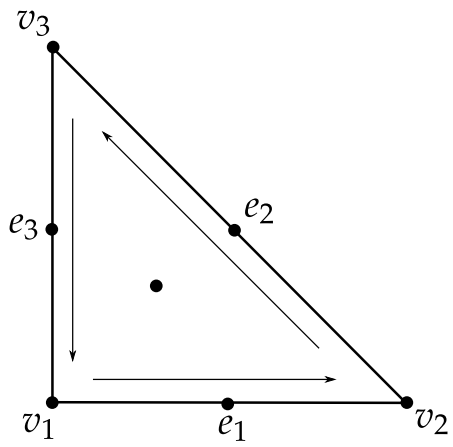



Figure 2.6: Reference triangular element $K_t$ with 3 vertex nodes, 3 edge nodes and interior (bubble) node.

The reference geometry $K_t = \{\boldsymbol{\xi} \in \mathbb{R}^2; -1 < \xi_1, \xi_2; \xi_1 + \xi_2 < 0\}$ is equipped with the affine (barycentric) coordinates

$$\lambda_1(\xi_1, \xi_2) = -\frac{\xi_1 + \xi_2}{2}, \qquad \lambda_2(\xi_1, \xi_2) = \frac{\xi_1 + 1}{2}, \qquad \lambda_3(\xi_1, \xi_2) = \frac{\xi_2 + 1}{2}. \tag{2.21}$$

For the definition of the higher-order shape functions on a triangle it is convenient to decompose the higher-order Lobatto functions $l_2, l_3, \ldots$ from Definition 2.5 into products of the form

$$l_k(x) = l_0(x) l_1(x) \phi_{k-2}(x), \quad 2 \leq k. \tag{2.22}$$

where the **kernel functions** $\phi_{k-2}$, $k = 2, 3, \ldots$ are polynomials of degree $k - 2$.

The edges $e_1, e_2, e_3$ are associated with local polynomial degrees $p^{e_1}, p^{e_2}, p^{e_3}$ and the element interior with one local order of approximation $p^b$. Again, these local polynomial degrees originate in the physical mesh, obeying the minimum rule. Hence, the local polynomial space $W_t$ on $K_t$ has the following form

$$W_t = \{w \in P_{p^b}(K_t);\ w|_{e_j} \in P_{p^{e_j}}(e_j),\ j = 1, 2, 3\}, \tag{2.23}$$

where

$$P_p(K_t) = \operatorname{span}\left\{\xi_1^i \xi_2^j;\ (\xi_1, \xi_2) \in K_t;\ i, j = 0, \ldots, p;\ i + j \leq p\right\}.$$

Hierarchic basis of the space $W_t$ again consist of vertex, edge and bubble functions:

*Vertex functions* $\varphi_t^{v_1}, \varphi_t^{v_2}, \varphi_t^{v_3}$ are assigned to vertices $v_1, v_2, v_3$. Each function is equal to one at corresponding vertex and vanishes at remaining two vertices. Vertex functions

are linear, defined by following formulas

$$\begin{aligned} \varphi_t^{v_1}(\xi_1,\xi_2) &= \lambda_1(\xi_1,\xi_2), \\ \varphi_t^{v_2}(\xi_1,\xi_2) &= \lambda_2(\xi_1,\xi_2), \\ \varphi_t^{v_3}(\xi_1,\xi_2) &= \lambda_3(\xi_1,\xi_2). \end{aligned} \tag{2.24}$$

*Edge functions* $\varphi_{k,t}^{e_j}$, $k = 2,\dots,p^{e_j}$, $j = 1,2,3$ coincide with one-dimensional Lobatto functions on corresponding edge and vanish on all remaining edges. They can be written using the kernel functions $\phi_{k-2}$, defined by (2.22), in the following form

$$\begin{aligned} \varphi_{k,t}^{e_1} &= \lambda_1\lambda_2\phi_{k-2}(\lambda_2-\lambda_1), \quad 2 \le k \le p^{e_1}, \\ \varphi_{k,t}^{e_2} &= \lambda_2\lambda_3\phi_{k-2}(\lambda_3-\lambda_2), \quad 2 \le k \le p^{e_2}, \\ \varphi_{k,t}^{e_3} &= \lambda_3\lambda_1\phi_{k-2}(\lambda_1-\lambda_3), \quad 2 \le k \le p^{e_3}. \end{aligned} \tag{2.25}$$

**Remark 2.5** *On both reference domains $K_q, K_t$ shape functions coincide on edges with Lobatto functions $l_0, l_1, l_2, \dots$, which allows us to combine triangular and quadrilateral elements in one hybrid mesh.*

*Bubble shape functions* $\varphi_{n_1,n_2,t}^{b}$ complete the basis of the space $W_t$. These functions vanish on the whole element boundary and their choice does not affect compatibility between quadrilateral and triangular meshes. They will be defined using affine coordinates and kernel functions as follows

$$\varphi_{n_1,n_2,t}^{b} = \lambda_{1,t}\lambda_{2,t}\lambda_{3,t}\phi_{n_1-1}(\lambda_{3,t}-\lambda_{2,t})\phi_{n_2-1}(\lambda_{2,t}-\lambda_{1,t}), \tag{2.26}$$

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

Total number of hierarchic shape functions on triangular master element is summarized in Tab. 2.2.

Table 2.2: Hierarchic shape functions on $K_t$.

| **Node type** | **Pol. degree** | **# of shape functions** | **# of nodes** |
|---|---|---|---|
| Vertex | always | 1 | 3 |
| Edge | $p^{e_j} \ge 2$ | $p^{e_j}-1$ | 3 |
| Bubble | $p^b \ge 3$ | $(p^b-1)(p^b-2)/2$ | 1 |

### 2.2.2 Construction of the reference mappings

The mapping from the reference domain $\hat{K}$ onto the physical linear element $K$ (with straight edges) can be simply expressed as a vertex interpolant

$$\boldsymbol{x}_K(\boldsymbol{\xi}) = \sum_{i=1}^{n} \boldsymbol{x}_i \varphi^{v_i}(\boldsymbol{\xi}), \tag{2.27}$$

where $n$ is the number of element vertices, $\boldsymbol{x}_i$ are coordinates of vertices in the physical mesh and $\varphi^{v_i}$ are master element vertex shape functions. The reference mapping for elements with curved edges could in general be nonpolynomial. In order to get feasible algorithms, the reference mapping is approximated using polynomial isoparametric approximation constructed with the aid of all types of shape functions - vertex, edge and bubble functions, using the same set of functions as for the approximation of the solution.

**Remark 2.6** *Using the shape functions for an approximation of nonpolynomial reference maps makes the mapping easy to store (a list of coefficients) and its values and derivatives as well as values and derivatives of its inverse can be easily accessed.*

The inversion of the reference mapping $\boldsymbol{x}_K$ is required only in case we need to locate corresponding point $\boldsymbol{\xi} \in \hat{K}$, given its image $\boldsymbol{x} \in K$. This is the case when we want to access solution at some given point $\boldsymbol{x}$ in the computational domain. In case the reference map is affine (i.e., for linear triangles and linear parallelograms), we have

$$\frac{D\boldsymbol{x}_K}{D\boldsymbol{\xi}}(v)(\boldsymbol{\xi} - v) = \boldsymbol{x} - \boldsymbol{x}_K(v),$$

where $v$ is, for example, one of the vertices of the reference domain and $\boldsymbol{x}_K(v)$ corresponding vertex in the physical mesh. Thus,

$$\boldsymbol{\xi} = v - \left(\frac{D\boldsymbol{x}_K}{D\boldsymbol{\xi}}\right)^{-1}(v)\ (\boldsymbol{x}_K(v) - \boldsymbol{x}). \tag{2.28}$$

In case the reference map is not affine, its inverse can be a nonpolynomial mapping and $\boldsymbol{\xi}$ must be sought iteratively by the Newton-Raphson technique

$$\boldsymbol{\xi}_{j+1} = \boldsymbol{\xi}_j - \left(\frac{D\boldsymbol{x}_K}{D\boldsymbol{\xi}}\right)^{-1}(\boldsymbol{\xi}_j)\ (\boldsymbol{x}_K(\boldsymbol{\xi}_j) - \boldsymbol{x}), \tag{2.29}$$

where initial guess $\boldsymbol{\xi}_0$ can be taken as any point in the reference domain $\hat{K}$.

### 2.2.3 Transformation of master element polynomial spaces

Following the affine concept described in Section 2.2.1, let us describe how the integrals in the discrete problem (2.13) are transformed from the mesh elements $K_m \in \mathcal{T}_{h,p}$ to the appropriate reference domain $\hat{K}$, where the integration takes place. Let us consider reference mapping $\boldsymbol{x}_{K_m}(\boldsymbol{\xi}) : \hat{K} \to K_m$ and a finite master element $\mathcal{K} = (\hat{K}, \hat{W}, \Sigma)$ from Def. 2.3. Then in the $H^1$-conforming case the mapping $\boldsymbol{\Phi}_{K_m}$ from the polynomial space $\hat{W}(\hat{K})$ to the polynomial space $W(K_m)$

$$\hat{W}(\hat{K}) \xrightarrow{\boldsymbol{\Phi}_{K_m}} W(K_m)$$

requires that the function value of any polynomial $\hat{w} \in \hat{W}(\hat{K})$ at each reference point $\boldsymbol{\xi} \in \hat{K}$ coincides with the value of the transformed function $w \in W(K_m)$ at its image $\boldsymbol{x} = \boldsymbol{x}_{K_m}(\boldsymbol{\xi}) \in K$. Thus,

$$w(\boldsymbol{x}) = (\boldsymbol{\Phi}_{K_m}(\hat{w}))\,(\boldsymbol{x}) = (\hat{w} \circ \boldsymbol{x}_{K_m}^{-1})(\boldsymbol{x}) = \hat{w}(\boldsymbol{x}_{K_m}^{-1}(\boldsymbol{x})), \tag{2.30}$$

or

$$\hat{w}(\boldsymbol{\xi}) = (w \circ \boldsymbol{x}_{K_m})(\boldsymbol{\xi}) = w(\boldsymbol{x}_{K_m}(\boldsymbol{\xi})). \tag{2.31}$$

Now, using the chain rule of differentiation we obtain

$$\begin{aligned}
\frac{\partial \hat{w}}{\partial \xi_1}(\boldsymbol{\xi}) &= \frac{\partial w}{\partial x_1}\,\frac{\partial x_{K_m,1}}{\partial \xi_1} + \frac{\partial w}{\partial x_2}\,\frac{\partial x_{K_m,2}}{\partial \xi_1}, \\
\frac{\partial \hat{w}}{\partial \xi_2}(\boldsymbol{\xi}) &= \frac{\partial w}{\partial x_1}\,\frac{\partial x_{K_m,1}}{\partial \xi_2} + \frac{\partial w}{\partial x_2}\,\frac{\partial x_{K_m,2}}{\partial \xi_2}.
\end{aligned}$$

Thus,

$$\begin{pmatrix} \dfrac{\partial \hat{w}}{\partial \xi_1} \\ \dfrac{\partial \hat{w}}{\partial \xi_2} \end{pmatrix} = \begin{pmatrix} \dfrac{\partial x_{K_m,1}}{\partial \xi_1} & \dfrac{\partial x_{K_m,2}}{\partial \xi_1} \\ \dfrac{\partial x_{K_m,1}}{\partial \xi_2} & \dfrac{\partial x_{K_m,2}}{\partial \xi_2} \end{pmatrix} \begin{pmatrix} \dfrac{\partial w}{\partial x_1} \\ \dfrac{\partial w}{\partial x_2} \end{pmatrix} = \left(\frac{\partial D\boldsymbol{x}_{K_m}}{\partial \boldsymbol{\xi}}\right)^T \begin{pmatrix} \dfrac{\partial w}{\partial x_1} \\ \dfrac{\partial w}{\partial x_2} \end{pmatrix},$$

where $\dfrac{\partial D\boldsymbol{x}_{K_m}}{\partial \boldsymbol{\xi}}$ represents the Jacobi matrix of the mapping $\boldsymbol{x}_{K_m}$. Hence, the gradient $\nabla w$ at an arbitrary point $\boldsymbol{x} \in K_m$ is transformed to the corresponding point $\boldsymbol{\xi} \in \hat{K}$ as follows

$$\nabla w(\boldsymbol{x}) = \left(\frac{\partial D\boldsymbol{x}_{K_m}}{\partial \boldsymbol{\xi}}\right)^{-T} \nabla \hat{w}(\boldsymbol{\xi}). \tag{2.32}$$

Applying formulas (2.31) and (2.32), the bilinear form $a(\cdot, \cdot)$ can be transformed in

the following way

$$\begin{aligned} a(u,\,v) &= \int_{\Omega_h} \Big( a_1(\boldsymbol{x}) \nabla u(\boldsymbol{x}) \cdot \nabla v(\boldsymbol{x}) + a_0(\boldsymbol{x})\, u(\boldsymbol{x})\, v(\boldsymbol{x}) \Big)\, \mathrm{d}\boldsymbol{x} \\ &= \sum_{K_i \in \mathcal{T}_{hp}} \int_{K_i} \Big( a_1 \nabla u(\boldsymbol{x}) \cdot \nabla v(\boldsymbol{x}) + a_0(\boldsymbol{x})\, u(\boldsymbol{x})\, v(\boldsymbol{x}) \Big)\, \mathrm{d}\boldsymbol{x} \\ &= \sum_{K_i \in \mathcal{T}_{hp}} \left\{ \int_{\hat{K}} J_{K_i} \left[ \hat{a}_1(\boldsymbol{\xi}) \left( \frac{\partial D \boldsymbol{x}_{K_i}}{\partial \boldsymbol{\xi}} \right)^{-T} \nabla \hat{u}(\boldsymbol{\xi}) \right] \cdot \left[ \left( \frac{\partial D \boldsymbol{x}_{K_i}}{\partial \boldsymbol{\xi}} \right)^{-T} \nabla \hat{v}(\boldsymbol{\xi}) \right] \mathrm{d}\boldsymbol{\xi} \right. \\ &\qquad \left. + \int_{\hat{K}} J_{K_i}\, \hat{a}_0(\boldsymbol{\xi})\, \hat{u}(\boldsymbol{\xi})\, \hat{v}(\boldsymbol{\xi})\, \mathrm{d}\boldsymbol{\xi} \right\}. \end{aligned} \tag{2.33}$$

Here $J_{K_i}$ denotes Jacobian of the mapping $\boldsymbol{x}_{K_i}$

$$J_{K_i} = \det \left( \frac{\partial D \boldsymbol{x}_{K_i}}{\partial \boldsymbol{\xi}} \right).$$

**Remark 2.7** *For triangular elements with straight edges the reference mapping $\boldsymbol{x}_{K_i}$ is affine, therefore the Jacobian $J_{K_i}$ is constant and can be taken out of integrals. For quadrilateral elements or elements with curved edges the Jacobi matrix and Jacobian are not constant and must be integrated using numerical quadrature. This increases the order of the integration rule needed to evaluate integrals in (2.33) accurately.*

**Remark 2.8** *In order to avoid repeated transformations of shape functions and thus to speed up evaluation of the stiffness matrix entries in the element-by-element assembly procedure, transformed values of all shape functions and their derivatives*

$$\nabla u(\boldsymbol{x}) = \left( \frac{\partial D \boldsymbol{x}_{K_i}}{\partial \boldsymbol{\xi}} \right)^{-T} \nabla \hat{u}(\boldsymbol{\xi})$$

*can be precalculated in advance for an active element at all integration points, and stored. During the stiffness matrix assembly, when all possible pairs of shape functions are integrated, these precomputed values are repeatedly used, refraining from multiple matrix-times-vector evaluations, thus, accelerating the process. Indeed, let us look at the integral*

$$\int_{K_i} \nabla u \cdot \nabla v \, d\boldsymbol{x} =$$

$$\int_{\hat{K}} J_{K_i} \left( \left[ \left( \frac{\partial D \boldsymbol{x}_{K_m}}{\partial \boldsymbol{\xi}} \right)^{-T}_{11} \frac{\partial \hat{u}}{\partial \xi_1} + \left( \frac{\partial D \boldsymbol{x}_{K_m}}{\partial \boldsymbol{\xi}} \right)^{-T}_{12} \frac{\partial \hat{u}}{\partial \xi_2} \right] \left[ \left( \frac{\partial D \boldsymbol{x}_{K_m}}{\partial \boldsymbol{\xi}} \right)^{-T}_{11} \frac{\partial \hat{v}}{\partial \xi_1} + \left( \frac{\partial D \boldsymbol{x}_{K_m}}{\partial \boldsymbol{\xi}} \right)^{-T}_{12} \frac{\partial \hat{v}}{\partial \xi_2} \right]$$

$$+\left[\left(\frac{\partial D\boldsymbol{x}_{K_m}}{\partial\boldsymbol{\xi}}\right)^{-T}_{21}\frac{\partial\hat{u}}{\partial\xi_1}+\left(\frac{\partial D\boldsymbol{x}_{K_m}}{\partial\boldsymbol{\xi}}\right)^{-T}_{22}\frac{\partial\hat{u}}{\partial\xi_2}\right]\left[\left(\frac{\partial D\boldsymbol{x}_{K_m}}{\partial\boldsymbol{\xi}}\right)^{-T}_{21}\frac{\partial\hat{v}}{\partial\xi_1}+\left(\frac{\partial D\boldsymbol{x}_{K_m}}{\partial\boldsymbol{\xi}}\right)^{-T}_{22}\frac{\partial\hat{v}}{\partial\xi_2}\right]\Bigg)\,d\boldsymbol{\xi}.$$

*It contains 10 multiplication operations (multiplication by Jacobian is hidden in quadrature weights), while its simplified form with precalculated transformed derivatives*

$$\int_{\hat{K}} J_{K_i}\left(\frac{\partial u}{\partial x}\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\frac{\partial v}{\partial y}\right)\,d\boldsymbol{\xi}$$

*contains just 2 multiplications. Precalculation of transformed values takes only $O(p^4)$ operations since there are $O(p^2)$ shape functions evaluated at $O(p^2)$ integration points, while evaluation of the integrand itself takes $O(p^6)$ ($O(p^4)$ pairs of shape functions at $O(p^2)$ integration points). Therefore, theoretically we can speed up the stiffness matrix assembly up to 5 times.*

### 2.2.4 Design of global basis functions

Global basis of the finite-dimensional space $V_{hp}$ is now constructed by gluing together master element shape functions (transformed to the physical element by mapping $\boldsymbol{\Phi}_K$). It is done in such a way, that resulting basis functions satisfy conformity requirements of the approximated space $V$ (space $H^1$ in our model problem).

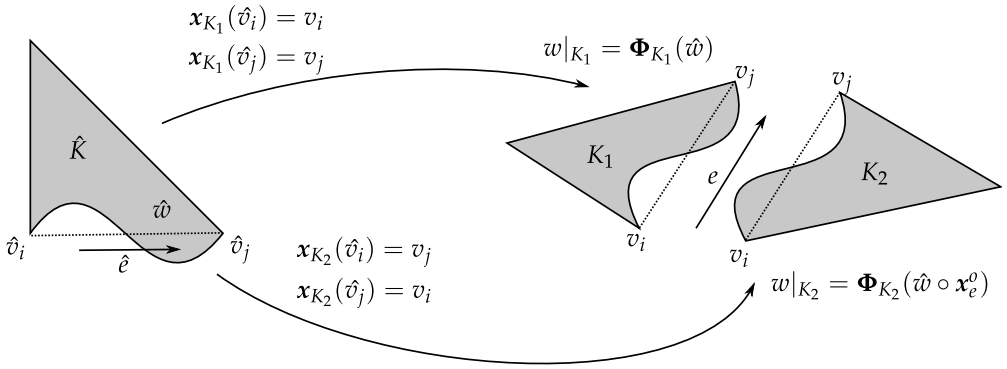



Figure 2.7: Example of an adjustment of edge shape function orientation.

In order to fulfill the conformity requirement of global continuity of basis functions, the orientation of physical mesh edges has to be taken into account. Edges of the reference element are oriented counterclockwise, while to each edge in the physical mesh an unique orientation is assigned (for example from vertex with lower index to the vertex with higher index). In case the orientation of the reference element edge

$\hat{e} = \hat{v}_i\hat{v}_j \subset \partial\hat{K}$ coincides with the orientation of the edge $e = v_iv_j \subset \partial K$,

$$\boldsymbol{x}_K(\hat{v}_i) = v_i, \quad \boldsymbol{x}_K(\hat{v}_j) = v_j,$$

then all master element shape functions associated with edge $\hat{e}$ stay unchanged. In case orientations differ, all master element edge functions have to be transformed by a suitable map $\boldsymbol{x}_e^o : \hat{K} \to \hat{K}$, which inverts the parametrization of the edge $\hat{e}$. In practice, we define edge shape functions on the reference domain with both orientations and pick out the correct one. For an illustrative example see Fig. 2.7, where on the element $K_1$ the master element edge function stays unchanged, while the edge function on the element $K_2$ has to be adjusted – its parametrization has to be inverted.

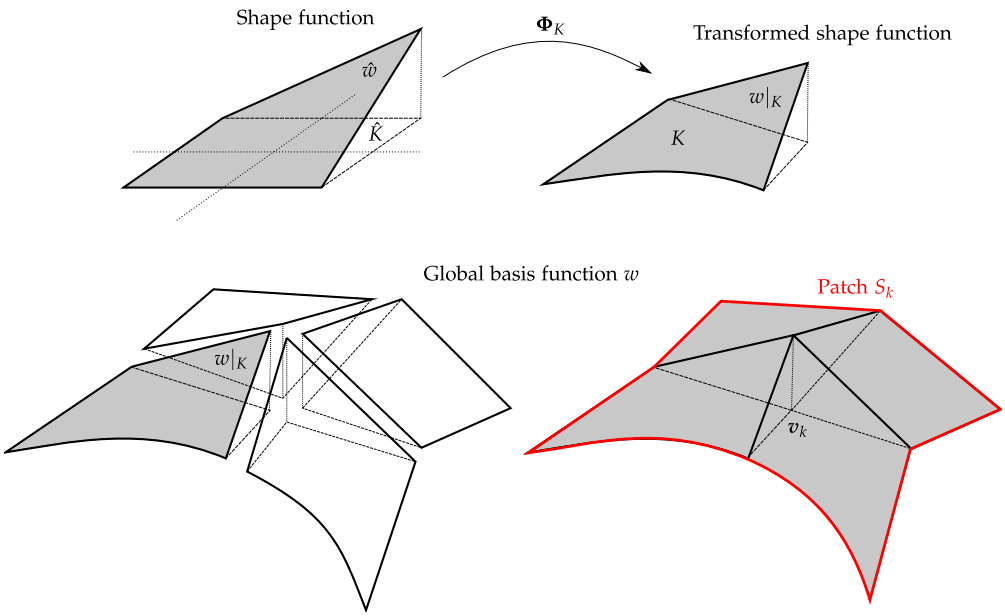



Figure 2.8: Construction of a global basis function from shape functions on the reference domain.

In the following we define the way globally conforming basis functions on element patches are constructed. Illustrative pictures for the vertex basis function are depicted in Fig. 2.8.

**Definition 2.6 (Vertex basis function)** *Let the space $V$ from the weak formulation be a subspace of $H^1(\Omega)$ and let $V_{hp}$ be a finite dimensional subspace of $V$. Vertex function $w \in V_{hp}$, associated with a mesh vertex $\boldsymbol{v}_k$, is a continuous function defined on $\Omega_h$ which equals to one at $\boldsymbol{v}_k$ and vanishes outside of the patch $S_k$ formed by all elements sharing the vertex $\boldsymbol{v}_k$. For each element $K_m \in S_k$*

$$w|_{K_m} = \boldsymbol{\Phi}_{K_m}(\hat{w}),$$

*where $\hat{w}$ is a master element vertex shape function associated with the vertex $v$, where $v = \boldsymbol{x}_{K_m}^{-1}(\boldsymbol{v}_k)$ is the corresponding vertex of the reference domain $\hat{K}$.*

**Definition 2.7 (Edge basis function)** *Let the space $V$ from the weak formulation be a subspace of $H^1(\Omega)$ and let $V_{hp}$ be a finite dimensional subspace of $V$. Edge function $w \in V_{hp}$, associated with a mesh edge $e_k$, is a continuous function defined on $\Omega_h$ which vanishes outside of the patch $S_k$ formed by all elements sharing the edge $e_k$. For each element $K_m \in S_k$*

$$w|_{K_m} = \boldsymbol{\Phi}_{K_m}(\hat{w} \circ \boldsymbol{x}_e^o),$$

*where $\hat{w}$ is a master element edge shape function associated with the edge $e$, where $e = \boldsymbol{x}_{K_m}^{-1}(e_k)$ is the corresponding edge of the reference domain $\hat{K}$, and $\boldsymbol{x}_e^o$ is the orientation adjustment.*

**Definition 2.8 (Bubble basis function)** *Let the space $V$ from the weak formulation be a subspace of $H^1(\Omega)$ and let $V_{hp}$ be a finite dimensional subspace of $V$. Bubble function $w \in V_{hp}$, associated with a mesh element $K_m$, is a continuous function defined on $\Omega_h$ which vanishes outside of the element $K_m$ such that*

$$w|_{K_m} = \boldsymbol{\Phi}_{K_m}(\hat{w}),$$

*where $\hat{w}$ is a master element bubble shape function.*

### 2.2.5 Assembling of the stiffness matrix

In linear finite elements all vertices can be simply numbered and stiffness matrix can be assembled in vertex-by-vertex fashion. Since degrees of freedom in higher-order finite elements are associated with vertices, edges and element interiors, vertices do not have that unique role in assembling of the stiffness matrix. Hence, it is better to assemble the linear system in an element-by-element fashion. Basic idea of element-by-element assembly respecting sparsity of the stiffness matrix is described in Alg. 1.

**Algorithm 1**: Element-by-element assembling procedure.

```
forall elements K_m ∈ T_hp do
    L = list of basis functions v_i whose supp(v_i) ∩ K_m ≠ ∅;
    for i ∈ L do
        for j ∈ L do
            S_ij = S_ij + a(v_i, v_j)|_{K_m};
```

Question of numbering of shape and basis functions arises. Local basis (shape functions) on the reference domain are divided into 3 types - vertex nodes, edge nodes and bubble nodes. Unique enumeration of shape functions is achieved by enumerating nodes (vertex, edge and interior) and then shape functions within these nodes. In

a similar way, following natural order, global basis is numbered. First vertices are visited and vertex functions are numbered, followed by edges and corresponding edge functions and ended by element interiors and bubble functions. One leaves out all vertices and edges lying on the Dirichlet boundary.

Now all shape functions on the reference element must be linked to an appropriate global basis function by so-called *connectivity arrays* $c(K, w)$, where $K \in \mathcal{T}_{hp}$ specifies an element and $w$ a master element shape function. Thus, the corresponding index to the global stiffness matrix for the shape function $w$ on the element $K$ is given by $c(K, w)$. In case the shape function contributes to the Dirichlet lift and is not related to any basis function we may indicate this by setting connectivity array to -1.

In practice, connectivity arrays are not stored for all elements, instead we assemble so-called local stiffness matrix $\boldsymbol{S}^K$ on element $K$ using all relevant shape functions $w_j$, $j = 1, \ldots, N_K$ and the reference map $\boldsymbol{x}_K$, where $N_K$ is number of degrees of freedom connected with the element $K$ (including shape functions representing the Dirichlet lift). Then, we construct *connectivity array* for element $K$ and use it to distribute values of the local stiffness matrix $\boldsymbol{S}^K$ to the global stiffness matrix $\boldsymbol{S}$ or to the global right-hand side vector $\boldsymbol{f}$ in case of Dirichlet DOFs (nodes lying on the Dirichlet boundary). See Alg. 2 for description of the assembling procedure in pseudocode. Note that we already incorporated ideas from Remark 2.8, hence, all shape functions are pretransformed at the beginning of the assembling process.

## 2.3 Practical and theoretical challenges of $hp$-FEM

Higher-order finite element method places high demands on both, mathematician and programmer. As any other numerical method $hp$-FEM has its advantages and drawbacks or challenges. As mentioned earlier, $hp$-FEM is theoretically capable of exponential convergence, higher-order accuracy can be achieved easily by combining low-order and higher-order elements. However, exponential convergence was proved by prof. Babuška in [23] for one dimensional problems only and its numerical attainment in practical problems is still a challenge. Moreover, optimal combinations of polynomial degrees (selection of the best refinement) rely on a posteriori error estimates of the finite element error, which are rarely known for $hp$-FEM, especially when solving real-life coupled problems. Several refinement strategies were employed using a priori knowledge about the solution (point singularities, boundary layers). In most of the approaches, an initial mesh is constructed (utilizing knowledge about solution behavior) and then uniform or adaptive $p$-refinements are made. In most practical problems behavior of the solution is not know a priori or it can even change in time. In the next Chapter we propose an automatic $hp$-adaptivity based on *reference solution* suitable for complicated engineering problems, where the nature of the solution may not be known.

**Algorithm 2**: Assembling procedure.

**forall** *elements* $K_m \in \mathcal{T}_{hp}$ **do**
  $L^{p_m}$ = list of shape function indices for polynomial degree $p_m$;
  **forall** $i \in L^{p_m}$ **do**
    precalculate transformed derivatives $\nabla v_i(\boldsymbol{x}) = \left(\frac{\partial D \boldsymbol{x}_{K_m}}{\partial \boldsymbol{\xi}}\right)^{-T} \nabla \hat{v}_i(\boldsymbol{\xi})$;

  // assemble local stiffness matrix and RHS vector
  **forall** $i \in L^{p_m}$ **do**
    **forall** $j \in L^{p_m}$ **do**
      $S_{ij}^{K_m} = S_{ij}^{K_m} + a(v_i, v_j)|_{K_m}$ ;
    $f_i^{K_m} = f_i^{K_m} + l(v_i)|_{K_m}$;

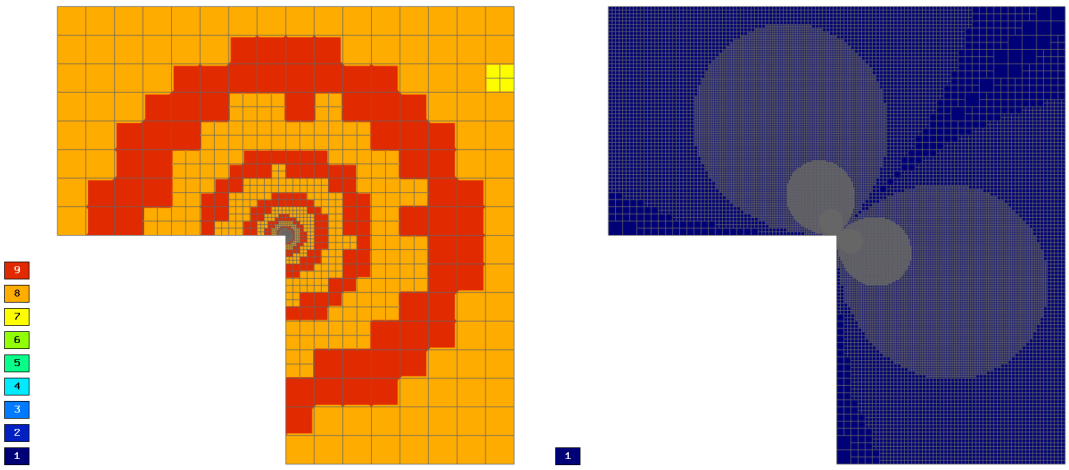
  // distribute entries of $S^{K_m}$ to the global stiffness matrix
  **forall** $i \in L^{p_m}$ **do**
    $r = c(K_m, i)$ ;  // global index for element $K_m$ and function $v_i$
    **forall** $j \in L^{p_m}$ **do**
      $s = c(K_m, j)$ ;  // global index for element $K_m$ and function $v_j$
      **if** $v_i$ *represents Dirichlet lift* **then**
        $f_r = f_r - S_{ij}^{K_m}$;
      **else**
        $S_{rs} = S_{rs} + S_{ij}^{K_m}$
    $f_r = f_r + f_i^{K_m}$;

Among more practical challenges of $hp$-FEM belongs numerical quadrature. In low-order FEM numerical quadrature is usually not the main issue, since integrals of products of shape functions can be evaluated exactly using one quadrature rule (using values at vertices, edge middlepoints, etc.). For higher-order finite elements, polynomial degree of integrands in the week form may vary - product of 2 vertex functions requires quadrature rule accurate for polynomials of the second order while product of two 9th order edge functions requires quadrature rule of the 18th order. Hence, using one (very accurate) quadrature rule for all integrals would result in tremendous assembling times. Moreover, in case of curved or distorted element geometry, polynomial order of inverse reference mapping has to be taken into account. For triangles and parallelograms, corresponding inverse of the Jacobi matrix and Jacobian of the reference map are constant and thus, do not affect the order of the quadra-

ture rule needed to evaluate stiffness matrix entries exactly. The quadrature rule can be determined from the shape functions degrees only. This is not true for deformed quadrilaterals or elements with curved edges - here the inverse reference mapping can be nonpolynomial in genreal and question of choice of sufficient quadrature rule arises. From our experience insufficient quadrature rule can lead to disastrous results.

Table 2.3: Comparison of computational times for low-order and higher-order FEM.

| | **# DOFs** | **Assembling time** | **Solving time** | $H^1$**-error** |
|---|---|---|---|---|
| higher-order FEM | 264809 | 26.4 | 10.04 | $2 \times 10^{-10}$ |
| linear FEM | 264337 | 4.96 | 8.48 | 0.0025 |
| higher-order FEM | 655689 | 69.61 | 31.53 | $1.7 \times 10^{-11}$ |
| linear FEM | 658401 | 12.22 | 38.00 | 0.001 |
| higher-order FEM | 9869 | 0.41 | 0.12 | 0.0023 |
| linear FEM | 264337 | 4.96 | 8.48 | 0.0025 |



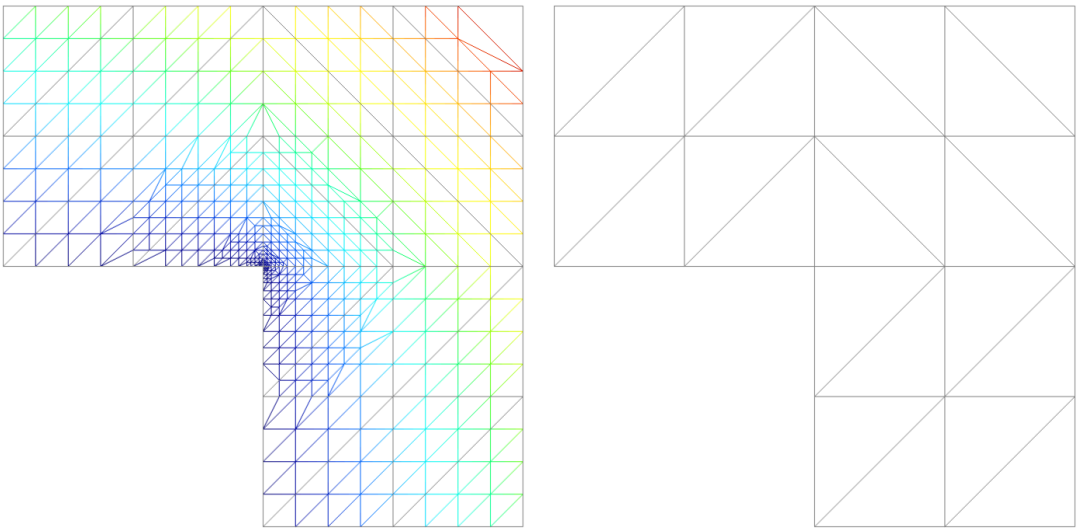

(a) higher-order FEM (655689 DOFs) (b) low-order FEM (658401 DOFs)

Figure 2.9: Adaptively obtained meshes for L-shape benchmark problem.

With numerical quadrature issues the computational time needed to assemble the stiffness matrix and the load vector is closely connected. For low-order FEM usually the majority of the computational time is spent in solving step, LU decomposition of the matrix or iterative solvers. For higher-order FEM the time needed to evaluate entries of the stiffness matrix usually exceeds the time needed to actually solve the resulting system. Comparisons of assembling and solving times for standard L-shape benchmark problem is shown in Tab. 2.3. It compares the time needed to evaluate entries

and the time the direct solver needed to solve the linear system for matrices obtained on $hp$-mesh (Fig. 2.9(a)) and linear mesh (Fig. 2.9(b)). For approximately the same sizes of matrices the assembling took more than twice longer than the direct solver in case of $hp$-FEM. For linear elements the time spent in the direct solver was twice or more times longer than the assembling time. Let us notice that resulting $H^1$-errors for both methods are incomparable (the last section in Tab. 2.3 shows sizes of matrices and corresponding times on the same error level). Comparison of computational times places even more emphasis on the optimal choice of quadrature rules. Possible speed-up of the assembling time in $hp$-FEM was also discussed in Remark 2.8, where pretransformed values of shape function derivatives are used.

The stiffness matrix represents another $hp$-FEM specialty – distinctive sparsity and conditioning properties of the matrix. Matrices obtained from $hp$-FEM problems are generally denser and worse conditioned than those coming from the linear finite elements. This has an affect on numerical methods, both direct and iterative, that are supposed to solve the resulting systems. Choices of another basis functions of the space $V_{hp}$ in order to improve conditioning properties of resulting matrices are discussed in [42, 50].



(a) 24 elements of the 10th degree (b) 24 elements of the 1st degree

Figure 2.10: Adaptively refined mesh for visualization purposes of higher-order solution; comparison with already linear triangles in linear FEM.

Concerning hierarchic $hp$-FEM, the last issue we would like to raise here is purely practical problem with visualization of results obtained by the method. Recall that in hierarchical approach solution is expressed as a linear combination of basis functions of various polynomial degrees and in order to visualize it, it has to be linearized

in some way. By visualization solution values only at, let us say, element vertices we are loosing information. To overcome this problem, elements should be "refined" for visualization purposes and solution on these subelements can be taken as linear. Visualization technique used in our code uses adaptive algorithm to get as accurate representation of higher-order solution as possible. It refines elements where solution oscillates or behaves wildly and it leaves large element where solution is more or less linear. Division of mesh elements into linear triangles for visualization purposes is depicted in Fig. 2.10. Each physical mesh element (triangle) is refined into subelements and solution on these subelements is visualized as linear. Notice, that more subelements is needed near the singularity due to the steep gradient there.

## 2.4 Demonstrative example

As mentioned in Chapter 1, higher-order finite element method is advantageous especially for solutions with smooth regions. By combining higher polynomial degrees ($p$-refinement) for smooth parts of the solution and $h$-refinements towards singularities, boundary or internal layers, an exponential convergence rate can be achieved.

To demonstrate superiority of higher-order elements over low-order elements for smooth solution, let us solve Poisson equation with known solution

$$-\nabla u = 2 \sin x \sin y \quad \text{in} \;\; \Omega = (0, \pi)^2, \tag{2.34}$$

with Dirichlet boundary conditions

$$u = \sin x \sin y \quad \text{on} \;\; \partial\Omega. \tag{2.35}$$

The exact solution to problem (2.34), (2.35) is equal to

$$u = \sin x \sin y.$$

We apply standard uniform $h$-refinements on bilinear, biquadratic and bicubic elements and uniform $p$-refinements on mesh consisting of 4 quadrilateral elements. Fig. 2.11 compares convergence of the relative error

$$\frac{||u_{hp} - u||_{H^1}}{||u||_{H^1}}$$

for these four approaches. Obviously, $p$-method gives exponential convergence, since the solution is infinitely smooth, approximation of smooth solution by low-order elements leads to very slow convergence. Approximated solutions are depicted in Fig. 2.12, where with the same number of DOFs we obtained 100 times better approximation using $p$-refinements.

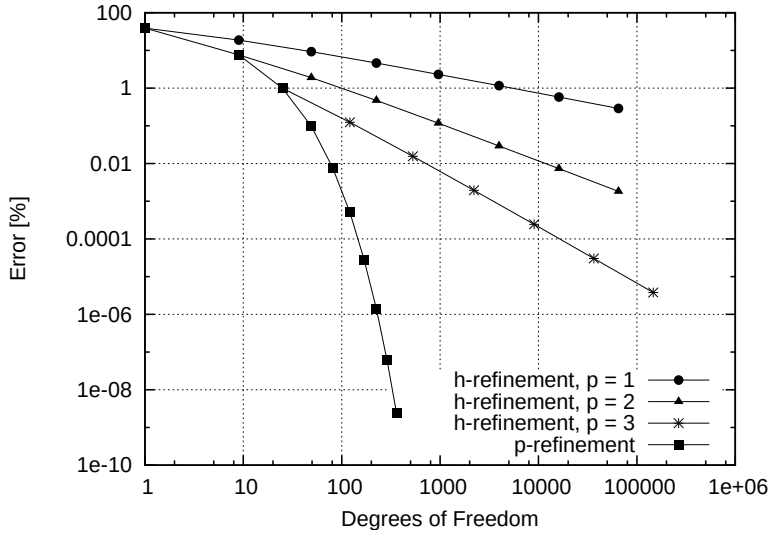
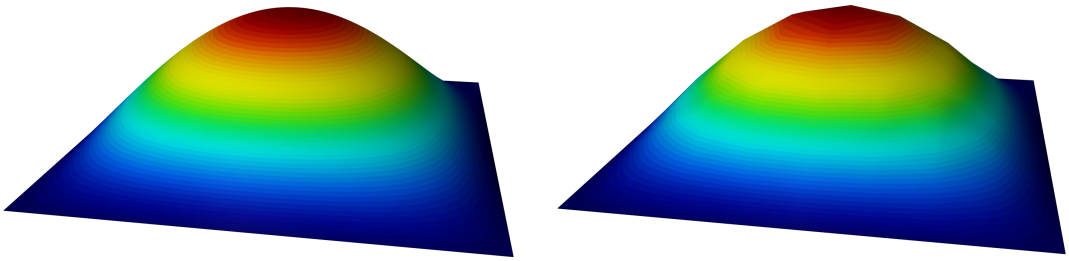



Figure 2.11: Comparison of convergence rates for $h$- and $p$-refinements.



(a) $p$-refinements (b) $h$-refinements

Figure 2.12: Approximated solution $u_{hp}$ obtained by (a) 4 biquartic elements (49 DOFs, error 0.0977%) and (b) 64 bilinear elements (49 DOFs, error 9.286%)

On the other hand, solutions with singularities or steep gradients are better approximated using piecewise low-order elements, hence, an optimal combination of large higher-order elements and small low-order elements is necessary in practical examples, resulting in need for $hp$-adaptivity.

CHAPTER 3

# Adaptive $hp$-FEM with arbitrary-level hanging nodes

The algorithmic difficulty of $hp$-adaptive algorithms is one of the main obstacles preventing adaptive $hp$-FEM from being employed widely in realistic engineering computations. In order to reduce its complexity, we split the algorithm into two parts – the treatment of arbitrary-level constrained approximation (arbitrary-level hanging nodes) and the fully local $hp$-refinement of elements. In addition to modularity and relative simplicity, another advantage of the presented approach is that it completely eliminates forced refinements which are induced by mesh regularity rules in all standard adaptivity algorithms. The algorithmic treatment of forced refinements is highly problematic due to their recursive nature, and obviously, they slow down the performance of the automatic adaptivity.

In this chapter we present a feasible algorithm for the treatment of multiple-level hanging nodes and we demonstrate its positive effect on both the performance and simplicity of automatic $hp$-adaptivity algorithms.

The chapter is organized as follows: Section 3.1 is devoted to the explanation of the treatment of arbitrary-level hanging nodes in $H^1$-conforming space. Simplified, fully local $hp$-adaptive algorithm together with several strategies how to select an optimal $hp$-refinement for one element is explained in Section 3.2. Effectivity of the automatic $hp$-adaptivity is demonstrated on two numerical examples in Section 3.3. Finally, an overview of four main open source adaptive $hp$-FEM codes is presented in Section 3.4.

## 3.1 Hanging nodes and irregular meshes

In order to introduce irregular meshes and hanging nodes, let us recall Definition 2.2. The mesh is called **regular** if for any two elements $K_i$ and $K_j$, $i \neq j$, just one of the following alternatives holds:

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

Otherwise the mesh is called **irregular**.

From all adaptive strategies let us start with the well-known *red-green* refinement strategy. This technique first subdivides elements with large error into geometrically nice subelements (red) and then it regularizes the mesh by additional refinements of adjacent elements (green). This approach preserves regularity of the mesh but when repeated refinements occur in the same part of the mesh it creates elements with sharp angles which cause problems in the finite element method. This is illustrated in Fig. 3.1.

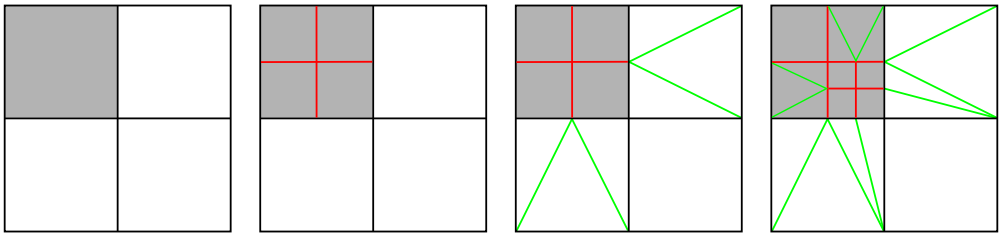


Figure 3.1: Red-green refinement - one level and repeated.

The "green" refinements can be avoided by introducing *hanging nodes*, i.e., by violating Definition 2.2, by allowing *irregular meshes* where element vertices lie in the interior of edges of other elements. In order to keep the implementation of the discretization simple, most finite element codes working with hanging nodes limit the maximum difference of refinement levels of adjacent elements to one (so-called 1-*irregularity rule* - see Definition 3.1) – see, e.g., [32, 34, 44].

**Definition 3.1 (Irregularity rules)** *By* ***k-irregularity rule*** *(or k-level hanging nodes) we mean a restriction on the mesh where the maximum difference of refinement levels of adjacent elements is k. In this context,* $k = 0$ *corresponds to the regular mesh and* $k = \infty$ *to the mesh with arbitrary-level hanging nodes.*

By using $k$-irregular meshes for $k \geq 1$ one can avoid badly shaped elements since the "green" refinements can be replaced by "red" ones and the last "green" levels of refinements (thin triangles) are avoided. Nevertheless, unwanted (forced) refinements are introduced anyway unless $k = \infty$ is allowed. In practice, usually 1-irregular meshes are sufficient when refining towards point singularities, but multiple level hanging nodes appear in problems with boundary or internal layers.

### 3.1.1 Arbitrary level hanging nodes

Arbitrary-level hanging nodes technique was first introduced in [41] in the context of continuous higher-order elements for second-order elliptic problems and in [46] in context of vector-valued $H(\mathrm{curl})$-conforming edge elements for Maxwell's equations. The requirement of continuity of approximation (or its tangential component in $H(\mathrm{curl})$ space) across an edge which contains hanging nodes is equivalent to a set of linear algebraic relations between coefficients of constraining and constrained shape functions on that edge. In [48], these relations were derived explicitly using transition matrices between pairs of constraining and constrained polynomial bases. In our approach the constraining-constrained relations are calculated in specific situations on-demand.

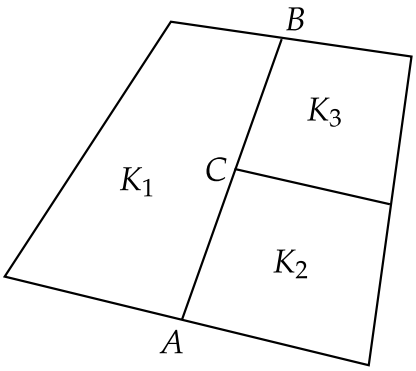



Figure 3.2: Example of a mesh with one-level hanging node.

To begin with, assume a simple geometrical situation shown in Fig. 3.2: Two elements $K_2$ and $K_3$ are adjacent to the edge $AB$ of an element $K_1$, and the edge $AB$ is equipped with the polynomial degree $1 \leq p_{AB}$. By design, every constrained edge inherits its orientation and polynomial degree from the constraining edge (even if this is in contradiction with the minimum rule (2.11)).

In $H^1$-conforming space there are $p_{AB} + 1$ constraining shape functions on the edge $AB$ which induce three different situations (Fig. 3.3 illustrates the simplest situation with $p_{AB} = 2$):

I: Constraining vertex function on $K_1$ corresponding to the vertex $A$ constrains the vertex functions on the elements $K_2$ and $K_3$ corresponding to the vertex $C$.

II: Constraining vertex function on $K_1$ corresponding to the vertex $B$ also constrains the vertex functions on the elements $K_2$ and $K_3$ corresponding to the vertex $C$.

III: $p_{AB} - 1$ constraining edge functions of polynomial degrees $p = 2, 3, \ldots, p_{AB}$ on $K_1$ corresponding to the edge $AB$ constrain:

(a) vertex functions on the elements $K_2$ and $K_3$ corresponding to the vertex $C$,

(b) $p-1$ edge functions on the element $K_2$ corresponding to the edge $AC$,

(c) $p-1$ edge functions on the element $K_3$ corresponding to the edge $CB$.

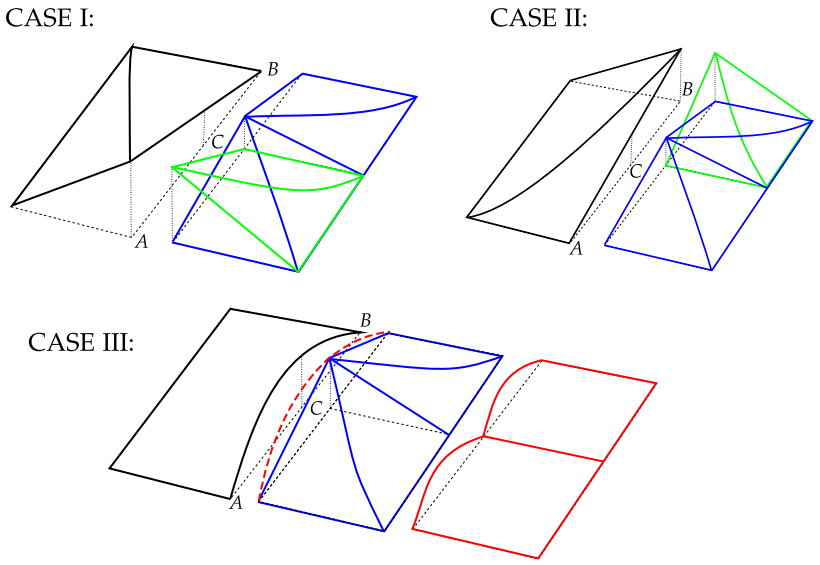



Figure 3.3: Constraining and constrained shape functions on a quadratic edge with one-level hanging node.

Note that interior bubble functions are neither constraining nor constrained functions since their support is one element and their values vanish on the element boundary, and therefore do not influence the calculation of constraint coefficients.

**Remark 3.1** *The minimum rule (2.11) is violated on edges with hanging nodes, where constraining element ($K_1$) has higher polynomial degree than constrained elements ($K_2$, $K_3$). On constrained elements additional higher-order edge functions need to be added in order to construct conforming basis function across the constraining edge $AB$.*

In case of multiple-level constraints we have two types of constraints - *direct* and *implied*. Consider, for illustration, the mesh with three-level hanging nodes shown in Fig. 3.4.

As in the previous case, there are $p_{AB}+1$ constraining shape functions on $K_1$ associated with the edge $AB$. *Direct constraints* are vertex functions associated with vertices lying on the edge $AB$ ($C_1, C_2$ and $C_3$) and edge functions associated with edges lying on the edge $AB$ ($AC_3, C_3C_2, C_2C_1$ and $C_1B$). Since values of constrained vertex functions are nonzero on element edges, they may further constrain vertex functions

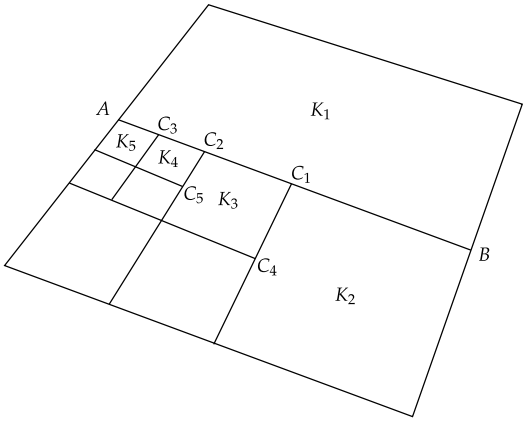



Figure 3.4: Example of a mesh with three-level hanging nodes.

associated with vertices lying away from the constraining edge $AB$ ($C_4$ and $C_5$). We call these constraints *implied constraints*. Note that directly constrained edge functions do not imply additional (implied) constraints (their values are zero on edges $e \not\subseteq AB$) and edge functions are never implied constraints since constrained vertex functions are linear on edges lying away from the edge $AB$. Example of two vertex functions (associated with vertex $A$ and vertex $B$) and a quadratic edge basis function associated with the edge $AB$ is shown in Fig. 3.5.

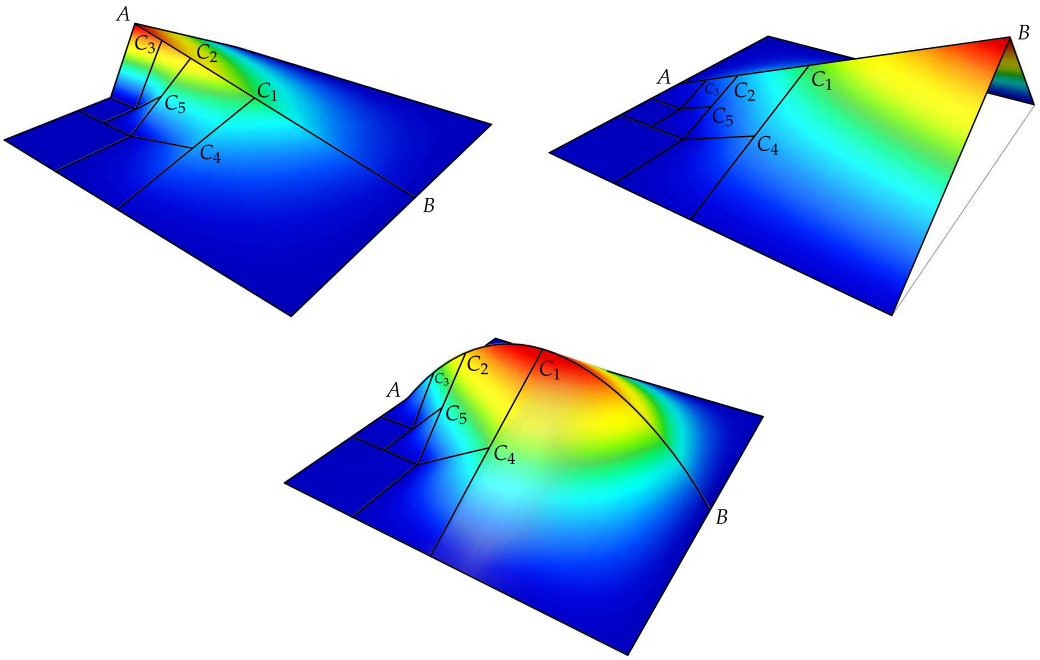



Figure 3.5: Two vertex functions and quadratic edge function on the edge $AB$.

Next, let us describe in detail how the constraint coefficients are calculated. This algorithm requires a unique enumeration of the basis functions $v_1, v_2, \ldots, v_N$ of the finite element space $V_{h,p}$ as well as a unique local enumeration of the shape functions on the reference domain $\hat{K}$. First assume a mesh element $K_i$ in the mesh whose edge $e = (a, b)$ is constrained by another mesh edge $AB$, $e \subset AB$. This element is mapped onto the reference domain $\hat{K}$ via an inverse reference map $\boldsymbol{x}_{K_i}^{-1} : K_i \to \hat{K}$. Let $\hat{e}$ be the edge of $\hat{K}$ such that $\boldsymbol{x}_{K_i}(\hat{e}) = e$, and by $\varphi_0^v$ and $\varphi_1^v$ let us denote vertex shape functions associated with endpoints $v_0, v_1$ of the edge $\hat{e}$, and by $\varphi_2^e, \varphi_3^e, \ldots, \varphi_{p_e}^e$ let us denote edge shape functions on $\hat{K}$ associated with the edge $\hat{e}$.

For each constrained edge we store a reference to the standard node associated with the constraining edge $AB$ and the index $q^e$ identifying uniquely the geometrical position of the constrained edge $e$ within $AB$. Fig. 3.6 shows the values of the index $q^e$ for various geometrical cases.

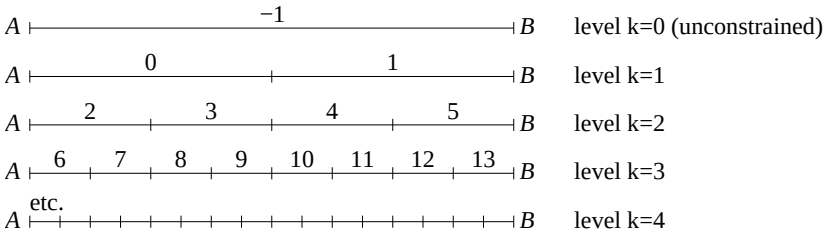



Figure 3.6: Geometrical situations on a constraining edge.

Now, let us assume a constraining edge function $\psi$ of the space $V_{h,p}$ on the edge $AB$ of the polynomial degree $p$. The function $\psi$ determines the constraint coefficients $\alpha_0^{e,p}, \alpha_1^{e,p}, \ldots, \alpha_p^{e,p}$ corresponding to vertex functions $\varphi_0^v$, $\varphi_1^v$ and edge functions $\varphi_2^e, \varphi_3^e, \ldots, \varphi_p^e$ on the element $K_i$ (associated with the edge $e \subset AB$). Constraining coefficients are determined as follows:

- $\alpha_0^{e,p} = \psi(a)$, where $a$ denotes one of the endpoints of the edge $e$.
- $\alpha_1^{e,p} = \psi(b)$, where $b$ denotes the other endpoint of the edge $e$.
- Values of coefficients $\alpha_2^{e,p}, \alpha_3^{e,p}, \ldots, \alpha_p^{e,p}$ are obtained by solving a system of $p-1$ linear algebraic equations of the form

$$\sum_{j=2}^{p} \alpha_j^{e,p} \tilde{\varphi}_j^e(y_i^p, -1) = \tilde{\psi}(y_i^p) - \alpha_0^{e,p} \tilde{\varphi}_0^v(y_i^p, -1) - \alpha_1^{e,p} \tilde{\varphi}_1^v(y_i^p, -1), \qquad 0 \le i \le p,$$

  where $y_i^p \in [-1, 1]$ are the $p+1$ Chebyshev points of degree $p$ on the edge $\hat{e}$, and $\tilde{\psi}(t) = \psi(\boldsymbol{x}_{K_i}(\boldsymbol{\xi}(t)))$, where $\boldsymbol{\xi}$ is a mapping from the interval $[-1, 1]$ onto the reference edge $\hat{e}$. Functions $\tilde{\varphi}_0^v, \tilde{\varphi}_1^v, \tilde{\varphi}_2^e, \tilde{\varphi}_3^e, \ldots, \tilde{\varphi}_p^e$ are shape functions associated

with the bottom edge $e_1$ of the reference domain. The bottom edge is used in order to simplify the implementation, otherwise the mapping $\xi$ from the interval $[-1, 1]$ onto edge $\hat{e}$ would have to used.

Then,

$$\hat{\psi}_e = \alpha_0^{e,p} \varphi_0^v + \alpha_1^{e,p} \varphi_1^v + \sum_{j=2}^{p} \alpha_j^{e,p} \varphi_j^e$$

is a new edge function of degree $p$ on $\hat{K}$. After transforming $\hat{\psi}_e$ to the element $K_i$, it matches exactly the corresponding part of the constraining edge function $\psi$ across the edge $e$.

More details on arbitrary-level hanging nodes technique can be found in [41, 46].

## 3.2 Adaptive $hp$-FEM

The major difference between the automatic adaptivity in the standard $h$-FEM and in the $hp$-FEM is that in the latter case, a higher-order element can be refined in many different ways. One can either increase its polynomial degree without spatial subdivision ($p$-refinement) or the element can be split in the space with various distributions of polynomial degrees in subelements. Fig. 3.7 illustrates this for a quartic triangular and quadrilateral element.

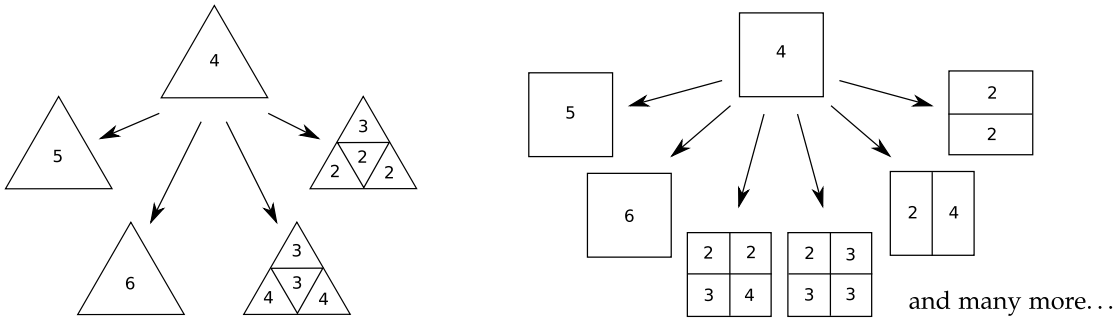



Figure 3.7: Multiple element refinement options in the adaptive $hp$-FEM

This means that traditional error estimates (that deliver one number per element) do not provide enough information to guide the $hp$-adaptivity. In order to select an optimal refinement, one needs to use some information about the *shape* of the error function $\mathcal{E}_{h,p} = u - u_{h,p}$. In principle, this information could be recovered from suitable estimates of higher derivatives of the solution, but such approach is not very practical and rarely used. We prefer to estimate the error by means of the so-called *reference solutions* [11]. This approach not only provides the desired information about the shape of the error $\mathcal{E}_{h,p}$, but it is very robust in the sense that it can be used for

various types of PDEs, and thus it is suitable for multiphysics coupled problems we are concerned with. A reference solution is an approximation $u_{ref}$ of the exact solution $u$, that is more accurate than the coarse mesh approximation $u_{h,p}$, and thus the difference $u_{ref} - u_{h,p}$ provides a meaningful information about the error $\mathcal{E}_{h,p}$. In practice, the reference solution $u_{ref}$ is sought in an enriched finite element space $V_{ref}$ such that we subdivide all elements in the mesh uniformly and increase their polynomial degrees by one, i.e., $V_{ref} = V_{h/2,p+1}$. With an a-posteriori error estimate of the form

$$\mathcal{E}_{h,p} \approx u_{ref} - u_{h,p}, \tag{3.1}$$

the outline of our $hp$-adaptivity algorithm is described in Alg. 3.

**Algorithm 3**: Algorithm for $hp$-adaptivity.

**Data**: Assume an initial coarse mesh $\mathcal{T}_{h,p}$, prescribed tolerance $TOL > 0$ and refinement coefficient $k \in (0,1)$ is a user defined parameter determining how many elements will be refined in one adaptive step.

**Result**: Approximated solution $u_{ref}$ with $||u_{ref} - u_{h,p}||/||u_{ref}|| < TOL$ and an optimal $hp$-mesh.

```
bool done = false;
repeat
    Compute solution on current mesh u_{h,p} ∈ V_{hp};
    Compute reference solution u_{ref} ∈ V_{ref};
    Evaluate local errors on all elements e_i = ||E_{h,p}|_{K_i}||;
    Evaluate global error estimate ||E_{h,p}|| = Σ_{i=1}^{M} e_i, where M is number of
    elements;
    if ||E_{h,p}|| / ||u_{ref}|| < TOL then
        done = true;
    else
        Sort all elements by their error e_i;
        Denote the maximal element error by e_max;
        foreach element K_i do
            if e_i < k * e_max then
                break;
            else
                Find and perform optimal refinement;
    Adjust polynomial degrees on unconstrained edges using the minimal rule;
until not done ;
```

**Remark 3.2** *The user defined parameter $k \in (0,1)$ prescribes how many elements will be refined in one adaptive step. For example, condition $e_i < 0.3 * e_{max}$ says that all elements with error higher than one third of the maximal element error will be refined. The smaller k is, the more elements is refined in each step.*

**Remark 3.3** *As a suitable norm $||\cdot||$ in Alg. 3 either the energy norm $||\cdot||_V$ or $H^1$ or $L^2$-norms can be taken, depending on the norm in which we want the error to be minimized.*

**Remark 3.4** *Note that the refinement of the element $K_i$ cannot cause any recursive refinements in the mesh due to the technique of arbitrary-level hanging nodes. In other words, no regularization of the mesh has to be performed after each adaptive step.*

**Remark 3.5** *As it is obvious from $V_{ref} = V_{h/2,p+1}$, the calculation of the reference solution is computationally expensive, mainly at later stages of the adaptive process. First, let us point out that the output of the adaptive procedure is the last reference solution, since it is the best approximation of the exact solution we obtained. Moreover, the computational cost of the reference solution may be reduced by using iterative solvers, where previous reference solution serves as a good initial guess for the next reference solution. Another possibility how to overcome this drawback of the hp-adaptivity is to define the basis of $V_{ref}$ as a hierarchic extension of the basis for $V_{hp}$ and solve for the reference solution using Schur complement [40]. For elliptic problems it is also possible to use a two-grid solver taking advantage of lower frequencies from $u_{hp}$ (see [31]).*

### 3.2.1 Selection of optimal refinement

Let $K \in \mathcal{T}_{h,p}$ be an element of polynomial degree $p_K$ marked for refinement. We consider the following $N_{ref} = l + (l+1)^4$ and $N_{ref} = l + (l+1)^4 + 2(l+1)^2$ refinement options for triangles and quadrilaterals respectively. Here $l \geq 0$ is a user input parameter and for $l = 2$ it yields 83 candidates for triangle and 101 for quadrilateral. Refinement options are divided into 3 types:

1. **$p$-candidates.**
   Increase the polynomial degree of $K$ by $1, 2, \ldots, l$ without spatial subdivision. This yields $l$ refinement candidates.

2. **Isotropic $hp$-candidates.**
   Split $K$ into four similar elements $K_1, K_2, K_3, K_4$ (as illustrated in Fig. 3.7). Define $p_0$ to be the integer part of $(p_K + 1)/2$. For each $K_i$, $1 \leq i \leq 4$ consider $l + 1$ polynomial degrees $p_0 \leq p_i \leq p_0 + l < p_K + 1$. This yields additional up to $(l+1)^4$ refinement candidates.

3. **Anisotropic $hp$-candidates.**
   In case of quadrilateral element, split $K$ into two subelements $K_5$, $K_6$ either vertically or horizontally (see Fig. 3.7). Define $p_0$ to be the integer part of $2(p_K+1)/3$. For both $K_5$, $K_6$ consider $l+1$ polynomial degrees $p_0 \leq p_i \leq p_0 + l < p_K + 1$. This yields additional up to $2(l+1)^2$ refinement options.

**Remark 3.6** *The condition $p_0 + l < p_K + 1$ eliminates candidates with polynomial degrees greater or equal to the polynomial degrees of the reference solution. For such candidates the reference mapping $V_{ref} = V_{h/2,p+1}$ is not sufficiently accurate.*

For each of these $N_{ref}$ options, let us define a piecewise-polynomial space $V_{cand} \subset V(K)$, $\dim V_{cand} = m_c$, generated by basis functions on element $K$ and its subelements

$$V_{cand} = \langle \varphi_1, \varphi_2, \ldots, \varphi_{m_c} \rangle,$$

where either $\varphi_i \in P^p(K)$ in case of $p$-candidate or $\varphi_i \in \{v \in V(K);\ v|_{K_j} \in P^{p_j}(K_j),\ j = 1,\ldots,4\}$ in case of isotropic $hp$-candidate or $\varphi_i \in \{v \in V(K);\ v|_{K_j} \in P^{p_j}(K_j),\ j = 5,6\}$ in case of anisotropic candidate.

Now for each refinement option we calculate an approximation of the reference solution $u_{ref}$ in the space $V_{cand}$ by one of the following algorithms:

**Projection-based interpolation**

The interpolant $u_{cand}$ is obtained as a sum of vertex, edge and bubble interpolants $u_{cand} = u^v + u^e + u^b$. Vertex interpolant is defined as

$$u^v = \sum c_i^v \varphi_i^v,$$

where $\varphi_i^v$ are vertex functions at all vertices of $K$ or $K_j, j = 1,\ldots,4$ and $c_i^v$ values of the reference solution $u_{ref}$ at these vertices. Edge interpolants $u^{e_j}$ for all edges of elements $K$ or $K_j, j = 1,\ldots,4$ are obtained as the best approximation of the residual $u_{ref} - u^v$ on the edge $e_j$:

$$u^{e_j} = \sum c_i^{e_j} \varphi_i^{e_j}, \qquad ||(u_{ref} - u^v) - u^{e_j}||_{H_0^1(e_j)} \to \min,$$

where $\varphi_i^{e_j}$ are edge functions associated with the edge $e_j$. In a similar way, the bubble interpolant $u^b = \sum u^{b_j}$, where $u^{b_j}$ are obtained as the best approximation of the residual $u_{ref} - u^v - u^e$, where $u^e = \sum u^{e_j}$, on elements interiors:

$$||(u_{hp} - u^v - u^e) - u^{b_j}||_{H_0^1(K_j)} \to \min.$$

Resulting interpolant $u_{cand}$ approximates the reference solution $u_{ref}$ exactly at all vertices and error is minimized on all edges and in the interior of element $K$ or elements $K_j, j = 1, \ldots, 4$.

### $H^1$ or $L^2$ projection

Projected solution $u_{cand} = \sum_{k=1}^{m_c} c_k \varphi_k$ is obtained by the standard projection onto the space $V_{cand}$:

$$(u_{cand}, \varphi_j) = (u_{ref}|_K, \varphi_j), \quad j = 1, \ldots, m_c \tag{3.2}$$

or

$$\sum_{k=1}^{m_c} c_k (\varphi_k, \varphi_j) = (u_{ref}|_K, \varphi_j), \quad j = 1, \ldots, m_c,$$

where scalar product $(\cdot, \cdot)$ is either $H^1$ or $L^2$ product. Thus, the error in corresponding norm is minimized on the whole area of element $K$:

$$||u_{cand} - u_{ref}|| \to \min.$$

### Fast projection in reference domain

Since we are only roughly estimating possible impact of particular refinements on the global solution, we may, in order to speed up the selection of optimal refinement, perform certain simplifications without bigger impact on the convergence rate. We design a partial projection taking place on the reference domain, where the shape functions are first orthonormalized on the reference element $\hat{K}$ and then the solution $u_{cand}$ can be evaluated in a faster way (without solving a system of linear equations for each candidate). Let us describe the procedure in detail. In the following all norms and scalar products are either from $H^1$ or $L^2$ space, depending in which norm we are decreasing the error in the adaptivity, and numerical integration takes place on the reference domain only, neglecting the effect of reference mappings on candidates' errors.

First, the shape functions for polynomial degrees $p = 1, \ldots, 10$ (the maximal allowed degree) are orthonormalized on the reference element $\hat{K}$ using Gram-Schmidt orthogonalization process. Let $\psi_k$, $k = 1, \ldots, N$ be all vertex, edge and bubble functions on element $\hat{K}$ for maximal polynomial degree. Orthonormal set of shape functions is obtained as follows

$$\bar{\psi}_k = \psi_k - \sum_{l=1}^{k-1} (\psi_k, \bar{\psi}_l)\ \bar{\psi}_l$$

$$\tilde{\psi}_k = \frac{\bar{\psi}_k}{||\bar{\psi}_k||}$$

Then for $p$-candidates the basis of the space $V_{cand}$ is a subset of our orthonormal basis

$$\{\tilde{\psi}_k,\, k = 1, \ldots, N\}$$

and therefore the projection (3.2) reduces just to evaluation of the right hand side

$$u_{cand} = \sum_{k=1}^{m_c} (\overline{u}_{ref}|_K,\ \tilde{\psi}_k)\ \tilde{\psi}_k,$$

where the values of the reference solution $\overline{u}_{ref}$ are taken untransformed, on the reference domain.

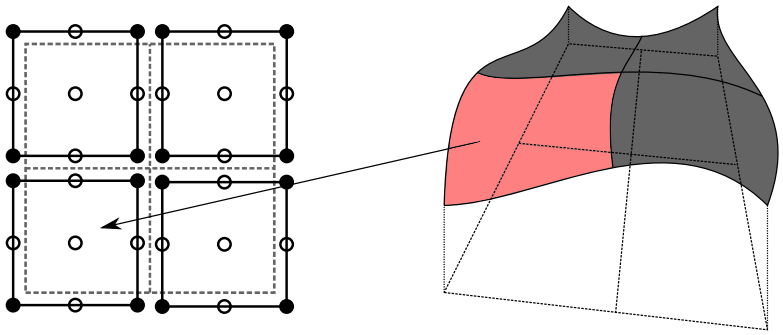


Figure 3.8: Separate treatment of subelements of $hp$-candidate.

For $hp$-candidates with an isotropic spatial subdivision the $u_{cand}$ is obtained as a sum of $u^j_{cand}$, $j = 1 \ldots, 4$, where $u^j_{cand}$ are projections of the reference solution $u_{ref}$ onto $P^{p_j}(K_j)$ obtained separately from each other (see Fig. 3.8):

$$u^j_{cand} = \sum_{k=1}^{m_j} \frac{1}{4}\, (\overline{u}_{ref}|_{K_j},\ \tilde{\psi}_k)\ \tilde{\psi}_k.$$

In order to have $p$ and $hp$-candidates comparable, projected solutions for $hp$-candidates are divided by 4, since we are integrating over four times larger domain.

Since no connection between subelements $K_j$ is considered (degrees of freedom are doubled on interior edges) the solution $u_{cand}$ obtained by gluing 4 projections together can be generally discontinuous on the interior edges. However the projected solution $u_{cand}$ serves only for decision between different refinement options and it turned out to be sufficient. The situation for anisotropic refinements would be analogous to isotropic candidates.

The error of the particular candidate is then obtained as a suitable norm of the difference $u_{cand} - u_{ref}|_K$. The goal is to select a candidate with small projection error but

also reasonable number of added degrees of freedom to obtain the steepest possible convergence rate. We are expecting the overall convergence to be exponential

$$||\mathcal{E}_{h,p}|| \approx Ce^{\alpha N}.$$

Thus, we choose a candidate which maximizes the rate

$$\frac{\ln(||u_{hp}|_K - u_{ref}|_K||) - \ln(||u_{cand} - u_{ref}|_K||)}{N_{cand} - N_{hp}} \to \max. \tag{3.3}$$

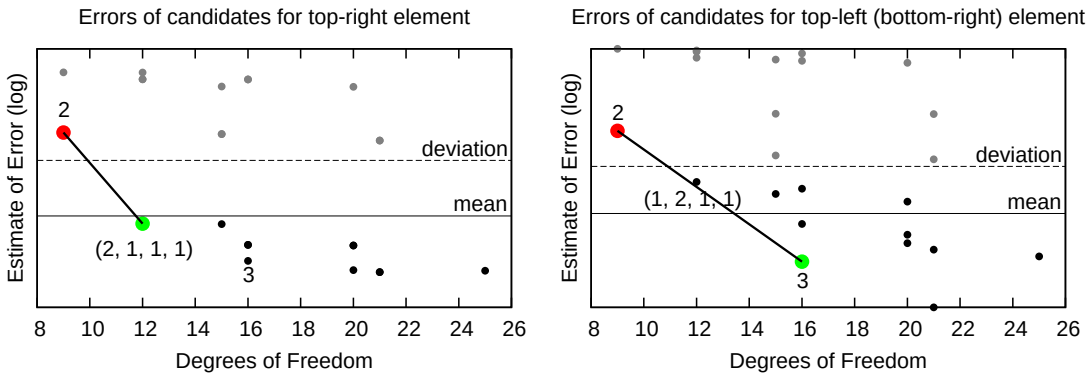



Figure 3.9: Refinement candidates - red: the original element (biquadratic), gray: rejected candidates (high error), black: considered candidates, green: selected candidate (black one with highest ratio (3.3)).

For illustration of the selection process, Fig. 3.9 shows candidates' errors with respect to their number of degrees of freedom for L-shape benchmark problem. It shows refinement options for two elements in the initial mesh, consisting of 3 biquadratic elements. First, all candidates with error higher than error of original element (red color) are rejected and the mean value and the deviation are computed from the rest. In order to avoid candidates with small DOFs increase and small error decrease (possibly with good ratio (3.3), but having almost no effect on the mesh and approximated solution), we also reject candidates with error higher than mean plus deviation. See Alg. 4 describing the process of selection of an optimal refinement.

**Remark 3.7** *Note that if the arbitrary-level hanging nodes are not allowed, all hp-candidates with spatial subdivision may influence the adjacent elements and thus this effect should be taken into account while searching for the optimal refinement. On the other hand the arbitrary-level hanging nodes technique makes the whole adaptive procedure local, thus simple.*

**Algorithm 4**: Algorithm for selection of an optimal refinement.

**Data**: List of $N$ candidates for refinement with errors $e_{cand}$ and numbers of DOFs $N_{cand}$. Original element is the first in the list $(e_0, N_0)$.
**Result**: Optimal refinement - index "best-cand" to the list of candidates.

$k = 0$;
mean = $\ln(e_0)$;
deviation = $(\ln(e_0))^2$;
**for** $cand = 1..N$ **do**
  **if** $(e_{cand} < e_0)$ **then**
    mean = mean + $\ln(e_{cand})$;
    deviation = deviation + $(\ln(e_{cand}))^2$;
    $k = k + 1$;

mean = mean$/k$;
deviation = $\sqrt{\text{deviation}/k - \text{mean}^2}$;

max-score = 0;
best-cand = 0;
**for** $cand = 1..N$ **do**
  **if** $(e_{cand} < mean + deviation)$ **then**
    score = $\dfrac{\ln(e_0) - \ln(e_{cand})}{N_{cand} - N_0}$;
    **if** $(score > max\text{-}score)$ **then**
      max-score = score;
      best-cand = cand;

## 3.3 Numerical examples

To demonstrate the automatic *hp*-adaptive process and to show its advantages over the standard *h*-adaptivity with low-order finite elements, we present two numerical examples. One is from electromagnetism and shows superiority of *hp*-FEM over low-order FEM for the discretization of the magnetic potential. Second is a benchmark problem for the incompressible flow - backward facing step, and it demonstrates that *hp*-FEM can be successfully used for the flow problems as well.

### 3.3.1 Continual induction heating of nonferrous cylindrical bodies

Continual induction heating of nonferrous metal bodies is a common technique used to improve mechanical properties of metals. Its numerical solution is relatively complicated problem for several reasons. First, the electromagnetic field is mutually coupled with temperature field (parameters are temperature-dependent), boundary conditions change in time due to the movement of the inductor, and a remeshing of the computational domain is necessary at each time level. Our goal in this section is to demonstrate the automatic $hp$-adaptivity for a single equation, therefore in order to keep the demonstrative example simple, we solve the equation for the electromagnetic field at particular time, ignoring the temperature field. We address coupled problems, where more physical fields are involved, in the next chapter.

#### Mathematical model

The computational domain, shown in Fig. 3.10, consists of a nonferrous cylinder, a moving inductor and a sufficiently large surroundings (air). Problem is modeled in the axisymmetric arrangement. The electromagnetic field satisfies the following equation

$$\frac{\partial^2 \underline{A}_\alpha}{\partial r^2} + \frac{1}{r}\frac{\partial \underline{A}_\alpha}{\partial r} - \frac{\underline{A}_\alpha}{r^2} + \frac{\partial^2 \underline{A}_\alpha}{\partial z^2} - i\,\omega\gamma\mu\underline{A}_\alpha - \mu\gamma v_z \frac{\partial \underline{A}_z}{\partial z} = -\mu \underline{J}_{\text{ext},\alpha}. \tag{3.4}$$

Here, $\underline{A}_\alpha$ is the angular component of the phasor of the magnetic vector potential $\boldsymbol{A}$, $\underline{J}_{\text{ext},\alpha}$ denotes external current prescribed in the inductor, and $v_z$ is the axial velocity of the moving inductor. Equation (3.4) is equipped with homogeneous Dirichlet boundary condition. Distribution of magnetic induction $\boldsymbol{B} = \text{curl}\underline{\boldsymbol{A}}$ at particular time level is depicted in Fig. 3.11.

#### Numerical results

The solution contains singularities along the inductor and pipe boundaries. The initial mesh consists of 45 quadrilateral elements of the first order. Since the exact solution $\underline{A}$ of the problem is unknown, we use the "overkill" solution to approximate the error of $A_{hp}$ in $H^1$-norm.

In Fig. 3.12 we plot the estimated relative error in percents in log–log scale with respect to the number of DOFs. For reference, we include the convergence for $h$-adaptivities on meshes with elements of low-order (first and second order). Notice that standard linear elements need almost 100 times more degrees of freedom than $hp$-FEM. Fig. 3.13 shows the convergence history of coarse and fine $hp$-meshes. Relative error in $H^1$-norm is estimated using the "overkill" solution with approximately 170000 DOFs (obtained by $hp$-adaptivity).

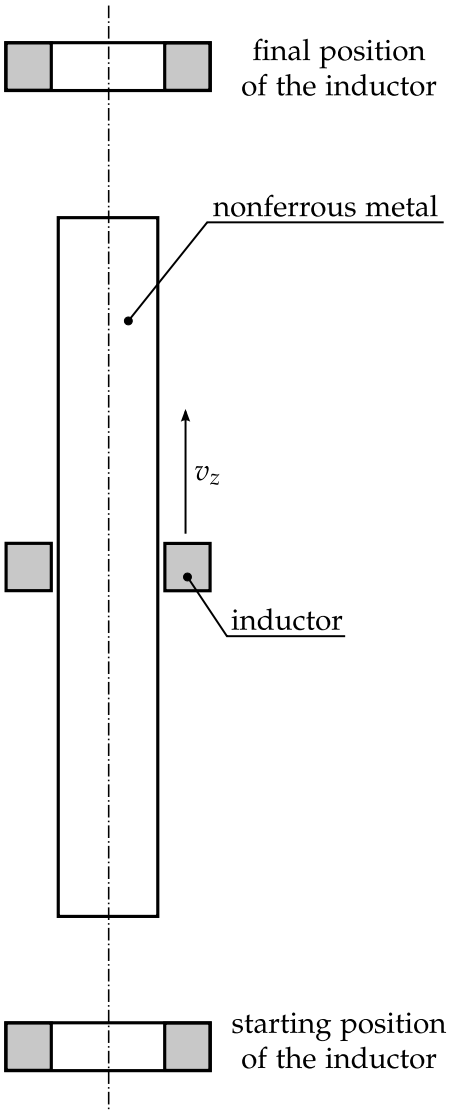



Figure 3.10: Computational domain for continual induction heating problem.

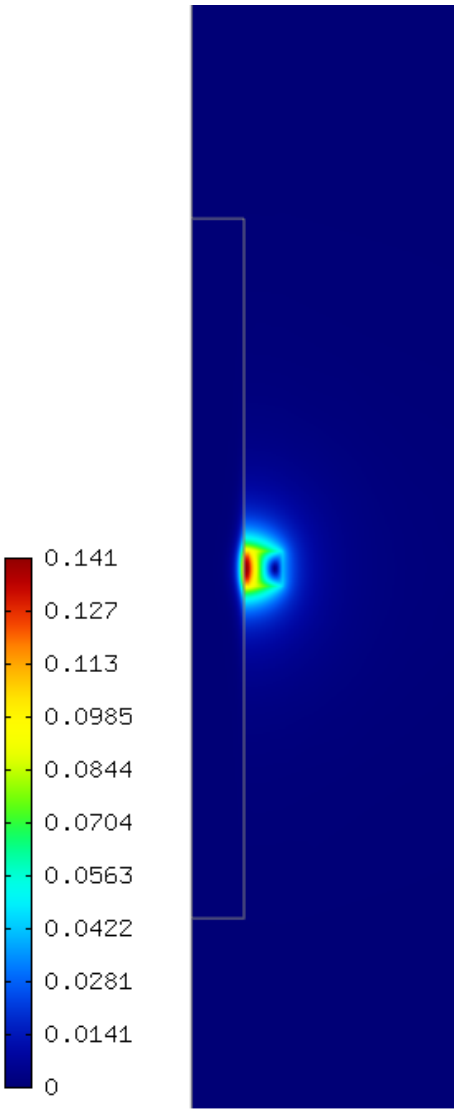



Figure 3.11: Distribution of magnetic induction $\boldsymbol{B} = \mathrm{curl}\underline{\boldsymbol{A}}$

Final coarse *hp*-mesh and coarse *h*-meshes with low-order elements are depicted in Fig. 3.14 together with the number of DOFs and estimated relative error in $H^1$-norm. In *hp*-meshes, different colors denote different polynomial degrees on elements (for particular colors and degrees see the scale next to the *hp*-mesh in Fig. 3.14). The algorithm automatically chooses larger elements with higher polynomial degree in areas where the solution is smooth and smaller elements with lower polynomial degree near singularities.

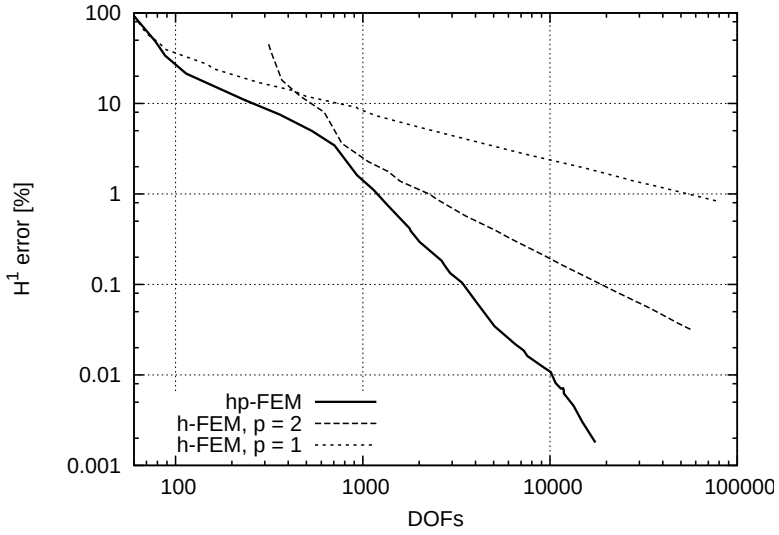



Figure 3.12: Estimated convergence curves for *hp*-meshes and low-order *h*-FEM.

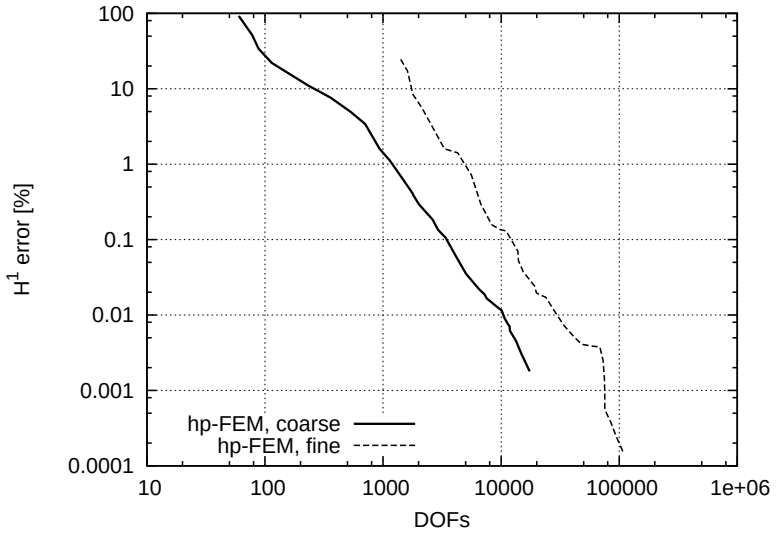



Figure 3.13: Estimated convergence curve for coarse and fine *hp*-meshes.

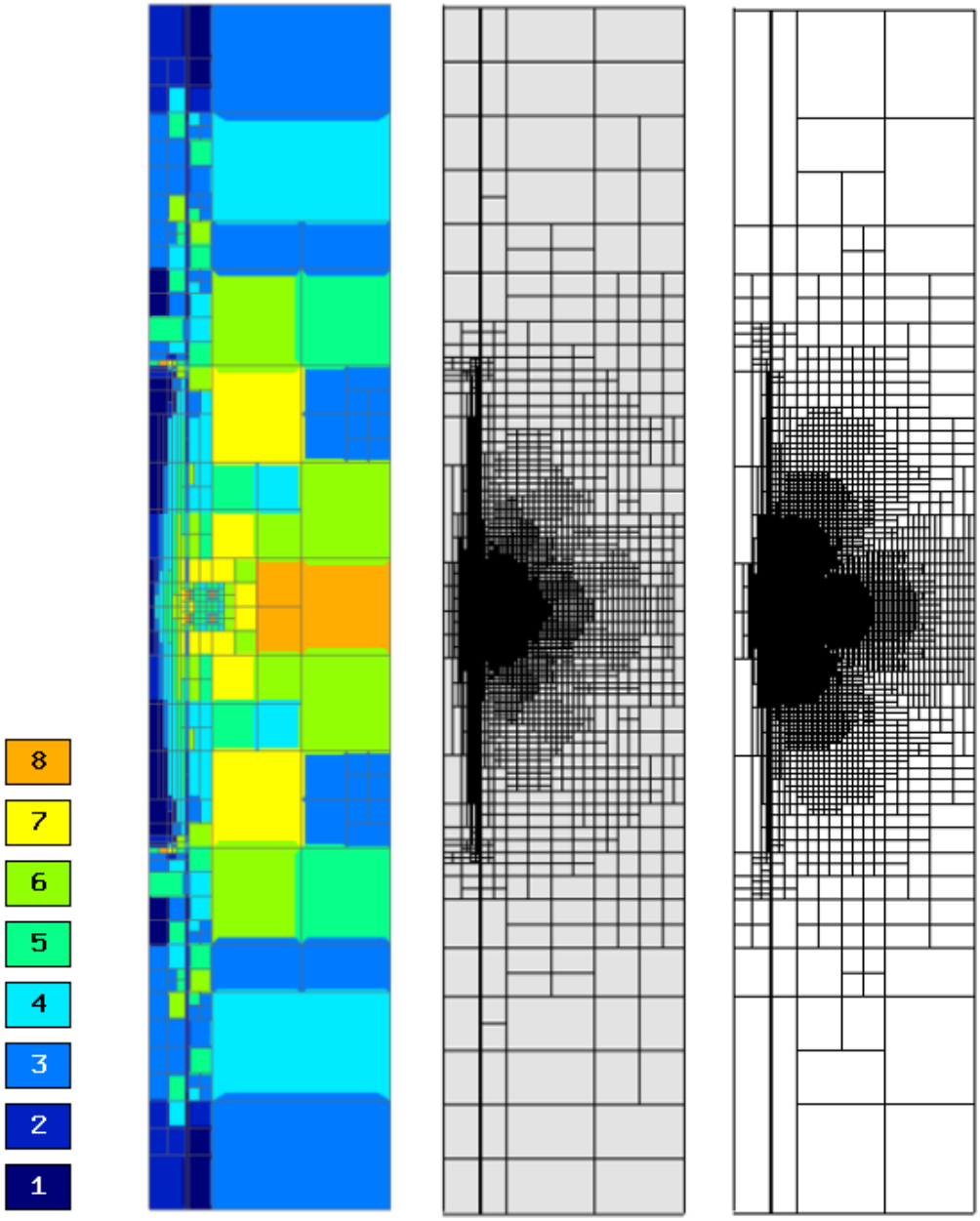



Figure 3.14: Final $hp$-mesh (17452 DOFs, error 0.0018%), $h$-mesh with $p = 2$ (57502 DOFs, error 0.031%) and $h$-mesh with $p = 1$ (77172 DOFs, error 0.838%).

### 3.3.2 Incompressible flow over a backward-facing step

Fluid flows in channels with flow separation and reattachment of the boundary layers are encountered in many problems. Among these problems, a backward-facing step has the simplest geometry and can be used as a benchmark problem. Flow separation and flow reattachment depend on the Reynolds number and geometrical parameters of the channel. We solve the incompressible flow over the step for several values of the Reynolds number (varying from 100 to 1000) and compare computations on meshes with uniform refinements, both $h$ and $p$, with our adaptive strategies described in this chapter. Computed reattachment lengths and velocity profiles are compared with numerical study [16].

#### Description of the problem

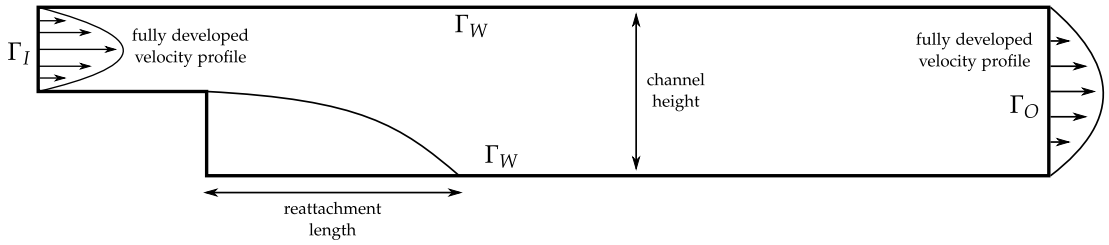



Figure 3.15: Backward-facing step (not to scale).

Incompressible flow obeys Navier-stokes equations

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p - \nu \Delta \boldsymbol{u} = 0 \tag{3.5}$$

and continuity equation

$$\operatorname{div} \boldsymbol{u} = 0. \tag{3.6}$$

Here $\boldsymbol{u}(\boldsymbol{x}, t) = (u_x, u_y)$ represents velocity vector, $p$ denotes kinematic pressure (dynamic pressure divided by density $\rho$) and $\nu$ kinematic viscosity. Computational domain and boundary notation is depicted in Fig. 3.15. Expansion ratio , i.e., the ratio of the channel height downstream of the step to the channel height upstream of the step is 2. Let us define characteristic length $D$ to be the hydraulic diameter of the inlet channel (which is twice the inlet channel height) and characteristic velocity $U$ to be the mean inlet velocity (which is two thirds of the maximum inlet velocity). Then the dimensionless length is $\boldsymbol{x}' = \boldsymbol{x}/D$ and the dimensionless velocity $\boldsymbol{u}' = \boldsymbol{u}/U$. The dimensionless time can be expressed as $t' = Ut/D$ and the dimensionless pressure satisfies $p' = p/U^2$. Using introduced notation Navier-stokes equations can be written

in a dimensionless form as follows

$$\frac{\partial \boldsymbol{u}'}{\partial t'} + \boldsymbol{u}' \cdot \nabla \boldsymbol{u}' + \nabla p' - \frac{1}{Re} \Delta \boldsymbol{u}' = 0, \tag{3.7}$$

where

$$Re = \frac{DU}{\nu}$$

denotes Reynolds number describing flow properties. For simplicity of notation we omit apostrophes in the following.

Equations (3.7) are equipped with initial condition

$$\boldsymbol{u}(\boldsymbol{x}, 0) = \boldsymbol{u}_0(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega,$$

and boundary conditions

$$\begin{aligned} \boldsymbol{u} &= \boldsymbol{u}_D && \text{on the inlet } \Gamma_I, \\ \boldsymbol{u} &= 0 && \text{on channel walls } \Gamma_W, \\ \nu \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} &= (p - p_{\text{ref}})\, \boldsymbol{n} && \text{on the outlet } \Gamma_O. \end{aligned}$$

Here $\boldsymbol{u}_D$ represents fully developed plane Poiseuille flow between parallel plates, so that the velocity profile is parabolic and mean dimensionless velocity is one

$$\boldsymbol{u}_D(x, y) = \left( \frac{3}{2}\, 4(y-1)(2-y),\ 0 \right).$$

**Numerical results**

Well-known Taylor-Hood $Q_{k+1}/Q_k$ elements satisfying Babuška-Brezzi condition are used. Here velocities $u_x$, $u_y$ and $p$ are approximated by continuous finite elements, where polynomial degree on element $K$ for velocity is $p_K + 1$, while polynomial degree for pressure is $p_K$.

$$\begin{aligned} W &= \{v \in H^1(\Omega), v = 0 \text{ on } \Gamma_I \cup \Gamma_W\}, \\ \boldsymbol{u} \in \boldsymbol{V}_{hp} &= \left\{ \boldsymbol{v} \in W^2 \cap (C(\overline{\Omega}))^2; \boldsymbol{v}|_K \in (P^{p_K+1}(K))^2, \forall K \in \mathcal{T} \right\}, \\ p \in Q_{hp} &= \left\{ q \in H^1(\Omega) \cap C(\overline{\Omega}); q|_K \in P^{p_K}(K), \forall K \in \mathcal{T} \right\}. \end{aligned}$$

Thus, there is one mesh for all three fields, but polynomial degrees on elements may generally vary throughout the mesh.

The steady state solution was sought for several values of the Reynolds number

($Re = 100 - 1000$) using the automatic $hp$-adaptation described in this chapter. The automatic $hp$-adaptive strategy was applied to find an optimal mesh for the velocity field starting from a very coarse mesh depicted in Fig. 3.16.

Figure 3.16: Initial coarse mesh - starting point for adaptive processes.

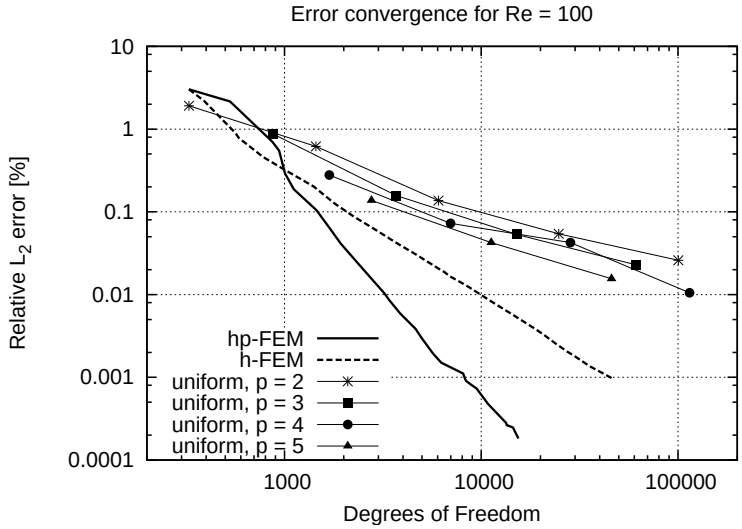



Figure 3.17: Convergence history of $L^2$-errors in magnitude of velocity - comparison of $hp$-adaptivity, $h$-adaptivity and uniform refinements.

The adaptivity was guided by relative $L^2$-error of the magnitude of the velocity. On each time level, adaptivity was run to find an optimal mesh for particular time instant until we reached steady state. In this chapter we do not describe how we deal with different meshes for particular time steps (this will be described in Chapter 4) or how to optimize time stepping. At this point our goal is to demonstrate advantages of described $hp$-adaptive strategies over the standard $h$-adaptivity or uniform meshes.

Convergence history of errors of the velocity field (at steady state time level for $Re = 100$) is shown in Fig. 3.17. For comparison we display also convergence history for $h$-adaptivity on $Q^2/Q^1$ elements as well as errors obtained on several uniformly refined meshes with $Q^2/Q^1$, $Q^3/Q^2$, $Q^4/Q^3$, $Q^5/Q^4$ elements. Note, that with standard $Q^2/Q^1$ elements one needs more than three times more degrees of freedom than with $hp$-FEM to reach the same level of accuracy and with uniformly refined meshes with uniform

distribution of polynomial orders many more DOFs would be needed. Adapted $hp$-meshes and $h$-meshes for all adaptive steps for $Re = 100$ are shown in Fig. 3.18 step by step. The algorithm automatically focuses on the singularity at the step by choosing smaller elements with lower polynomial orders there, while it selects larger elements with higher orders in areas where solution is smoother.

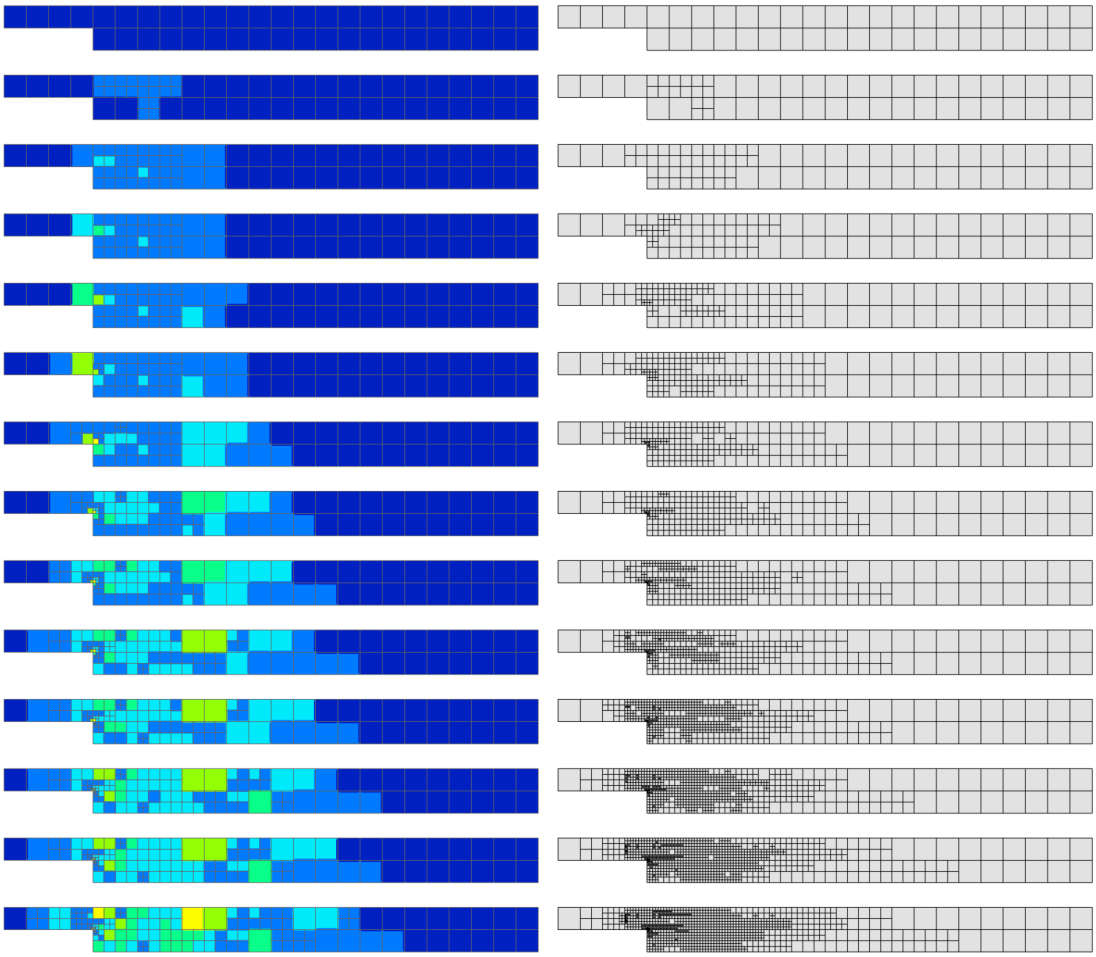


Figure 3.18: Comparison of meshes during the adaptive process (guided by the $L^2$-norm) for $Re = 100$. Left: $hp$-meshes (last 4942 DOFs, error 0.0042%). Right: $h$-meshes (last 9759 DOFs, error 0.0102%).

Similarly, the adaptive computation was run guided by the $H^1$-norm of $|\boldsymbol{u}|$; comparison of convergence history is shown in Fig. 3.19. The adaptive process clearly targets the singularity at the step using many spatial refinements and lower polynomial orders since the singularity in the $H^1$-norm is much stronger than in $L^2$-norm. Obtained

optimal meshes are displayed in Fig. 3.20.

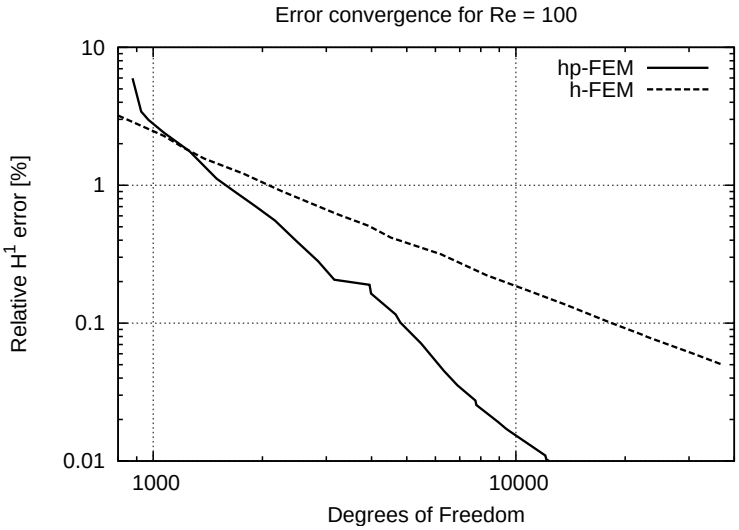



Figure 3.19: Convergence history of the $H^1$-errors in magnitude of velocity ($Re = 100$) - comparison of $hp$-adaptivity and $h$-adaptivity.

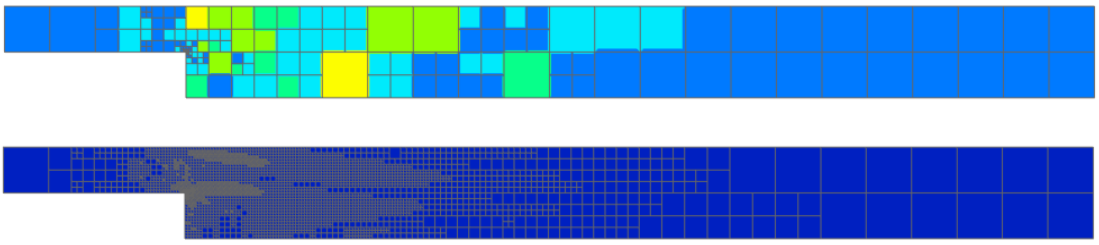


Figure 3.20: Optimal $hp$-mesh (6330 DOFs, error 0.045%) and $h$-mesh (37075 DOFs, error 0.050%) for steady state velocity field for $Re = 100$ using adaptivity guided by $H^1$-norm.

Fig. 3.21 shows convergence history of $hp$- and $h$-adaptivities for Reynolds number 800. Even though $hp$-adaptivity behaves worse at the beginning of the process, in asymptotic phase it converges much faster than standard $Q^2/Q^1$ elements. Final $hp$-meshes and for comparison final $h$-meshes for different values of Reynolds number ($Re = 200, 300, 400, 800$) are shown in Fig. 3.22 together with sizes of resulting systems. Notice exceedingly finer meshes (3 – 5 times more DOFs) resulting from $h$-adaptivity in comparison to $hp$-adaptive meshes, where DOFs are saved by using higher-order elements in areas where the solution is smooth. Fig. 3.23 shows distributions of the magnitude of the velocity $|\boldsymbol{u}|$ for different values of Reynolds number

($Re = 100 - 1000$) with increasing reattachment lengths. Dependence of computed reattachment lengths on Reynolds number and comparison with data from literature [16] is presented in Fig. 3.24. The flow field is separating and reattaching also on the upper wall of the channel (for higher Reynolds numbers) but we restricted ourselves in comparisons to the main reattachment length in the first recirculation region. Velocity profiles for $Re = 800$ at various downstream locations are depicted in Fig. 3.25. Fig. 3.26 compares our results with numerical study [16], where horizontal velocity $u_x$ and vertical velocity $u_y$ are plotted at various downstream locations, particularly for $x/c = 6, 14$ and 30 ($c$ denotes the channel height upstream of the step). Horizontal velocity $u_x$ agrees well at all locations, the vertical velocity $u_y$ agrees well at $x/c = 6$ and 30, while it differs a little at $x/c = 14$. It can be caused by insufficient length of the channel both upstream of the step and downstream of the step.

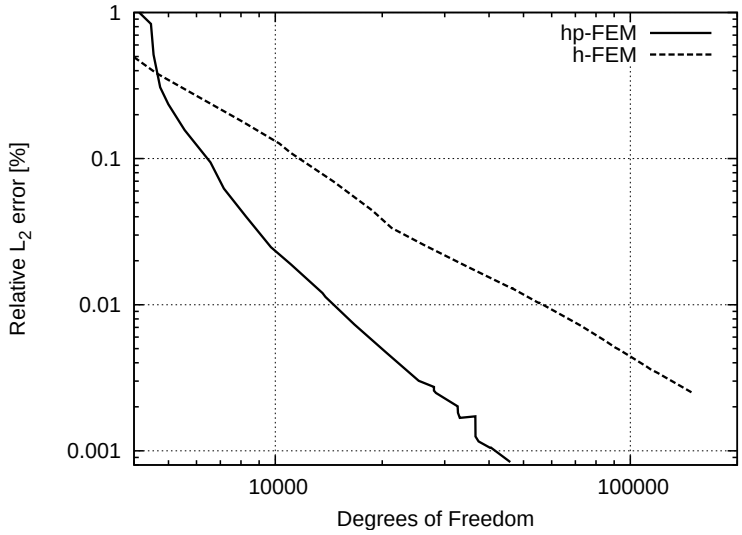
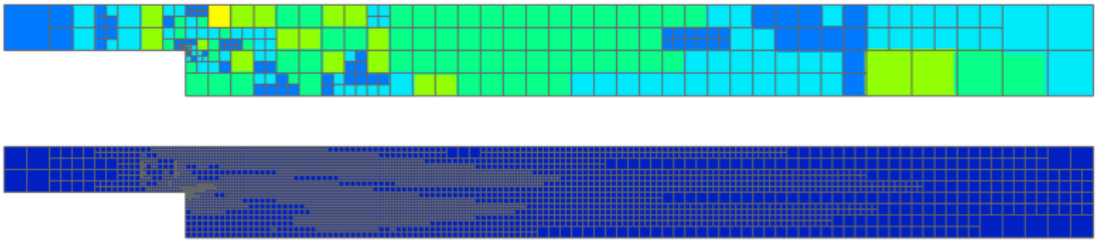
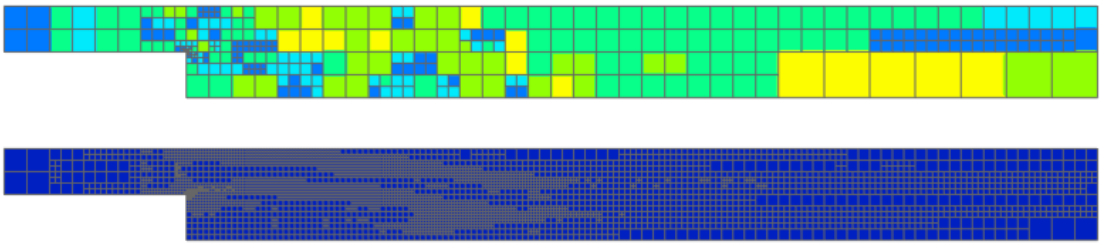


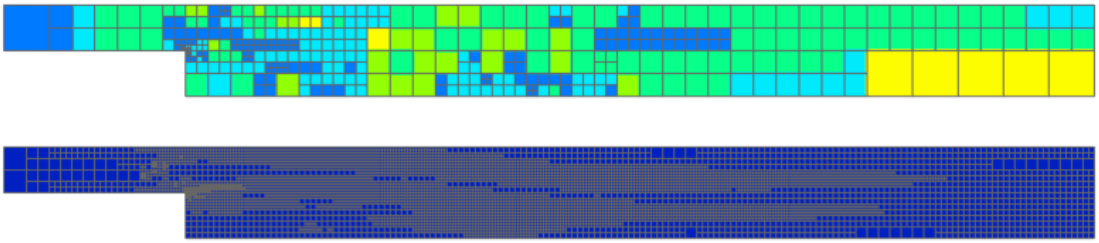

Figure 3.21: Convergence of relative $L^2$-errors of $|\boldsymbol{u}|$ for $Re = 800$.



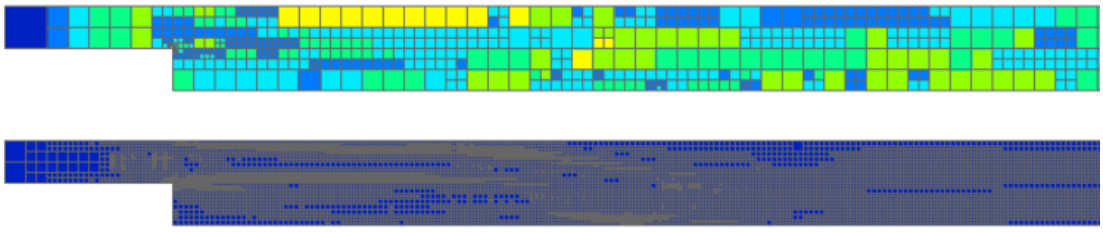
(a) $Re = 200$ ($hp$-mesh: 8929 DOFs, 0.004%, $h$-mesh: 35493 DOFs, 0.004%)



(b) $Re = 300$ ($hp$-mesh: 13226 DOFs, 0.003%, $h$-mesh: 46180 DOFs, 0.004%)



(c) $Re = 400$ ($hp$-mesh: 16522 DOFs, 0.003%, $h$-mesh: 56677 DOFs, 0.005%)



(d) $Re = 800$ (zoom) ($hp$-mesh: 28346 DOFs, 0.0025%, $h$-mesh: 150515 DOFs, 0.0025%)

Figure 3.22: Optimal $hp$-meshes and $h$-meshes for steady state velocity field for different values of Reynolds number. Error minimized in $L^2$-norm.

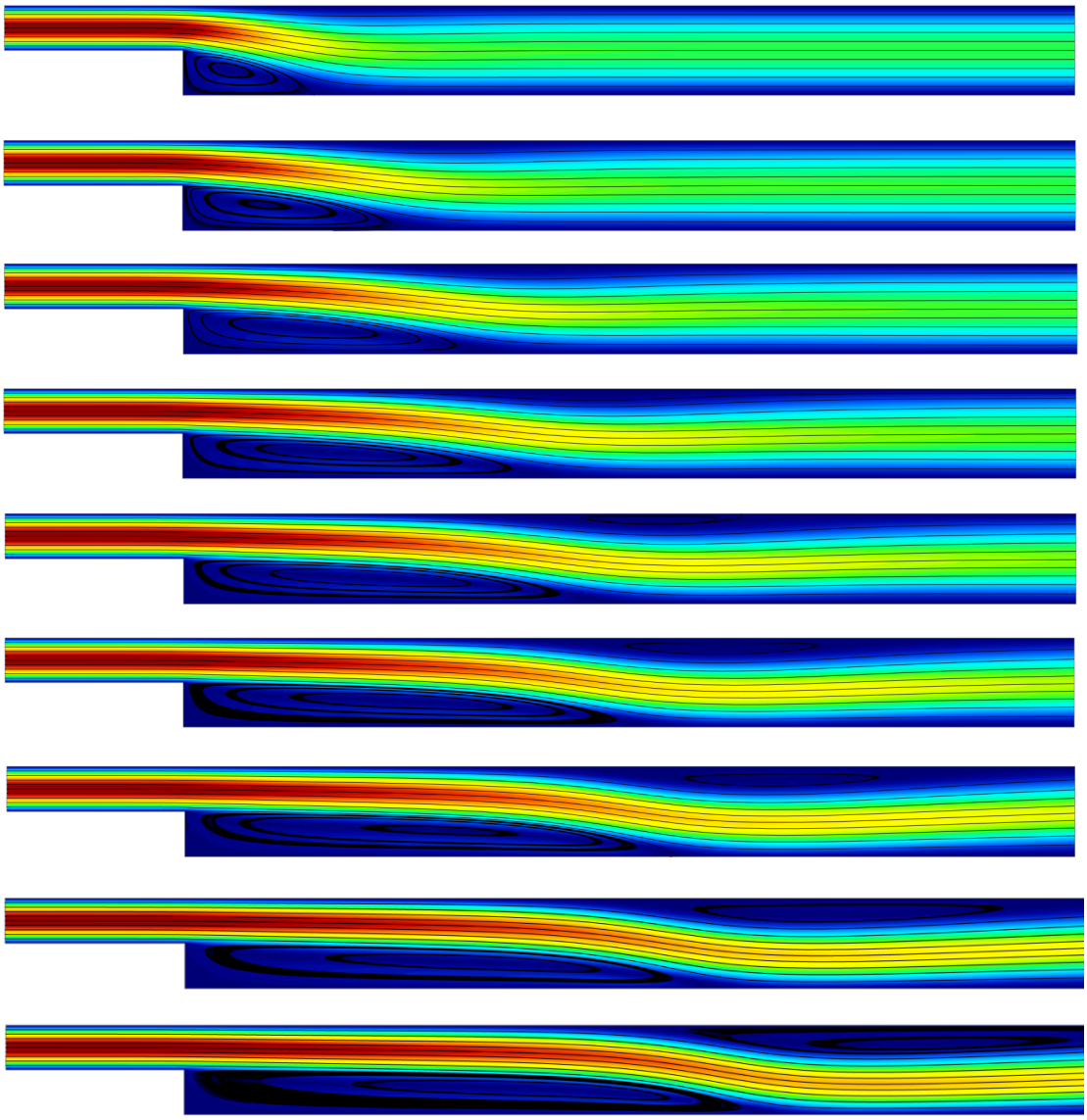


Figure 3.23: Distribution of the magnitude of velocity with streamlines for $Re = 100, 200, 300, 400, 500, 600, 700, 800, 1000$.

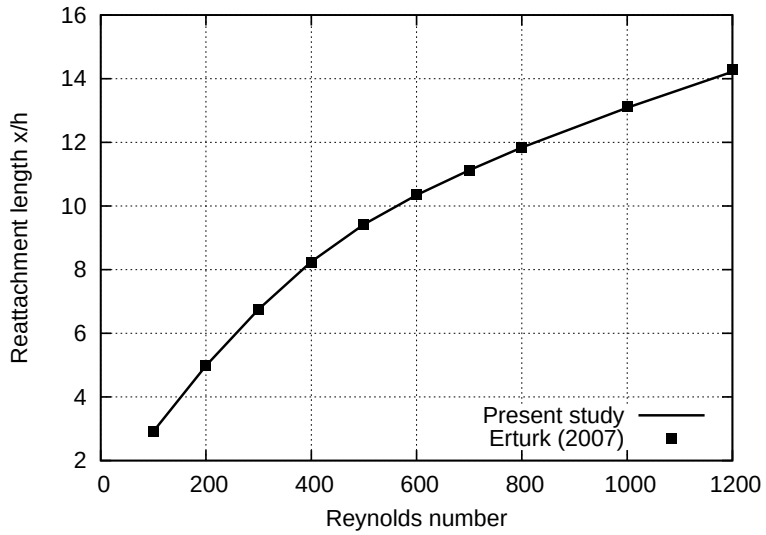



Figure 3.24: Dependence of reattachment length on Reynolds number.

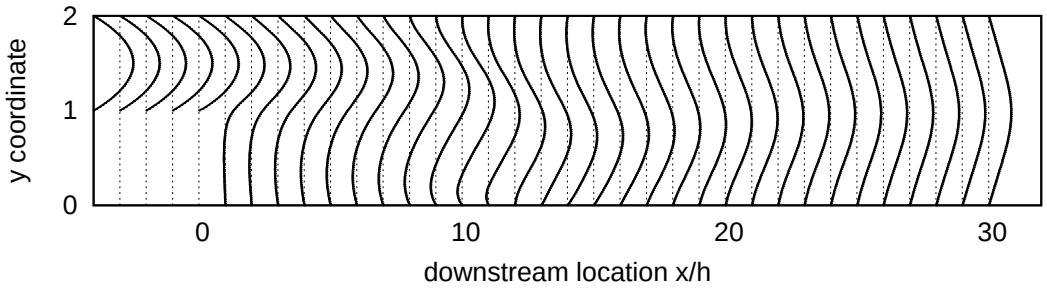



Figure 3.25: Velocity profiles at various downstream locations for $Re = 800$.

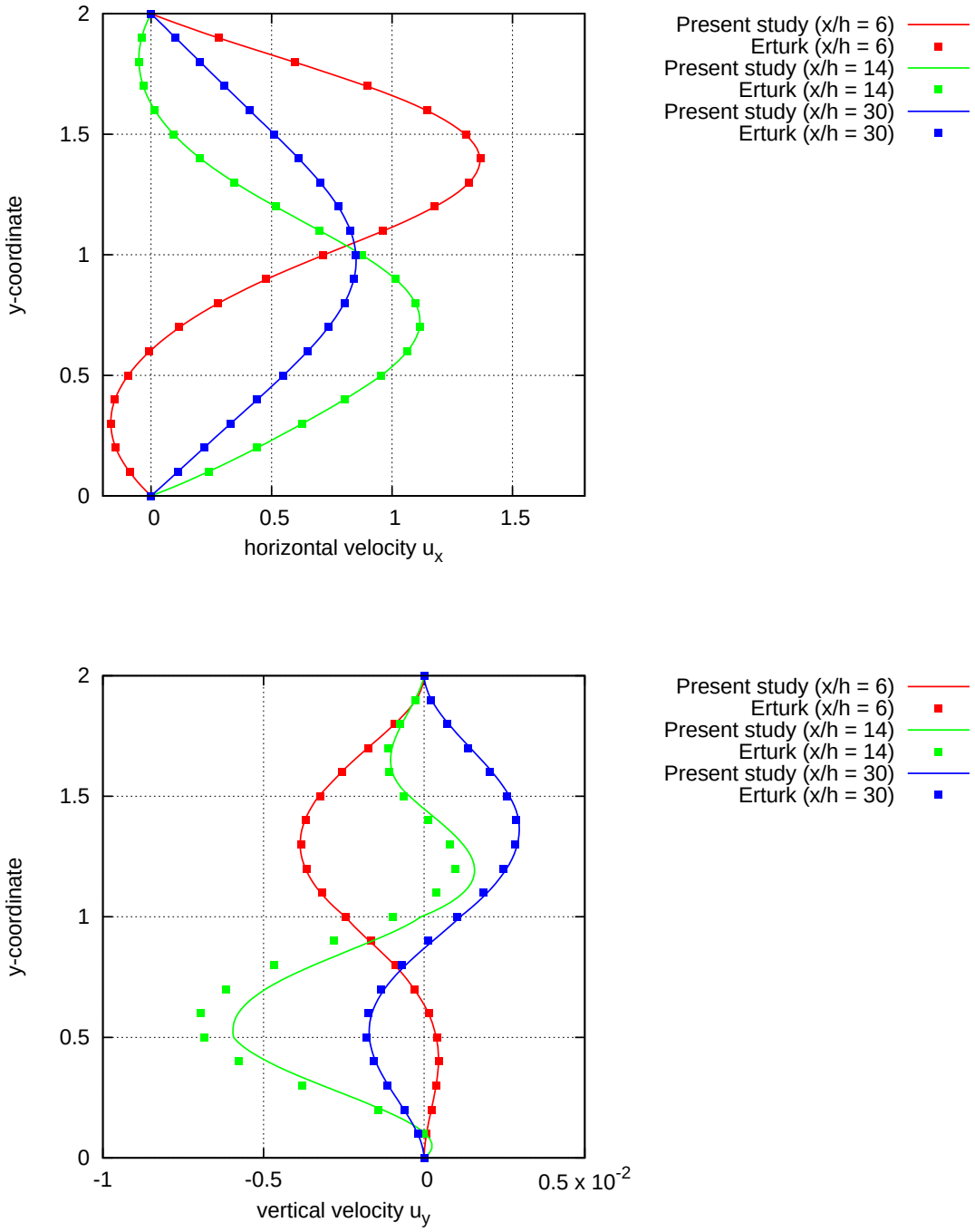



Figure 3.26: Comparison of components of velocity $u_x$, $u_y$ at various downstream locations for $Re = 800$.

## 3.4 Overview of open source adaptive $hp$-FEM softwares

In last decades, higher-order finite element methods became very popular mainly due to their capability of high convergence rates. With rapid progress in computational technologies (super computers, parallel programming, powerfull programming languages), $hp$-FEM is becoming more and more popular in both academic and commercial community. The main goal of $hp$-FEM lies in the optimal selection of $h$ and $p$ refinements, hence, the $hp$-adaptivity. There are several research teams working in the field of automatic $hp$-adaptivity. Let us mention the following open source projects:

- **hp90**[1] – $hp$-adaptive finite element package (written in FORTRAN) developed at Texas Institute for Computational and Applied Mathematics at UT Austin by the group of prof. L. Demkowicz. Their software supports one, two and three dimensional problems and $H^1$, $H$(curl) and $H$(div) conforming elements. It is capable of $h$, $p$ and $hp$ adaptivity based on reference solution and one level constraint meshes. For more information on the research group, see [11, 32, 34, 44].

- **Concepts**[2] – a set of classes (written in C++) for elliptic partial differential equations. It supports various methods – $hp$-FEM, DGFEM and BEM. Concepts is being developed by a group of PhD students of prof. Dr. Christoph Schwab at Seminar for Applied Mathematics at Swiss Federal Institute of Technology in Zurich. It supports meshes with hanging nodes and $hp$-adaptivity based on a priori knowledge of the solution. For more information on the project, see [20].

- **deal.II**[3] – a C++ library targeted at the computational solution of partial differential equations by adaptive finite elements. It was developed at the Numerical Methods Group at University of Heidelberg, Germany. Its main authors presently work at Texas A&M University in College Station and at Institute of Aerodynamics and Flow Technology of the German Aerospace Center (DLR) in Braunschweig. It supports one, two and three dimensional problems, handling locally refined meshes (one level constraints), different adaptive strategies based on local error indicators and error estimators. Both $h$- and $hp$-refinements are supported for both continuous and discontinuous elements. For more information on the project, see [5].

- **Hermes2D**[4] – a C++ library for solution of partial differential equations by adaptive $hp$-FEM developed by the group at University of Nevada at Reno and Institute of Thermomechanics in Prague. Developement of the software started at

[1] http://users.ices.utexas.edu/~leszek/2dhp90.html
[2] http://www.concepts.math.ethz.ch/
[3] http://www.dealii.org/
[4] http://hpfem.org/hermes2d/

University of Texas at El Paso by a group of students around prof. Pavel Šolín. Its main goal was solution of strongly coupled physical and engineering problems, where discretization of each physics is performed on an individual mesh adaptively obtained by the automatic $hp$-adaptivity. This goal was partially achieved also as a part of this thesis together with other PhD students (Jakub Červený, David Andrš and Pavel Kůs). Hermes2D and its three-dimensional version Hermes3D are capable of $h$-, $p$- and $hp$-adaptivity on meshes with arbitrary level hanging nodes, multiple meshes can be used in one computation allowing monolithic discretization of coupled problems where components exhibit qualitatively different behavior and allowing dynamically changing meshes for nonstationary problems. For more information on the research group, see [41, 43, 14, 47].

Chapter 4

# Coupling strategies for multiphysics PDE problems

Most physical and engineering processes are described as systems of PDEs and their numerical modeling usually requires simultaneous simulation of several physical fields in one computation. Such multiphysics problems are difficult to solve for several reasons:

1. In most cases these physics are strongly coupled in a nonlinear fashion. Either some decoupling techniques have to be employed resulting in reduced accuracy or one has to apply nonlinear methods, such as Newton's method to resolve the problems accurately.

2. The qualitative behavior of various physical fields may vary significantly, leading to the need for individual meshes for different solution components. In that case, either methods for transfer of required values from one mesh to another has to be used or an assembling over geometrically different meshes is necessary. If Newton's method is used, data-transfer methods are not an option anymore since the whole problem is solved as one system. Thus, in most models individual meshes are usually not possible and a mesh containing all necessary refinements has to be used.

3. Different components of the solution lie in different types of function spaces resulting in the need for different types of finite elements - such as piecewise continuous elements for elliptic problems, edge elements for electric field, H(div) elements for magnetic fields or discontinuous $L^2$ elements for pressure.

In this chapter we first review the most common decoupling techniques used for multiphysics problems and compare their performance on an example from fluid dynamics in Section 4.1. Section 4.2 is concerned with several data transfer methods for nonmatching meshes and their effect on the approximation error. In Section 4.3 monolithic

discretization of multiphysics problems via Newton's method and its Jacobian-free version is discussed. Our multimesh assembling technology allowing computation on multiple meshes is described in Section 4.4. The automatic adaptive algorithm, which was described in Chapter 3, is extended to coupled systems of PDEs in Section 4.5. The chapter is concluded in Section 4.6 by an example from civil engineering.

## 4.1 Operator-splitting methods

In conventional operator-splitting methods (or partition methods) for multiphysics problems the phenomenon is decoupled in order to simplify the solution of the problem, all physical fields are then computationally treated as isolated problems and can be separately solved both in space and time. Interactions between fields are then viewed as forcing terms and all necessary data are exchanged between models using prediction and/or substitution.

For illustration purposes, let us write down the following two-physics coupled problem

$$\begin{aligned} \frac{\partial u_1}{\partial t} &= a_{11}u_1 + a_{12}u_2 + F_1(u_1, u_2), \\ \frac{\partial u_2}{\partial t} &= a_{21}u_1 + a_{22}u_2 + F_2(u_1, u_2), \end{aligned}$$

where $F_i$ are strictly nonlinear operators. Two most common coupling schemes for two-physics problem are shown in Fig. 4.1 and 4.2. In Fig. 4.1 both physics are advanced in time simultaneously and then they exchange appropriate data. This approach easily allows parallel computing. The corresponding scheme using simple backward Euler scheme for time discretization is as follows

$$\begin{aligned} \frac{u_1^{n+1} - u_1^n}{\tau} &= a_{11}u_1^{n+1} + a_{12}u_2^n + F_1(u_1^n, u_2^n), \\ \frac{u_2^{n+1} - u_2^n}{\tau} &= a_{21}u_1^n + a_{22}u_2^{n+1} + F_2(u_1^n, u_2^n). \end{aligned}$$

In Fig. 4.2 physics are computed in strictly serial fashion, the result of one is used as data for the other. The numerical scheme of this approach goes as follows

$$\begin{aligned} \frac{u_1^{n+1} - u_1^n}{\tau} &= a_{11}u_1^{n+1} + a_{12}u_2^n + F_1(u_1^n, u_2^n), \\ \frac{u_2^{n+1} - u_2^n}{\tau} &= a_{21}u_1^{n+1} + a_{22}u_2^{n+1} + F_2(u_1^{n+1}, u_2^n). \end{aligned}$$

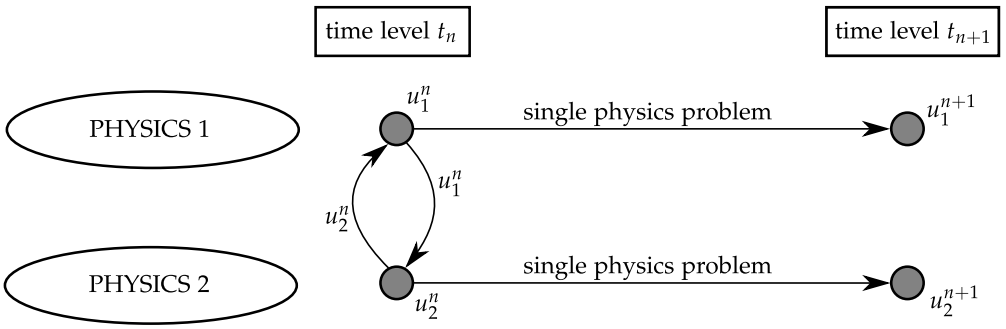



Figure 4.1: Operator splitting coupling technique - simultaneous advance.

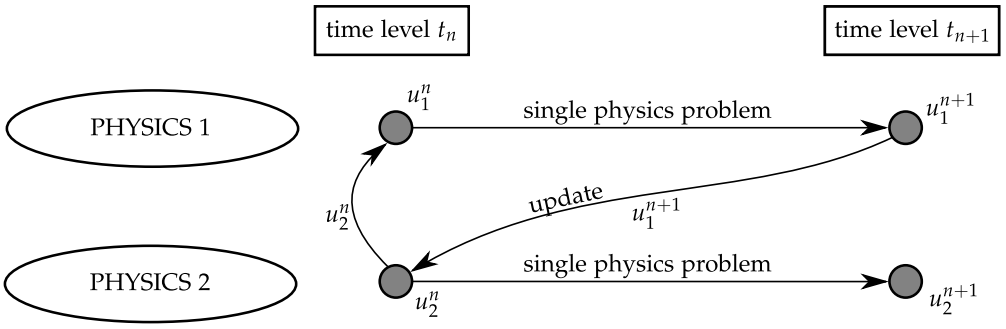



Figure 4.2: Operator splitting coupling technique - staggered update.

In both cases, single-physics softwares can be easily used with some data communication in between. This makes the approach popular. Furthermore, since the whole problem is divided, the resulting systems of equations are smaller, which makes them easier to solve by numerical solvers (both direct and iterative). But this advantage is not cost free. Traditional operator-splitting schemes can cause degradation in both stability and accuracy [35]. They are known to be only first order accurate in time regardless of the time discretization technique chosen, which makes them unsuitable for higher-order time integration schemes. In order to achieve desired accuracy, very small time steps have to be used, therefore increasing computational time.

To improve traditional splitting schemes one can use higher-order linearization for treatment of nonlinear terms. Basic idea is to predict solution values on the next time level using computed values from the previous levels, where the predictor is derived to be of the same order of accuracy as the time integration scheme used. The first-order and the second-order formula for predictor follow, resulting coupling scheme is in Fig. 4.3.

$$\begin{aligned} u^{n+1} &\approx u^n, \\ u^{n+1} &\approx 2u^n - u^{n-1}. \end{aligned}$$

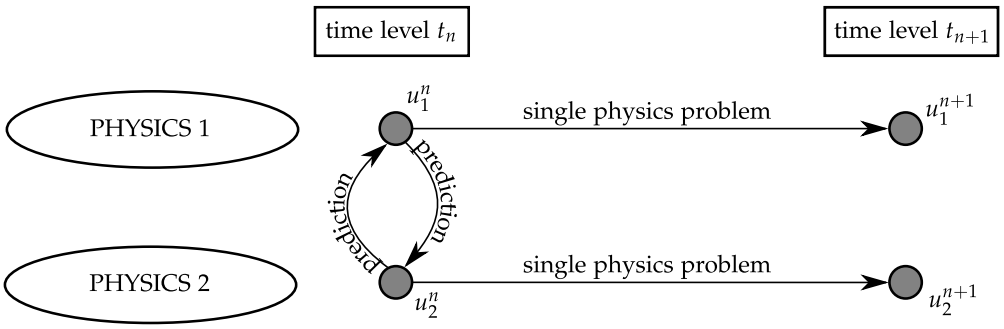



Figure 4.3: Operator splitting coupling technique - predicted solutions.

Alternatively, one can solve the nonlinear coupled system using fixed-point iterations (i.e. *Picard iterations*). In this process, on each time level several iterations are performed until global convergence of all physics is achieved (see Fig. 4.4). This procedure can restore the accuracy of time integration scheme used, but in case the problem is tightly coupled the iterations usually converge very slowly if at all. Also from the point of view of the automatic adaptivity, fixed-point iterations may not converge on initial very coarse meshes.

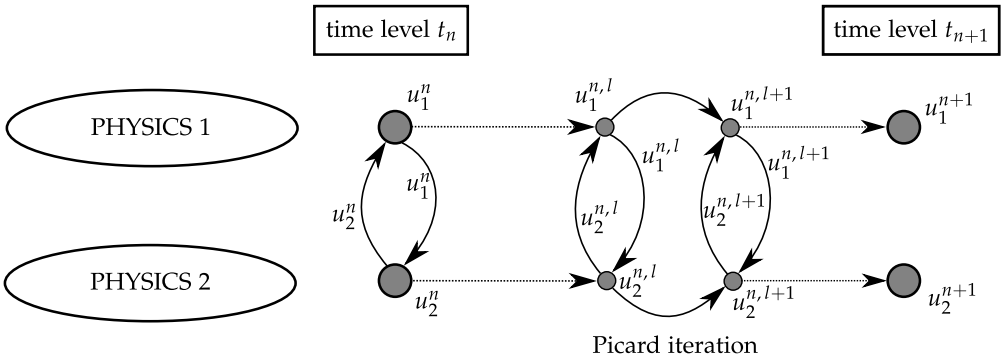



Figure 4.4: Operator splitting coupling technique - fixed-point iterations.

Described loss of accuracy of conventional operator splitting schemes is demonstrated on an example of incompressible flow over a backward facing step. Similar comparisons are also shown later in this chapter. On the example of the flame propagation problem we will see that even the fixed-point iterations may not recover the expected accuracy and on the heat and moisture problem we will show that even for linear problems simple linearization reduces the order of convergence.

**Example 4.1** *Let us recall Example 3.3.2 from Chapter 3, where the incompressible Navier-stokes equations were solved in the domain with a backward facing step with Reynolds number* $Re = 200$*. Time derivative is discretized by the second-order backward difference formula and we are interested in the solution after* 5 *seconds (depicted in Fig. 4.5).*

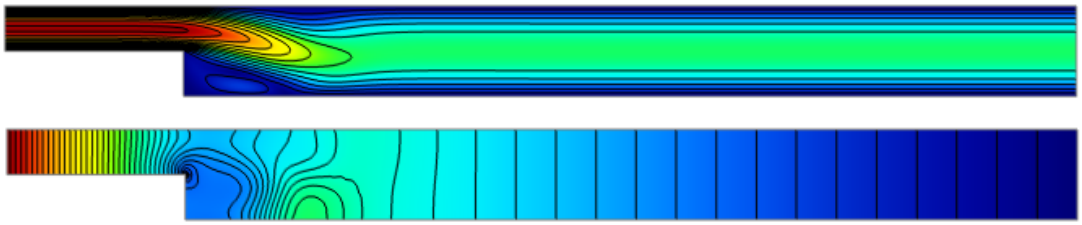


Figure 4.5: The magnitude of velocity and the pressure at $t = 5$s.

*Fig. 4.6 shows the convergence of the velocity magnitude in $L^2$-norm in time for two different treatments of the nonlinear convective term $(\boldsymbol{u} \cdot \nabla)\,\boldsymbol{u}$ – simple linearization and fixed-point iterations. For all computations the same space discretization was used and shown errors are with respect to the "overkill" reference solution. As was said before, traditional operator-splitting with a simple linearization reduces the order of convergence to one, while fixed-point iterations almost recovered the expected second-order convergence. Due to a singularity in the velocity at the step, the error in time can be influenced by the insufficient spatial discretization.*

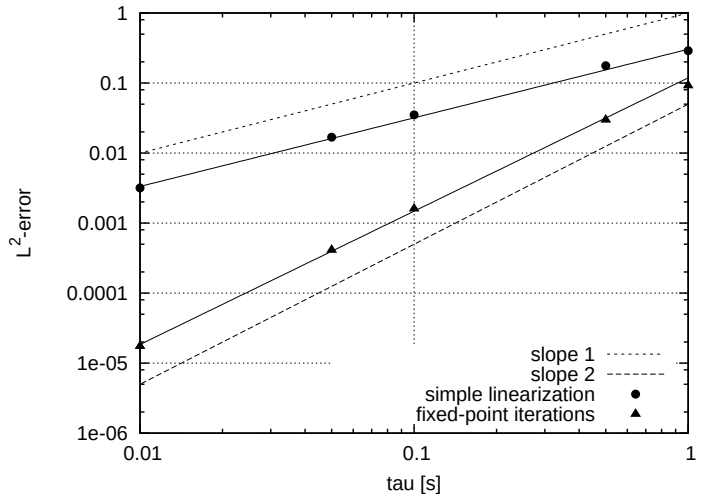



Figure 4.6: Order of convergence for different OS schemes for Navier-stokes equations.

## 4.2 Data transfer between non-matching meshes

In operator splitting methods, when each physics component is solved on its own mesh (or even its own geometric domain), simulation data (such as heat source, reaction rate, fussion source or various nonlinear parameters) must be exchanged between these meshes. The meshes can be generally non-matching, but in order to compare

several data transfer methods with our approach (to be described in Section 4.4) we limit ourselves in examples to meshes which share a very coarse *master mesh* $\mathcal{T}_m$, but are obtained by independent refinement processes.

In the following paragraphs we present a brief overview of the most commonly used data-transfer methods in multiphysics simulations. Our goal is not to conduct an extensive survey but to demonstrate that even very accurate methods can fail on some problems, and later to compare these methods with our approach described in Section 4.4.

In the higher-order finite element method, the solution associated with a mesh is defined as a linear combination of higher-order basis functions associated with vertices, edges and element interiors. Let us denote the finite dimensional subspace of $H^1(\Omega)$ connected with a mesh $\mathcal{T}_A$ by $V_{hp}^A$ and the finite dimensional subspace connected with a mesh $\mathcal{T}_B$ by $V_{hp}^B$. Let us assume that function $u_{hp} = \sum_{i=1}^{N} c_i^A v_i^A$, where $v_i^A$, $i = 1, \dots N$, form a basis of the space $V_{hp}^A$, has to be transferred to the mesh $\mathcal{T}_B$. Thus, we are looking for a new vector of coefficients $\{c_i^B\}_1^M$, such that $\tilde{u}_{hp} \approx u_{hp}$, where

$$\tilde{u}_{hp} = \sum_{i=1}^{M} c_i^B v_i^B, \quad v_i^B \in V_{hp}^B = \langle v_1^B, \dots, v_M^B \rangle. \tag{4.1}$$

We will mention here three most common data transfer methods. For more options and more extensive comparisons see, e.g., [28].

### 4.2.1 Linear interpolation

For nodal linear finite elements, the linear interpolation is probably the most popular and the most widely used method for transferring data between meshes. By linear interpolation we mean construction of a linear interpolant $\tilde{u}_{hp}$ that evaluates source function $u_{hp}$ exactly at vertices of mesh $\mathcal{T}_B$. Therefore, coefficients $\{c_i^B\}_1^M$ are either equal to $c_j^A$ in case the vertices coincide (values of the source function at vertices are equal to coefficients in nodal FEM) or obtained by 1D linear interpolation between coefficients $\{c_i^A\}_1^N$. It is clear that in areas where the source mesh $\mathcal{T}_A$ is finer than $\mathcal{T}_B$, the transfer causes error and in case of multiple transfers between meshes it can lead to the loss of accuracy.

### 4.2.2 Projection-based interpolation

For the higher-order hierarchic finite elements, the natural extension of the linear interpolation would be a combination of the lagrange interpolation and projection (so called *projection-based interpolation* [48, 10]). In hierarchic FEM, coefficients in the linear combination (4.1) are not associated with function values at particular points (as

in nodal FEM) but with general basis functions. In the projection-based interpolation the interpolant is constructed as a sum of vertex, edge and bubble interpolants:

$$\tilde{u}_{hp} = u^v + u^e + u^b,$$

where interpolants $u^v$, $u^e$ and $u^b$ are linear combinations of vertex, edge and bubble basis functions, respectively. The accuracy of interpolation is controlled by the polynomial orders of the basis functions on the mesh $\mathcal{T}_B$.

**The vertex interpolant** matches the source function $u_{hp}$ at all mesh vertices and corresponds to the linear interpolation described in the previous paragraph,

$$u^v = \sum_{i=1}^{n} c_i^v v_i^{B,v}, \tag{4.2}$$

where $n$ is the number of mesh vertices, $v_i^{B,v}$ are vertex functions in the space $V_{hp}^B$ and the coefficients $c_i^v$ are equal to the values of $u_{hp}$ at the vertices of $\mathcal{T}_B$. Fig. 4.7 shows the source function, vertex interpolant and the residual $u_{hp} - u^v$ for a single element mesh.

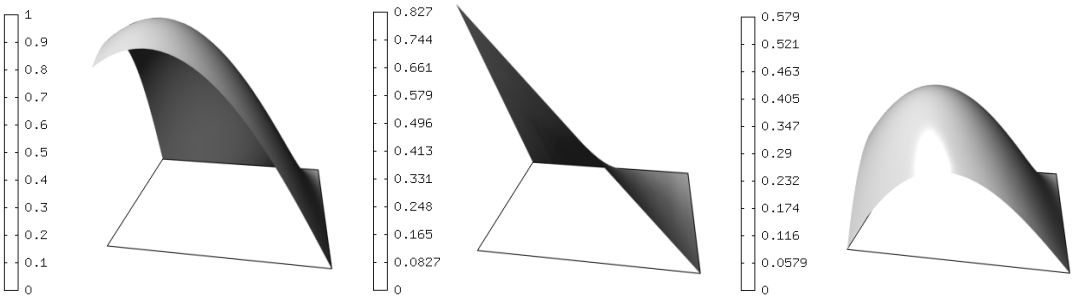



Figure 4.7: Source function, vertex interpolant $u^v$ and the residual $u_{hp} - u^v$.

**The edge interpolant** is constructed as a sum of edge interpolants for each mesh edge.

$$u^e = \sum_{j=1}^{m} u^{e_j},$$

where $m$ is the number of edges in $\mathcal{T}_B$. The distance between the residual $u_{hp} - u^v$ and the edge interpolant $u^e$ should be minimized on all edges in $\mathcal{T}_B$. Therefore, we solve the following minimization problems

$$||(u_{hp} - u^v) - u^{e_j}||_{H_0^1(e_j)} \to \min, \quad j = 1, \ldots, m. \tag{4.3}$$

Each of these problems can be transformed into a system of $p^{e_j} - 1$ linear equations for unknown coefficients $c_i^{e_j}$, $i = 2, \ldots, p^{e_j}$, where $p^{e_j}$ denotes the polynomial order equipped with the edge $e_j$,

$$\sum_{i=2}^{p^{e_j}} c_i^{e_j} (v_i^{B,e_j},\ v_k^{B,e_j})_{H_0^1(e_j)} = (u_{hp} - u^v,\ v_k^{B,e_j})_{H_0^1(e_j)} \quad k = 2, 3, \ldots, p^{e_j}, \tag{4.4}$$

where the edge interpolant $u^{e_j}$ is written as a linear combination of the edge functions $v_i^{B,e_j} \in V_{hp}^B$, $i = 2, 3, \ldots, p^{e_j}$,

$$u^{e_j} = \sum_{i=2}^{p^{e_j}} c_i^{e_j} v_i^{B,e_j}.$$

Edge interpolant and resulting residual $u_{hp} - u^v - u^e$ are depicted in Fig. 4.8.

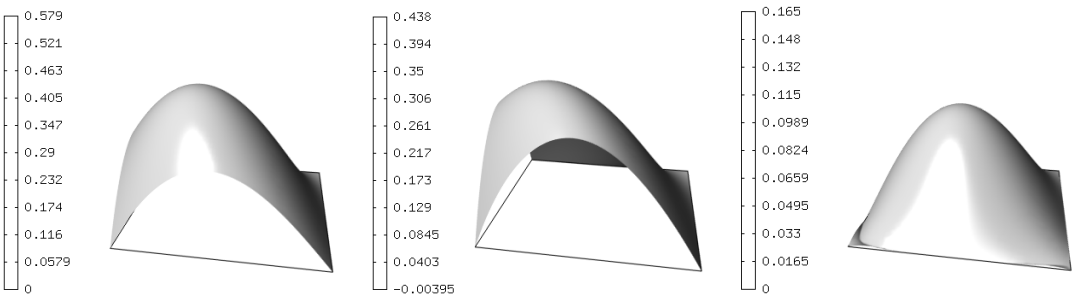



Figure 4.8: Residual $u_{hp} - u^v$, edge interpolant $u^e$ and residual $u_{hp} - u^v - u^e$.

**The bubble interpolant** is constructed as a sum of bubble interpolants for each element $K_j$

$$u^b = \sum_{j=1}^{l} u^{b_j},$$

where $l$ denotes the number of elements of $\mathcal{T}_B$. Local bubble interpolants $u^{b_j}$ are obtained by projecting the residual $u_{hp} - u^v - u^e$ onto the polynomial space generated by the bubble functions $v_i^{B,b_j}$, $i = 1, \ldots, l_{p^j}$, where $p^{b_j}$ is the polynomial order associated with element $K_j$ and $l_{p^j}$ denotes the number of bubble functions for polynomial degree $p^{b_j}$. The corresponding minimization problems read

$$|(u_{hp} - u^v - u^e) - u^{b_j}|_{H^1(K_j)} \to \min, \tag{4.5}$$

where the bubble interpolants are written as a linear combination of bubble functions

$$u^{b_j} = \sum_{i=1}^{l_{p^j}} c_i^{b_j} v_i^{B,b_j}.$$

Therefore, we can rewrite minimization problem (4.5) as a system of algebraic equations

$$\sum_{i=1}^{l_{p^j}} c_i^{b_j} \int_{K_j} \nabla v_i^{B,b_j} \cdot \nabla v_k^{B,b_j} \, \mathrm{d}\boldsymbol{x} = \int_{K_j} \nabla (u_{hp} - u^v - u^e) \cdot \nabla v_k^{B,b_j} \, \mathrm{d}\boldsymbol{x}, \quad k = 1, 2, \ldots, l_{p^j} \quad (4.6)$$

for unknown coefficients $\{c_i^{b_j}\}_1^{l_{p^j}}$. This is illustrated in Fig. 4.9, where the final error of our interpolation is displayed on the very right.

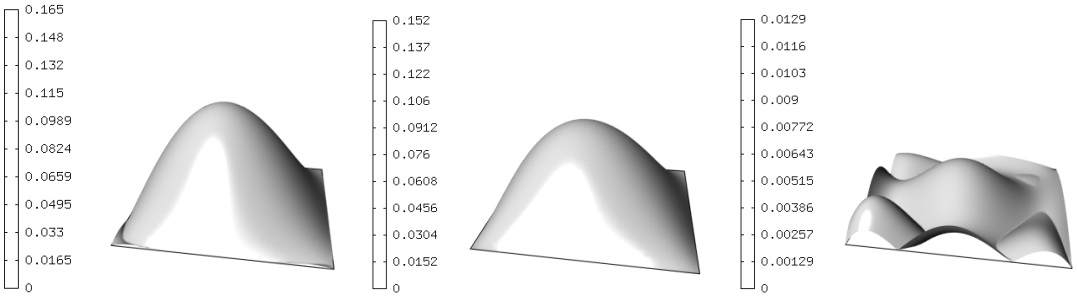



Figure 4.9: Residual $u_{hp} - u^v - u^e$, bubble interpolant $u^b$ and residual $u_{hp} - u^v - u^e - u^b$.

**Remark 4.1** *Notice that in Fig. 4.8 on the very right the residual $u_{hp} - \tilde{u}_{hp}$ vanishes at all four vertices, but it does not vanish on the edges and in the element interiors. The reason is that meshes $\mathcal{T}_A$ and $\mathcal{T}_B$ are geometrically different and their elements are generally equipped with different polynomial orders.*

**Remark 4.2** *Hierarchic finite elements are much less suitable for interpolation between meshes than nodal elements are. In nodal approach coefficients in linear combination are equal to function values at particular points (vertices, edge midpoints or element centers) and thus the evaluation of functions at these points is fast. On the other hand, in hierarchic finite elements to evaluate function at particular point one has to locate element in which the point lies and then evaluate all basis functions at that point, which is time consuming.*

*Instead one can write the source function in terms of basis functions on the mesh $\mathcal{T}_A$ and*

*rewrite linear system (4.4) as follows (system (4.6) could be rewritten in a similar way)*

$$\sum_{i=2}^{p^{e_j}} c_i^{e_j} (v_i^{B,e_j}, v_k^{B,e_j})_{H_0^1(e_j)} = \sum_{i=1}^{N} c_i^A (v_i^A, v_k^{B,e_j})_{H_0^1(e_j)} - \sum_{i=1}^{n} c_i^v (v_i^v, v_k^{B,e_j})_{H_0^1(e_j)}. \tag{4.7}$$

*On the other hand this leaves us with the problem of integrating basis function $v_i^A$ and $v_k^{B,e_j}$ which are defined on different meshes.*

### 4.2.3 $L^2$ minimization

This method is based on minimizing the error $u_{hp} - \tilde{u_{hp}}$ globally in $L^2$-norm. Source function $u_{hp}$ is defined on $\mathcal{T}_A$ and its sought approximation $\tilde{u}_{hp}$ on the mesh $\mathcal{T}_B$. Thus,

$$||u_{hp} - \tilde{u}_{hp}||_{L^2} \to 0$$

can be written in the weak sense as follows

$$(u_{hp} - \tilde{u}_{hp},\, v)_{L^2} = 0, \qquad \text{for any function } v \in V_{hp}^B.$$

This results into two integrals, where $\int_\Omega \tilde{u}_{hp}\, v \, \mathrm{d}x$ is computed over elements of $\mathcal{T}_B$ and $\int_\Omega u_{hp}\, v \, \mathrm{d}x$ is computed over elements of so-called union mesh, which contains refinements from both meshes. Similar approach can be found in [28]. Details on numerical integration of functions defined on different meshes will be discussed in Section 4.4.

### 4.2.4 Demonstrative example

In this paragraph we use a problem with known exact solution to compare the described methods. Let us solve a heat transfer equation with a spatially-dependent thermal conductivity $k = k(x, y)$,

$$\begin{aligned} -\nabla \cdot (k\, \nabla u) &= f \qquad \text{in } \Omega = (0.01, 1.0)^2, \\ u &= u_D \quad \text{on } \partial\Omega. \end{aligned} \tag{4.8}$$

With

$$k(x,y) = x^3 + y^3, \qquad f(x,y) = 2 - 2\left(\frac{y^3}{x^3} + \frac{x^3}{y^3}\right) \qquad \text{and} \qquad u_D(x,y) = \frac{1}{x} + \frac{1}{y},$$

the exact solution is given by

$$u(x,y) = \frac{1}{x} + \frac{1}{y}.$$

The weak formulation of the problem reads: Find $u \in H^1(\Omega)$, such that

$$\int_\Omega k\, \nabla u \cdot \nabla v \, \mathrm{d}x = \int_\Omega f\, v \, \mathrm{d}x, \qquad \text{for all } v \in H_0^1(\Omega). \tag{4.9}$$

For comparison purposes, two different meshes $\mathcal{T}_\mathcal{A}$ and $\mathcal{T}_\mathcal{B}$ were generated as follows: A mesh $\mathcal{T}_A$ is a uniform piecewise-linear $256 \times 256$ subdivision of the computational domain containing 65025 DOF, as shown in the left part of Fig. 4.10. A piecewise-linear mesh $\mathcal{T}_\mathcal{B}$ is obtained by solving equation (4.9) using adaptive $h$-FEM. This mesh contains 189 DOF and it is virtually optimal to represent the exact solution $u$ (see right part of Fig. 4.10). The thermal conductivity $k(x, y)$ is represented on the mesh $\mathcal{T}_A$ via its continuous, piecewise-linear vertex interpolant $k_h(x, y)$.

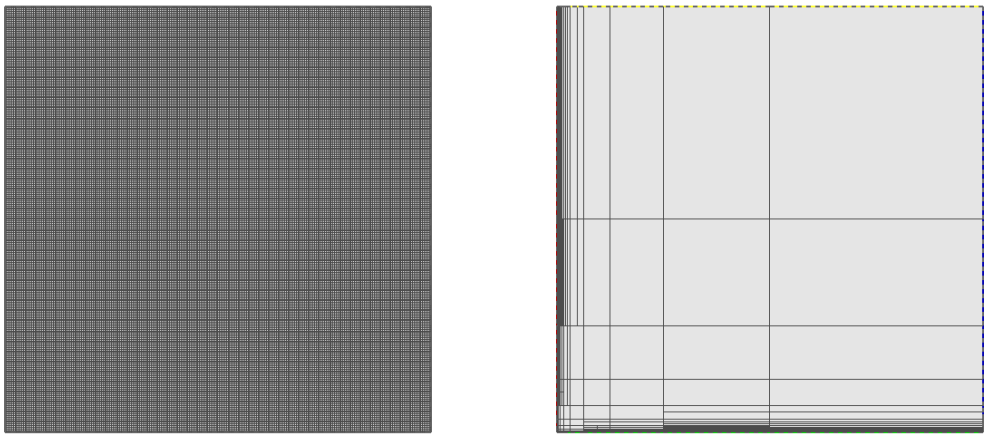


Figure 4.10: Left: uniform piecewise-linear mesh $\mathcal{T}_A$ with 65025 DOF where the thermal conductivity $k(x, y)$ is represented via its continuous, piecewise-linear vertex interpolant $k_h(x, y)$. Right: non-uniform piecewise-linear mesh $\mathcal{T}_A$ with 189 DOF where the solution $u$ of equation (4.9) is sought.

In the following we solve equation (4.9) in three different ways and compare the results to the exact solution $u$:

**(1) Transferring $k_h(x, y)$ from $\mathcal{T}_A$ to $\mathcal{T}_B$ via interpolation and solving on $\mathcal{T}_B$**

Let us begin with linear interpolation of vertex values, which probably is the most widely used method for transferring solution data between different meshes. The piecewise-linear function $k_h(x, y)$ defined on the mesh $\mathcal{T}_A$ is approximated on the mesh $\mathcal{T}_B$ by means of a continuous function $k_h^{(1)}(x, y)$ that matches $k_h(x, y)$ exactly at all grid vertices of the mesh $\mathcal{T}_B$ and is bilinear in each element of $\mathcal{T}_B$. The corresponding approximation $u_h^{(1)}$, computed on the mesh $\mathcal{T}_B$ using linear finite elements

and the approximate thermal conductivity $k_h^{(1)}$, is shown in part (a) of Fig. 4.11. The approximation error $e_h^{(1)}(x,y) = u(x,y) - u_h^{(1)}(x,y)$ is shown in part (a) of Fig. 4.12.

**(2) Transferring $k_h(x,y)$ from $\mathcal{T}_A$ to $\mathcal{T}_B$ via $L^2$-projection and solving on $\mathcal{T}_B$**

The approximation of thermal conductivity $k_h^{(2)}$ is obtained from the projection problem $||k_h^{(2)} - k_h||_{L^2} \rightarrow 0$. The corresponding approximation $u_h^{(2)}$, computed on the mesh $\mathcal{T}_B$ using linear finite elements and the approximate thermal conductivity $k_h^{(2)}$, is shown in part (b) of Fig. 4.11. The approximation error $e_h^{(2)}(x,y) = u(x,y) - u_h^{(2)}(x,y)$ is shown in part (b) of Fig. 4.12.

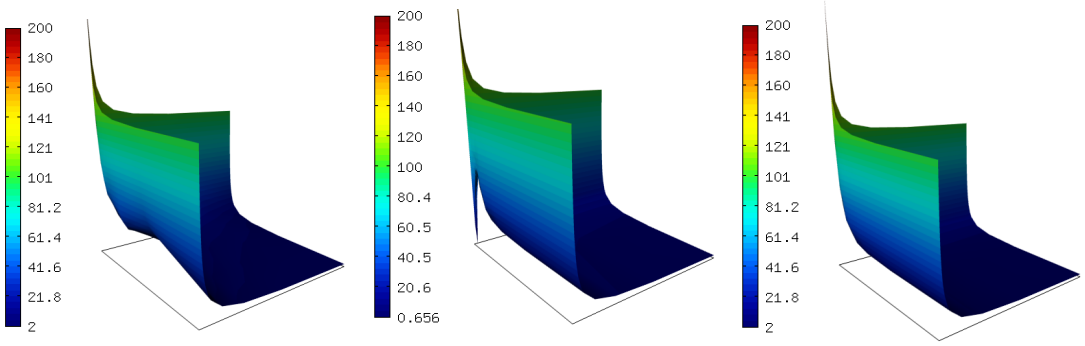
**(3) Solving on $\mathcal{T}_B$ with the exact thermal conductivity $k(x,y)$**

For comparison purposes we also compute a piecewise-linear approximation $u_h^{(3)}$ on the mesh $\mathcal{T}_B$ using the original function $k(x,y)$. This is the best finite element approximation one can obtain on the mesh $\mathcal{T}_B$. The result is shown in part (c) of Fig. 4.11 and the corresponding approximation error $e_h^{(3)}(x,y) = u(x,y) - u_h^{(3)}(x,y)$ is shown in part (c) of Fig. 4.12.

The results of the three computations are summarized in Tab. 4.1 which shows $H^1$-norms of the approximation errors on the mesh $\mathcal{T}_B$ calculated with respect to the exact solution $u$. Approaches (1) and (2) resulted in a large error in the solution. In particular, notice the extreme error in case (2). As discussed in [28], one must typically use a higher order integration on the target mesh. Thus, a second calculation was performed using quadratic elements on $\mathcal{T}_B$ to yield the third row of Tab. 4.1.

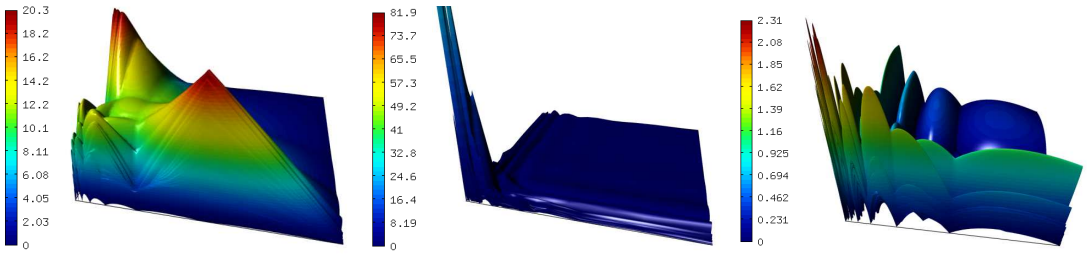
Table 4.1: Comparison of relative $H^1$-norm errors for the three methods described above.

| **Approach** | **Relative $H^1$-error** |
|---|---|
| Interpolation $k_h^{(1)}$ | 17.766% |
| $L^2$-projection $k_h^{(2)}$ (lin.) | 31.808% |
| $L^2$-projection $k_h^{(2)}$ (quadr.) | 6.734% |
| Using exact data $k(x,y)$ | 6.729% |




(a) Approximation $u_h^{(1)}$ (b) Approximation $u_h^{(2)}$ (c) Approximation $u_h^{(3)}$

Figure 4.11: Finite element solutions to (4.9). Note that the both data-transfer methods possess significant error close to the origin.




(a) Error $e_h^{(1)}$ (b) Error $e_h^{(2)}$ (c) Error $e_h^{(3)}$

Figure 4.12: Approximation errors on the mesh $\mathcal{T}_B$ with respect to the exact solution $u$. Note that the scale of the error magnitude is significantly different between the plots.

## 4.3 Monolithic discretization of nonlinear coupled problems

As described above the conventional operator-splitting methods are flexible in the way how different physics are solved but contain inconsistency and instability since coupled (nonlinear) terms are not dealt accurately. An alternative lies in solving for all variables simultaneously by formulating a large system of nonlinear algebraic equations on each time level

$$F_i(\boldsymbol{Y}) = 0, \qquad i = 1, \ldots, N. \tag{4.10}$$

Discrete problem (4.10), written in the form of residual functions, was obtained by applying an implicit time discretization scheme and by deriving the weak formulation of the specific set of PDEs. In (4.10), $F_i : \mathbb{R}^N \to \mathbb{R}$ are nonlinear operators, $\boldsymbol{Y}$ is the vector of $N$ unknowns, which consists of the spatial unknowns for all physics involved in the problem. Thus, in the finite element method the vector of unknowns $\boldsymbol{Y}$ consists

of several sets of coefficients corresponding to the basis functions

$$u_1 = \sum_{i=1}^{N_1} y_i^1 v_i^1, \quad u_2 = \sum_{i=1}^{N_2} y_i^2 v_i^2, \quad \ldots, \quad u_r = \sum_{i=1}^{N_r} y_i^r v_i^r,$$

$$\boldsymbol{Y} = (y_1^1, \ldots, y_{N_1}^1,\ y_1^2, \ldots, y_{N_2}^2, \ldots,\ y_1^r, \ldots, y_{N_r}^r),$$

where $u_j$, $j = 1, \ldots, r$ are particular physical fields, $\{v_i^j\}_{i=1}^{N_j}$ are the basis functions associated with the solution component $u_j$, and $r$ is the number of physics involved in the problem. Thus, the Jacobian of the system (4.10), $\mathcal{J} : \mathbb{R}^N \rightarrow \mathbb{R}^{N \times N}$, is a sparse matrix

$$\mathcal{J}_{kl}(\boldsymbol{Y}) = \frac{\partial F_k(\boldsymbol{Y})}{\partial (\boldsymbol{Y})_l}. \tag{4.11}$$

One iteration of Newton's method can be expressed in the following form

$$\boldsymbol{Y}^{k+1} = \boldsymbol{Y}^k - \mathcal{J}^{-1}(\boldsymbol{Y}^k)\ \boldsymbol{F}(\boldsymbol{Y}^k),$$

or

$$\mathcal{J}(\boldsymbol{Y}^k)\delta\boldsymbol{Y}^{k+1} = -\boldsymbol{F}(\boldsymbol{Y}^k), \tag{4.12}$$

$$\boldsymbol{Y}^{k+1} = \boldsymbol{Y}^k + \delta\boldsymbol{Y}^{k+1}. \tag{4.13}$$

With this single-block approach to the solution of nonlinear coupled problems all nonlinearities inherent to the problem are resolved accurately. Nevertheless, the main drawbacks of the Newton's method are the necessity to formulate the Jacobian in analytical terms and the need to actually compute and store the matrix itself. Indeed, when the number of variables increases, the matrix containing Jacobian entries grows, and in case of strongly coupled problems it can get substantially dense, which results in greater memory requirements. Moreover, for strongly coupled and strongly nonlinear problems difficulty with formulating the entries of the Jacobian matrix mathematically appears. A remedy to these drawbacks can be a Jacobian-free Newton-Krylov method (JFNK) [29, 21]. The Krylov iterative methods (such as GMRES) for solving linear systems like (4.12) typically require only matrix-vector product $\mathcal{J} \cdot \boldsymbol{v}$ rather than the full matrix. In the Jacobian-free Newton-Krylov method the matrix-vector product is approximated by a finite difference formula, which results into

$$\mathcal{J}(\boldsymbol{Y}^k) \cdot \boldsymbol{v} = \frac{\partial \boldsymbol{F}(\boldsymbol{Y}^k)}{\partial \boldsymbol{Y}^k} \cdot v \approx \frac{\boldsymbol{F}(\boldsymbol{Y}^k + \epsilon \boldsymbol{v}) - \boldsymbol{F}(\boldsymbol{Y}^k)}{\epsilon}, \tag{4.14}$$

where $\epsilon$ is a small perturbation [29]. It is clear from equation (4.14) that the Jacobian $\mathcal{J}$ does not have to be constructed at all, and that just the residual function $\boldsymbol{F}$ is evaluated with different inputs. The recent research [29, 30] shows that the JFNK can be used to solve nonlinear coupled problems more efficiently than the standard

Newton's method.

With Jacobian-free Newton-Krylov method the main computational cost lies in the iterative solvers since the assembling of the Jacobian is not necessary. The crucial problem with iterative solvers is an efficient preconditioning of the linear system arising in the Newton's method. We may solve the right preconditioned system

$$\mathcal{J}(\boldsymbol{Y}^k)P^{-1}(P\delta\boldsymbol{Y}^{k+1}) = -\boldsymbol{F}(\boldsymbol{Y}^k) \tag{4.15}$$

via two systems

$$\mathcal{J}(\boldsymbol{Y}^k)P^{-1}(\boldsymbol{X}) = -\boldsymbol{F}(\boldsymbol{Y}^k), \qquad P\delta\boldsymbol{Y}^{k+1} = \boldsymbol{X}. \tag{4.16}$$

The goal of the preconditioned system (4.16) is to reduce the condition number of $\mathcal{J}(\boldsymbol{Y}^k)P^{-1}$ while the system $P\delta\boldsymbol{Y}^{k+1} = \boldsymbol{X}$ stays easily solvable. In case of JFNK the matrix-vector product becomes

$$\mathcal{J}(\boldsymbol{Y}^k)P^{-1}\boldsymbol{v} \approx \frac{\boldsymbol{F}(\boldsymbol{Y}^k + \epsilon P^{-1}\boldsymbol{v}) - \boldsymbol{F}(\boldsymbol{Y}^k)}{\epsilon}. \tag{4.17}$$

Traditionally, the preconditioner $P^{-1}$ is chosen to approximate the inverse of the Jacobian matrix $J$. Reasonable choice for the preconditioner $P^{-1}$ for multiphysics problems can be so called *physics-based preconditioning* [30]. The main idea is to simplify the original Jacobian using operator-splitting and linearizations to obtain $P$ and use an approximation of its inverse as preconditioner $P^{-1}$. We should note that the operator-splitting method is used only to obtain the preconditioner not for the the solution itself. The fully nonlinear coupled problem is solved accurately, while the convergence of JFNK is fast since the number of Krylov iterations in GMRES method is lowered by an effective preconditioner. Recent works [29, 30, 21] show the effectiveness of this approach on various multiphysics problems.

Effectiveness of Newton's method over traditional operator-splitting schemes is demonstrated on an combustion example since flame propagation problems exhibit strong nonlinear behavior. Physics-based preconditioning together with a comparison of Newton's method and JFNK with different types of preconditioning is illustrated on the same problem in Example 4.3.

**Example 4.2** *We use a simplified model of freely propagating laminar flame and its response to a heat-absorbing obstacle represented by a set of cooled parallel rods with a rectangular cross-section. The domain $\Omega$ is shown in Fig. 4.13.*

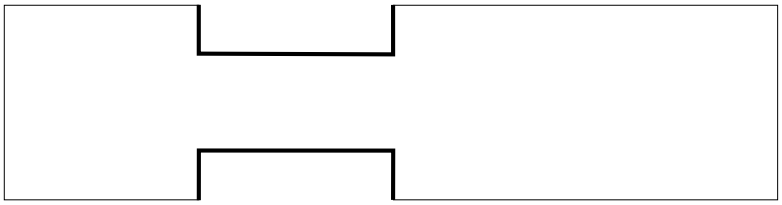


Figure 4.13: Computational domain.

*The model consists of a system of two coupled nonlinear parabolic equations for $\theta$ and $Y$,*

$$\frac{\partial \theta}{\partial t} - \Delta\theta = \omega(\theta, Y) \quad \textit{in } \Omega \times (0, T_0), \tag{4.18}$$

$$\frac{\partial Y}{\partial t} - \frac{1}{\mathrm{Le}}\,\Delta Y = -\omega(\theta, Y) \quad \textit{in } \Omega \times (0, T_0). \tag{4.19}$$

*Here, the reaction rate $\omega(\theta, Y)$ is defined by the Arrhenius law*

$$\omega(\theta, Y) = \frac{\beta^2}{2\mathrm{Le}} Y e^{\frac{\beta(\theta-1)}{1+\alpha(\theta-1)}}, \tag{4.20}$$

*where $\alpha$ is the gas expansion coefficient in a flow with nonconstant density, $\beta$ the non-dimensional activation energy, and* Le *the Lewis number (ratio of diffusivity of heat and diffusivity of mass).*

*The problem was solved by two methods. First, temperature and concentration were decoupled, nonlinear term $\omega(\theta, Y)$ was linearized and then the fixed-point iteration procedure was run. In the second approach the problem was solved as fully nonlinear, thus employing Newton's method. For discretization of both components an uniformly refined mesh with cubic elements was used. Average number of nonlinear iterations on one time level for both methods is shown in Tab. 4.2. Note that for larger time steps fixed-point iterations converge slowly if at all. Fig. 4.14 shows the error convergence of the reaction rate $\omega(\theta, Y)$ in $H^1$-norm in time for three different treatments of the nonlinearity – simple linearization, fixed-point iterations and Newton's method. For all computations the second order scheme was used for time discretization and shown errors are with respect to the "overkill" reference solution. As is obvious from the graph, traditional operator-splitting schemes reduce the order of convergence to one (even fixed-point iterations do not recover the second order accuracy), while by Newton's method the expected second-order convergence is obtained.*

Table 4.2: Average number of nonlinear iterations.

| **time step** | $\tau = 0.1$ | $\tau = 0.01$ | $\tau = 0.001$ | $\tau = 0.0001$ |
|---|---|---|---|---|
| fixed-point iterations | DNC | 19 | 6 | 4 |
| newton's method | 4 | 3 | 2 | 2 |

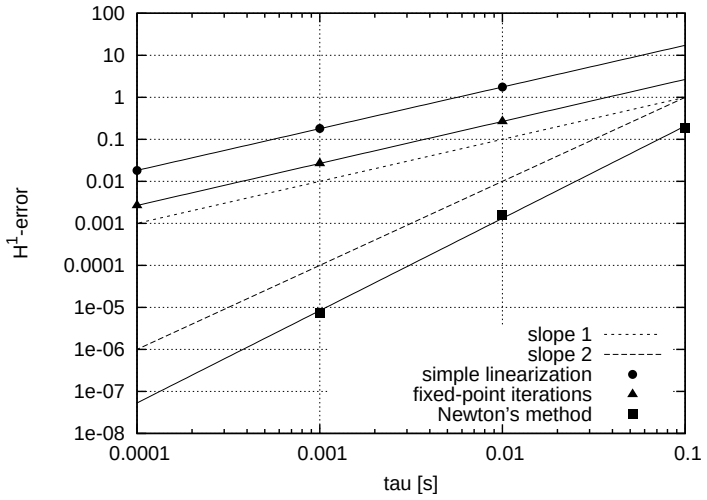



Figure 4.14: Comparison of order of convergence for operator-splitting schemes and Newton's method for the flame propagation problem.

**Example 4.3** *The same problem as in Example 4.2 was solved using the Newton's method and the Jacobian-Free Newton Krylov method with different preconditioning approaches. Let us state the Jacobian matrix for problem (4.18) - (4.19):*

$$\mathcal{J} = \begin{pmatrix} \frac{1}{\tau}\varphi_i\varphi_j + \nabla\varphi_i \cdot \nabla\varphi_j - \frac{\partial\omega}{\partial\theta}\varphi_i\varphi_j, & -\frac{\partial\omega}{\partial Y}\phi_i\varphi_j \\ -\frac{\partial\omega}{\partial\theta}\varphi_i\phi_j, & \frac{1}{\tau}\phi_i\phi_j + \frac{1}{\mathrm{Le}}\nabla\phi_i \cdot \nabla\phi_j + \frac{\partial\omega}{\partial Y}\phi_i\phi_j \end{pmatrix},$$

*where $\varphi_i$ and $\phi_j$ correspond to the test functions for temperature and concentration, respectively. For Newton's method the matrix $\mathcal{J}$ is assembled and resulting linear system is solved in each nonlinear iteration. As mentioned in the previous paragraphs, to speed up the linear solver suitable matrix $P^{-1}$ is used as preconditioner. In our computations $P^{-1} \approx \mathcal{J}^{-1}$ is obtained using Trilinos package* ML *[22] developed in Sandia National Laboratories, which uses multigrid methods to approximately invert the Jacobian matrix. In the Jacobian-free Newton-Krylov method the matrix $\mathcal{J}$ does not have to be assembled, since it is approximated using the residual F itself. Instead, the approximation of $\mathcal{J}$ is used as the physics-based preconditioner. In this procedure the residual F is first linearized using the solution from the previous level and then the Jacobian of such linearized residual is derived:*

$$P^{-1} \approx \tilde{\mathcal{J}}^{-1},$$

$$\tilde{\mathcal{J}} = \begin{pmatrix} \frac{1}{\tau}\varphi_i\varphi_j + \nabla\varphi_i \cdot \nabla\varphi_j, & 0 \\ 0, & \frac{1}{\tau}\phi_i\phi_j + \frac{1}{\mathrm{Le}}\nabla\phi_i \cdot \nabla\phi_j \end{pmatrix}.$$

*Tab. 4.3 compares computational times and numbers of linear iterations for Newton's method and JFNK with different preconditioners. As a preconditioner was taken the full Jacobian $\mathcal{J}$ and also its approximation via operator-splitting $\tilde{\mathcal{J}}$. It can be seen that even though the physics-based preconditioner is slightly worse than the full Jacobian as preconditioner (the number of iterations in the linear solver is higher), the time to invert the approximation of Jacobian $\mathcal{J}$ is much shorter (mainly due to the fact that the assembling of the Jacobian takes long).*

Table 4.3: Computational times and numbers of iterations in 1 nonlinear iteration.

| **Method** | assembling of Jacobian $\mathcal{J}$ | time to create preconditioner | linear solver | # of linear iterations |
|---|---|---|---|---|
| Newton's method with preconditioning | 11.6 | 0.147 | 0.25 | 35 |
| JFNK with Jacobian precond. | 0 | 11.8 | 3.34 | 4 |
| JFNK with physics-based precond. | 0 | 1.3 | 6.26 | 9 |

## 4.4 Independent meshes for different physical fields

As we described in Section 4.3, an alternative to solving nonlinear system in a loosely coupled manner (operator-splitting) is to solve them monolithically, thus using a tight coupling. But in case the qualitative behavior of various physics components differs significantly then in order to capture individual behaviors more efficiently, different physics should be discretized on individual $hp$-meshes, each of them tailored to its component. Since the solution components are coupled and the nonlinear system is solved monolithically, independent meshes for different components are not generally feasible. In reality, we are usually forced to use one mesh for all components, which contains the union of all required refinements. But this is a waste of degrees of freedom as is illustrated in Fig. 4.15 on an example of microwave heating (for more details on the example see [14]).

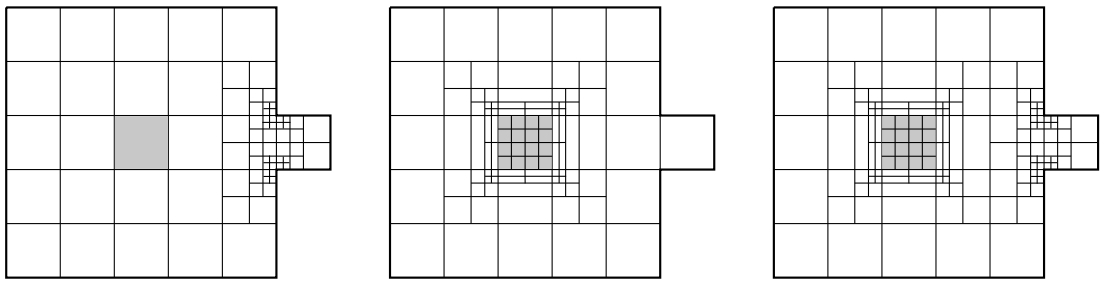



Figure 4.15: Schematic pictures of individual meshes for a coupled problem, where electric field heats up a load in the microwave oven. Electric field has singularities at the corners, while temperature has a steep internal layer at the material interface. Union mesh containing all refinements would be unnecessarily fine.

To obtain the solution of a nonlinear system we solve in each Newton's iteration the following system

$$\mathcal{J}(\boldsymbol{Y}^k)\delta\boldsymbol{Y}^{k+1} = -\boldsymbol{F}(\boldsymbol{Y}^k), \tag{4.21}$$

where elements of Jacobi matrix $\boldsymbol{J}$ involve evaluation of various integrals. Functions to be integrated can be from different function spaces (corresponding to different solution components) since the problem is coupled. Thus, for instance we are forced to integrate integrals of the type

$$\int_\Omega \lambda(T) v_i v_j \mathrm{d}\boldsymbol{x}, \tag{4.22}$$

where parameter $\lambda$ depends on one physics while $v_i, v_j$ are basis functions for two different physics. Now, let us assume that individual meshes $\mathcal{T}_i$ are necessary for all components in order to capture different behavior of components. Here, the problem of numerical evaluation of integral (4.22) arises.

In [43, 14, 47] we proposed a novel approach – a multimesh assembling technology for efficient solution of coupled problems. It allows monolithic solution of nonlinear coupled problems with solution components discretized on geometrically different meshes. Let us describe the algorithm and its main characteristics in the following paragraphs.

### 4.4.1 Multimesh assembling technology

Ideally, the meshes $\mathcal{T}_i$ would be completely independent. However, for algorithmic reasons, we introduce a simplifying assumption that each of them is defined starting

from a common coarse master mesh $\mathcal{T}_m$ by a finite sequence of (mutually independent) elementary refinement operations. The master mesh $\mathcal{T}_m$ is very coarse and it may not be used for discretization purposes. It serves as the top of a tree-like structure of meshes which is used by the multimesh assembling procedure. The geometrical union of all meshes in the system, containing all refinements, is called the union mesh and denoted by $\mathcal{T}_u$. The situation is illustrated in Fig. 4.16.

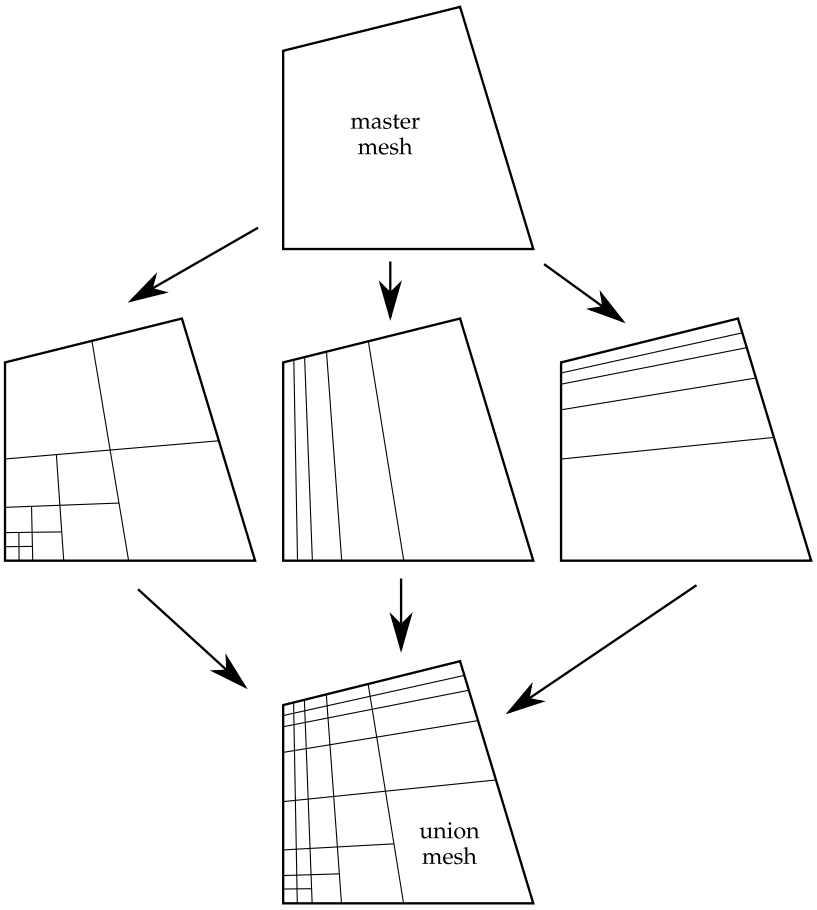



Figure 4.16: Master mesh $\mathcal{T}_m$, three different component meshes $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_3$ and their geometric union $\mathcal{T}_u$.

In the assembling procedure, when evaluating integrals, we walk through the elements $Q_k$ of the union mesh and restrict the evaluation of the bilinear or linear form to these subelements of the component meshes, as is shown in Fig. 4.17. Evaluation of integrals on the subelements $Q_k$ is described in detail in Section 4.4.2.

Even though evaluation of integrals takes place on subelements of the union mesh $\mathcal{T}_u$, the union mesh does not have to be explicitly constructed. A simple recursive algorithm for the traversal of a virtual union mesh is described in Section 4.4.3. It

should be noted that the assembling over different meshes is not more efficient than standard assembling over the real union mesh used for all solution components. The time savings are expected in the solution of the resulting linear system, whose size is smaller than when the union mesh was used for all solution components.

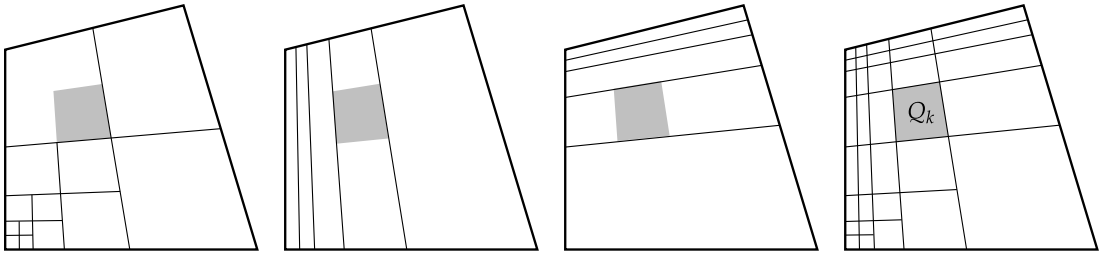



Figure 4.17: Virtual element $Q_k$ in component meshes and in the union mesh $\mathcal{T}_u$.

### 4.4.2 Integration over virtual elements

Let us consider a virtual element $Q_k$ of the union mesh $\mathcal{T}_u$ and $n$ individual meshes for solution components. In order to evaluate integrals in (4.21) over the virtual element $Q_k$, we need to extend the affine concept from Section 2.2.1. In Fig. 4.17 we see that the shaded areas in each mesh correspond to those depicted in Fig. 4.18 in the reference domain.

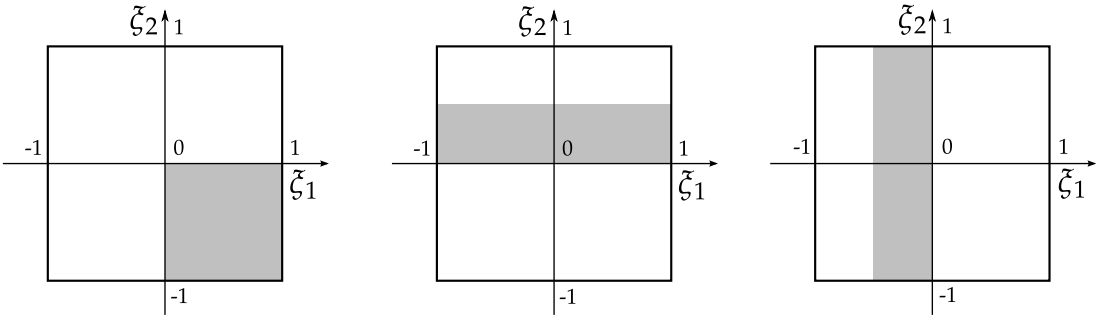



Figure 4.18: Areas of the reference domain corresponding to $Q_k$ in each mesh from Fig. 4.17.

In each mesh $\mathcal{T}_i$, $i = 1, \ldots, n$, there is exactly one element $K(i)$ such that $Q_k \subset K(i)$. Since we are only able to integrate over the whole reference domain (where the Gauss quadrature points and weights are defined), we need to transform subelements $Q_k$ to the reference domain $K_q = (-1, 1)^2$. We know, that for any physical element $K = K(i)$, the corresponding reference map is denoted by $\boldsymbol{x}_K : K_q \to K$. It can be seen that while $\boldsymbol{x}_K^{-1}(K) = K_q$, the virtual element $Q_k \subset K$ transforms to a subset $\boldsymbol{x}_K^{-1}(Q_k) \subset K_q$, as is shown in Fig. 4.18. Thus, we need to introduce an additional mapping $\boldsymbol{r} : K_q \to K_q$

such that $\boldsymbol{r}(K_q) = x_K^{-1}(Q_k)$. This is illustrated in Fig. 4.19. Hence, for each virtual element $Q_k \subset K$ (the right part of Fig. 4.19), Gauss integration of sufficiently high order is performed on $K_q$ (the left part of Fig. 4.19). Note that the mapping $\boldsymbol{r}$ has the form $\boldsymbol{r}(\boldsymbol{\xi}) = \boldsymbol{R}\boldsymbol{\xi} + \boldsymbol{s}$, and its Jacobi matrix is diagonal.

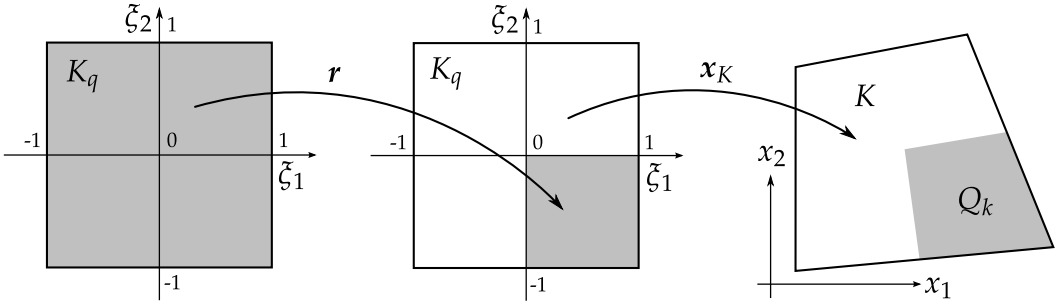



Figure 4.19: The mappings $\boldsymbol{r}$ and $\boldsymbol{x}_K$.

For example, in Fig. 4.19 the mapping has the form

$$\boldsymbol{r}(\boldsymbol{\xi}) = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix} \boldsymbol{\xi} + \begin{pmatrix} 0.5 \\ -0.5 \end{pmatrix}$$

Now let us describe how the integration of the weak form (2.8) changes. Recall that $v_i$ and $v_j$ are basis functions defined on geometrically different meshes $\mathcal{T}_i$, $\mathcal{T}_j$, respectively. We transform the integral over the virtual element $Q_k$ to the reference domain $K_q$ applying the mapping $\boldsymbol{x}_K \circ \boldsymbol{r}$ as follows

$$a(v_j, v_i) = \sum_{Q_k \in \mathcal{T}_u} \int_{Q_k} a_1 \nabla v_j(\boldsymbol{x}) \cdot \nabla v_i(\boldsymbol{x}) + a_0\, v_j(\boldsymbol{x})\, v_i(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} =$$

$$= \sum_{Q_k \in \mathcal{T}_u} \int_{K_q} J\, \hat{a}_1(\boldsymbol{\xi}) \left[ \left( \frac{D(\boldsymbol{x}_{K_j} \circ \boldsymbol{r}_j)}{D\boldsymbol{\xi}} \right)^{-T} \nabla \hat{v}_j(\boldsymbol{\xi}) \right] \cdot \left[ \left( \frac{D(\boldsymbol{x}_{K_i} \circ \boldsymbol{r}_i)}{D\boldsymbol{\xi}} \right)^{-T} \nabla \hat{v}_i(\boldsymbol{\xi}) \right] \mathrm{d}\boldsymbol{\xi}$$

$$+ \sum_{Q_k \in \mathcal{T}_u} \int_{K_q} J\, \hat{a}_0(\boldsymbol{\xi})\, \hat{v}_j(\boldsymbol{\xi}) \hat{v}_i(\boldsymbol{\xi})\, \mathrm{d}\boldsymbol{\xi} =$$

$$= \sum_{Q_k \in \mathcal{T}_u} \int_{K_q} J\, \hat{a}_1(\boldsymbol{\xi}) \left[ \left( \frac{D\boldsymbol{r}_j}{D\boldsymbol{\xi}} \right)^{-T} \left( \frac{D\boldsymbol{x}_{K_j}}{D\boldsymbol{\xi}} \right)^{-T} \nabla \hat{v}_j(\boldsymbol{\xi}) \right] \cdot \left[ \left( \frac{D\boldsymbol{r}_i}{D\boldsymbol{\xi}} \right)^{-T} \left( \frac{D\boldsymbol{x}_{K_i}}{D\boldsymbol{\xi}} \right)^{-T} \nabla \hat{v}_i(\boldsymbol{\xi}) \right] \mathrm{d}\boldsymbol{\xi}$$

$$+ \sum_{Q_k \in \mathcal{T}_u} \int_{K_q} J\, \hat{a}_0(\boldsymbol{\xi})\, \hat{v}_j(\boldsymbol{\xi}) \hat{v}_i(\boldsymbol{\xi})\, \mathrm{d}\boldsymbol{\xi},$$

where

$$\begin{aligned} \hat{v}_i(\boldsymbol{\xi}) &= (v_i \circ \boldsymbol{x}_{K_i} \circ \boldsymbol{r}_i)(\boldsymbol{\xi}), \\ \hat{v}_j(\boldsymbol{\xi}) &= (v_j \circ \boldsymbol{x}_{K_j} \circ \boldsymbol{r}_j)(\boldsymbol{\xi}), \end{aligned}$$

and the Jacobian of the mapping,

$$J = \det\left(\frac{D\boldsymbol{x}_{K_i}}{D\boldsymbol{\xi}}\right) \det\left(\frac{D\boldsymbol{r}_i}{D\boldsymbol{\xi}}\right) = \det\left(\frac{D\boldsymbol{x}_{K_j}}{D\boldsymbol{\xi}}\right) \det\left(\frac{D\boldsymbol{r}_j}{D\boldsymbol{\xi}}\right).$$

Here $\boldsymbol{x}_{K_i}$ and $\boldsymbol{x}_{K_j}$ are traditional reference mappings which correspond to two meshes $\mathcal{T}_i$, $\mathcal{T}_j$, thus $Q_k \subset K_i \in \mathcal{T}_i$ and $Q_k \subset K_j \in \mathcal{T}_j$. Mappings $\boldsymbol{r}_i$ and $\boldsymbol{r}_j$ are restricting mappings to two different subelements of the reference domain depending on the position of $Q_k$ in $K_i$ and $K_j$, respectively.

The extended reference mapping itself should not change the performance of the code, since the combined Jacobian and the inverse Jacobi matrices can be precalculated as usual.

### 4.4.3 Union mesh traversal

As mentioned above, the union mesh is virtual, thus never constructed physically in the memory. We just need to enumerate its elements $Q_k$ together with corresponding restricting mappings $\boldsymbol{r}$ for each component mesh $\mathcal{T}_i$. The algorithm presumes that all component meshes are obtained from one common master mesh $\mathcal{T}_m$ by elementary refinements. This allows us to predefine all possible transformations $\boldsymbol{r}$ using the four elementary transformations for triangles and eight elementary transformations for quadrilaterals, as is shown in Fig. 4.20.

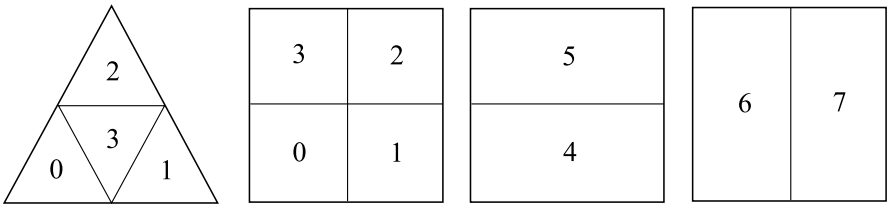



Figure 4.20: Predefined subelement transformations.

Mapping $\boldsymbol{r}$ for the next level of refinements is obtained as a composition of two elementary transformations, as is illustrated in Fig. 4.21 and mathematically expressed as follows

$$(\boldsymbol{r}_7 \circ \boldsymbol{r}_3)(\boldsymbol{\xi}) = \boldsymbol{R}_7(\boldsymbol{R}_3\boldsymbol{\xi} + \boldsymbol{s}_3) + \boldsymbol{s}_7 = (\boldsymbol{R}_7\boldsymbol{R}_3)\boldsymbol{\xi} + (\boldsymbol{R}_7\boldsymbol{s}_3 + \boldsymbol{s}_7).$$

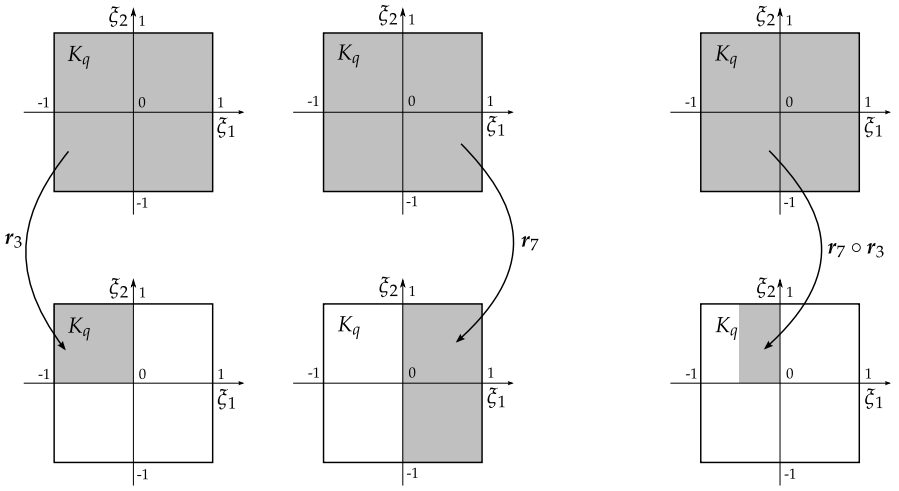



Figure 4.21: Composition of basic transformations.

The traversal algorithm begins with a main loop visiting each element of the coarse master mesh. For these elements it calls a recursive procedure which traverses the refinement trees in all component meshes. It goes deeper and deeper in the mesh hierarchy until it finds the virtual element $Q_k$, such that all elements $K_i$, $i = 1, \ldots, n$, $Q_k \subset K_i \in \mathcal{T}_i$ are active. Then it performs the assembling procedure on $Q_k$ as was described in Section 4.4.2.

### 4.4.4 Illustrative example

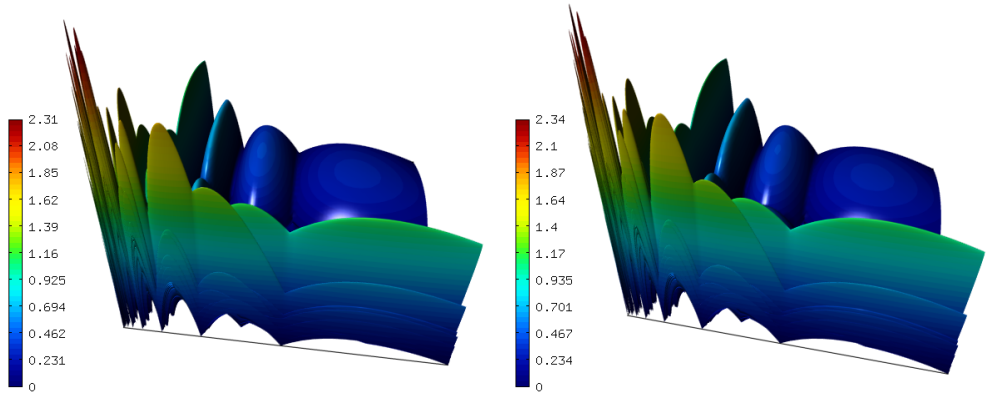
Let us recall the example from Section 4.2.4, where the material parameter was defined on the mesh different from the mesh on which the solution was sought. In order to avoid error resulting from data-transfer methods (as illustrated in Section 4.2.4), either the union mesh has to be constructed or the assembling over different meshes must be employed.

Let us compare the performance of the multimesh assembling with the data-transfer methods described in Section 4.2. The problem (4.9) was solved without necessity of handling different meshes via the transfers between meshes. It can be seen from Tab. 4.4 and Fig. 4.22 that by using the multimesh, the same accuracy as by using the exact data was achieved.

**Remark 4.3** *Let us note that even though the resulting linear system has the same size as in case of data-transfer methods, the assembling procedure takes place over the subelements of the virtual union mesh, resulting in increase of computational time. Nevertheless, in case of data-transfer methods usually an additional problem must be solved in order to transfer the solution from the source mesh onto target mesh, which increases the computational time as well.*

Table 4.4: Comparison of relative $H^1$-norm errors for the multimesh technology and data-transfer methods.

| **Approach** | **Relative $H^1$-error** |
|---|---|
| Interpolation $k_h^{(1)}$ | 17.766% |
| $L^2$-projection $k_h^{(2)}$ (lin.) | 31.808% |
| Multimesh | 6.727% |
| Using exact data $k(x, y)$ | 6.729% |




(a) Using exact data (b) Multimesh

Figure 4.22: Approximation errors on the mesh $\mathcal{T}_B$ calculated with respect to the exact solution $u$.

## 4.5 Automatic $hp$-adaptivity for coupled problems

The automatic $hp$-adaptive procedure for a single PDE was described in detail in Chapter 3. Let us extend this concept to a system of coupled PDEs. Assume a system of $n$ coupled, possibly nonlinear equations written in the form of residual functions

$$F_i(u_1, \dots, u_n) = 0, \quad i = 1, \dots, n, \tag{4.23}$$

where $u_1 \in V_{hp}^1, \dots, u_n \in V_{hp}^n$ are solution components generally discretized on individual meshes $\mathcal{T}_1, \dots, \mathcal{T}_n$ and equipped with different polynomial degrees. The system is solved either via Newton's method or fixed-point iterations. In agreement with the standard $hp$-adaptivity, reference solutions $u_1^{\text{ref}}, \dots, u_n^{\text{ref}}$ are obtained in enriched polynomial spaces. In constract to standart $hp$-FEM, the approximation error $\boldsymbol{e}_{hp}$ is

a vector-valued function with the components $e^1_{hp}, \ldots, e^n_{hp}$ (corresponding to the fields $u_1, \ldots, u_n$, respectively)

$$e^i_{hp} = u_i - u_i^{\text{ref}}, \quad i = 1, \ldots, n.$$

In the adaptive multimesh $hp$-FEM either each component error $e^i_{hp}$ is prescribed to be below given tolerance or the total error

$$||\boldsymbol{e}_{hp}||^2 = \sum_{i=1}^{n} ||e^i_{hp}||^2 \tag{4.24}$$

need to drop below prescribed tolerance to stop the adaptive procedure. In (4.24) the question of the suitable norm in which to measure component errors $e^i_{hp}$ arises. If solution components do not interact with each other, then an $H^1$-norm could be used for each component $||e^i_{hp}||_{H^1}$. But if the problem is strongly coupled and error in one component influences errors of the other ones, it should be also reflected in computed error $||e^i_{hp}||$. Thus,

$$\begin{aligned}
||e^1_{hp}||^2 &= a_{11}(e^1_{hp}, e^1_{hp}) + a_{12}(e^2_{hp}, e^1_{hp}) + \cdots + a_{1n}(e^n_{hp}, e^1_{hp}), \\
||e^2_{hp}||^2 &= a_{21}(e^1_{hp}, e^2_{hp}) + a_{22}(e^2_{hp}, e^2_{hp}) + \cdots + a_{2n}(e^n_{hp}, e^2_{hp}), \\
&\vdots \\
||e^n_{hp}||^2 &= a_{n1}(e^1_{hp}, e^n_{hp}) + a_{n2}(e^2_{hp}, e^n_{hp}) + \cdots + a_{nn}(e^n_{hp}, e^n_{hp}),
\end{aligned}$$

where $a_{ij}(\cdot, \cdot)$ are suitable bilinear forms expressing the interactions between individual physical fields. In case the system is linear, corresponding energy norms give a possible choice [43].

For each element $K^{(i)} \in \mathcal{T}_i$, the error value $e^{(i)}_K$ is defined as an element contribution to the error $||e^i_{hp}||$. Thus,

$$||e^i_{hp}|| = \sum_{K^{(i)} \in \mathcal{T}_i} e^{(i)}_K.$$

Elements of all meshes $\mathcal{T}_1, \ldots, \mathcal{T}_n$ are then collected in a single list and sorted according to the values $e^{(i)}_K$ in descending order. Then certain amount of them is marked for refinement following the standard adaptivity algorithm described in Alg. 3. To select an optimal refinement for particular element $K^{(i)} \in \mathcal{T}_i$, corresponding error function $e^i_{hp}$ and the standard $H^1$ or $L^2$ projections are used as was described in Section 3.2.1.

### 4.5.1 Multimesh assembling and numerical quadrature

Let us address the biggest problem of the multimesh assembling technique from the point of view of the *hp*-adaptivity. The main purpose of the multimesh is to save degrees of freedom by allowing different meshes in one computation instead of using the union mesh for all solution components. As mentioned earlier, the time savings related to the savings in DOFs are expected in the solving time, since the resulting stiffness matrices are smaller. It was also mentioned that in the assembling stage, there are no time savings in comparison to the single-mesh approach (the union mesh for all components). The reason is that the evaluation of the stiffness matrix entries actually takes place on the virtual union mesh. When all meshes have the same polynomial degrees on corresponding elements (or for example in the whole computational domain), the assembling times for the multimesh and single-mesh are comparable. But let us assume a situation when meshes differ extremely in both geometry and polynomial orders in the same part of the domain (such a situation is obtained by the automatic *hp*-adaptivity individually applied to the solution components). For example, one meshes has 4 large elements of the 8th order and the other hundreds of linear elements. Let us look at the evaluation of the integral

$$\int_{Q_k} J\ \hat{v}_i^1\ \hat{v}_j^2\ \mathrm{d}\xi,$$

where $\hat{v}_i^1$ is the 8th order basis function defined on the large element and appropriately transformed to the $Q_k$, and $\hat{v}_j^2$ is linear basis function defined on the small element appropriately transformed to the $Q_k$. The quadrature rule necessary to evaluate such an integral is the sum of polynomial orders of $v_i^1$, $v_j^2$ plus the effect of the reference mapping. This is obviously time consuming, since the restriction of the high-order basis function $v_i^1$ on the subelement $Q_k$ may not necessarily be integrated by such a high-order quadrature rule to achieve numerically accurate results. Using single-mesh approach, where automatic *hp*-adaptivity would take into account both solution components and select an optimal refinement to suit them both as well as possible (even though worse than multimesh), would in such a situation lead to much lower assembling times and might lead to better overall times as well.

Possible remedy to this problem could be an adjustment of integration orders in these situations. Tables 4.5 – 4.8 show quadrature rules necessary to evaluate integrals of certain shape functions accurately. For illustration we take four bubble functions on the reference domain $(-1, 1)^2$ and integrate them over subelements $Q_k$ of six virtual union meshes $\mathcal{T}_1$ – $\mathcal{T}_6$. The mesh $\mathcal{T}_1$ consists of one element (the same as the mesh on which the bubble function is defined) and every consequent mesh is created by uniformly refining all elements, thus $\mathcal{T}_2$ consists of 4 elements, $\mathcal{T}_3$ consists of 16 elements, etc. The exact value of all integrals is zero and green color denotes "sufficiently" accurate results. It can be seen from the tables that in case the meshes differ significantly,

Table 4.5: Integrals of the bubble basis function of the 4th degree integrated by different quadrature rules. Integration takes place on elements $Q_k$ of meshes $\mathcal{T}_1$ – $\mathcal{T}_6$.

| | Mesh refinements | | | | | |
|---|---|---|---|---|---|---|
| quadr. | 1 elem. | 4 elem. | 16 elem. | 64 elem. | 256 elem. | 1024 elem. |
| 1 | 0.218 | 0.00769 | 0.00120 | 8.982e-05 | 5.853e-06 | 3.696e-07 |
| 3 | 0.0432 | 0.000168 | 6.593e-07 | 2.575e-09 | 1.006e-11 | 3.930e-14 |
| 5 | -2.775e-17 | 5.421e-18 | 0 | 1.301e-18 | 9.757e-19 | 1.524e-20 |

Table 4.6: Integrals of the bubble basis function of the 6th degree integrated by different quadrature rules. Integration takes place on elements $Q_k$ of meshes $\mathcal{T}_1$ – $\mathcal{T}_6$.

| | Mesh refinements | | | | | |
|---|---|---|---|---|---|---|
| quadr. | 1 elem. | 4 elem. | 16 elem. | 64 elem. | 256 elem. | 1024 elem. |
| 1 | 0.0859 | 0.0681 | 0.000187 | 9.823e-05 | 8.457e-06 | 5.69e-07 |
| 3 | 0.0679 | 0.00663 | 4.377e-05 | 1.913e-07 | 7.678e-10 | 3.019e-12 |
| 5 | 0.0198 | 4.834e-06 | 1.18e-09 | 2.881e-13 | 6.992e-17 | 1.486e-18 |
| 7 | -3.036e-18 | 2.385e-18 | 1.626e-19 | -1.816e-18 | 5.421e-19 | 4.472e-19 |

Table 4.7: Integrals of the bubble basis function of the 8th degree integrated by different quadrature rules. Integration takes place on elements $Q_k$ of meshes $\mathcal{T}_1$ – $\mathcal{T}_6$.

| | Mesh refinements | | | | | |
|---|---|---|---|---|---|---|
| quadr. | 1 elem. | 4 elem. | 16 elem. | 64 elem. | 256 elem. | 1024 elem. |
| 1 | 0.0457 | 0.021 | 0.00303 | 2.671e-05 | 8.78e-06 | 7.281e-07 |
| 3 | 0.000643 | 0.00541 | 0.000385 | 2.904e-06 | 1.308e-08 | 5.287e-11 |
| 5 | 0.0345 | 0.00119 | 4.715e-07 | 1.277e-10 | 3.197e-14 | 8.089e-18 |
| 7 | 0.0113 | 1.733e-07 | 2.644e-12 | 4.148e-17 | 2.711e-20 | -6.251e-19 |
| 9 | -4.025e-18 | -7.996e-19 | -1.64e-18 | -1.804e-18 | 2.338e-19 | 1.88e-19 |

thus the higher-order shape function is actually integrated over very small subelements, the order of the quadrature rule necessary to integrate such a shape function can be lowered. By utilizing this observation we could lower the assembling times for the multimesh, and thus improve its performance.

Table 4.8: Integrals of the bubble basis function of the 10th degree integrated by different quadrature rules. Integration takes place on elements $Q_k$ of meshes $\mathcal{T}_1$ – $\mathcal{T}_6$.

| | Mesh refinements | | | | | |
|---|---|---|---|---|---|---|
| quadr. | 1 elem. | 4 elem. | 16 elem. | 64 elem. | 256 elem. | 1024 elem. |
| 1 | 0.0284 | 0.00138 | 0.00510 | 4.574e-05 | 5.304e-06 | 7.952e-07 |
| 3 | 0.0186 | 0.00638 | 0.000191 | 1.689e-05 | 1.036e-07 | 4.491e-10 |
| 5 | 0.00145 | 0.000336 | 2.143e-05 | 8.736e-09 | 2.393e-12 | 6.011e-16 |
| 7 | 0.021 | 0.000148 | 3.589e-09 | 6.059e-14 | 2.337e-18 | 4.235e-20 |
| 9 | 0.00736 | 7.022e-09 | 6.697e-15 | -4.167e-19 | -8.743e-19 | -3.558e-20 |
| 11 | 8.648e-19 | -1.603e-18 | 2.86e-18 | -5.624e-19 | -2.965e-19 | -2.101e-19 |

## 4.6 Heat and moisture transfer in a nuclear reactor vessel

Let us demonstrate the monolithic discretization with adapted multiple meshes for solution components on an example from civil engineering – heat and moisture transfer in a nuclear reactor vessel described in [47]. The vessel is made of prestressed concrete, it is approximately 36 meters high and the thickness of the walls varies between 5 and 7.5 meters. The concrete is assumed homogeneous and isotropic. For this simulation, the vessel is assumed perfectly axisymmetric, although in reality there are small-scale features such as vents that make it nonsymmetric. The situation is illustrated in Fig. 4.23.

The unknown variables are the temperature $T$ [K], and relative humidity $w$ [-]. The corresponding gradients are denoted by $\boldsymbol{g}^{[T]}$ [K/m] and $\boldsymbol{g}^{[w]}$ [1/m], respectively, and the corresponding fluxes by $\boldsymbol{q}^{[T]}$ [J/m$^2$/s] and $\boldsymbol{q}^{[w]}$ [kg/m$^2$/s]. The heat flux obeys the Fourier law,

$$\boldsymbol{q}^{[T]}_{Fourier} = -\boldsymbol{D}^{[TT]}\boldsymbol{g}^{[T]}. \tag{4.25}$$

The moisture flux is described by the Fick law,

$$\boldsymbol{q}^{[w]}_{Fick} = -\boldsymbol{D}^{[ww]}\boldsymbol{g}^{[w]}. \tag{4.26}$$

Coupling between the fluxes is done using the Soret flux,

$$\boldsymbol{q}^{[w]}_{Soret} = -\boldsymbol{D}^{[wT]}\boldsymbol{g}^{[T]}, \tag{4.27}$$

and the Dufour flux,

$$\boldsymbol{q}^{[T]}_{Dufour} = -\boldsymbol{D}^{[Tw]}\boldsymbol{g}^{[w]}. \tag{4.28}$$

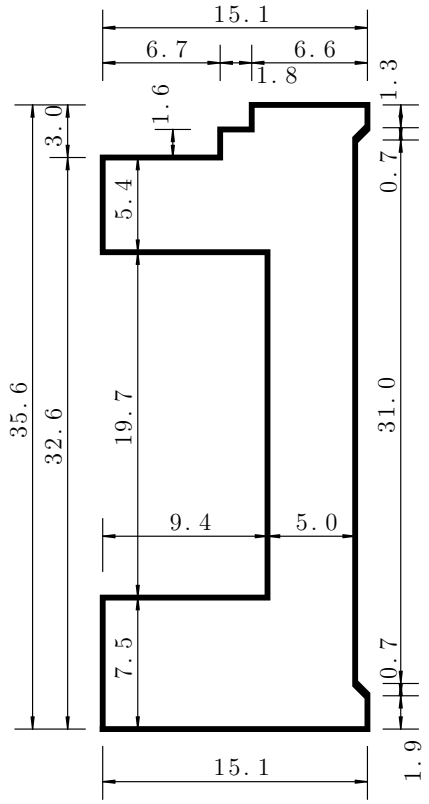



Figure 4.23: Geometry of the reactor vessel.

The total heat and moisture fluxes have the form

$$\boldsymbol{q}^{[T]} = -\boldsymbol{D}^{[TT]}\boldsymbol{g}^{[T]} - \boldsymbol{D}^{[Tw]}\boldsymbol{g}^{[w]}, \tag{4.29}$$

$$\boldsymbol{q}^{[w]} = -\boldsymbol{D}^{[wT]}\boldsymbol{g}^{[T]} - \boldsymbol{D}^{[ww]}\boldsymbol{g}^{[w]}. \tag{4.30}$$

In the case of a homogeneous and isotropic material, the matrices of material conductivities $\boldsymbol{D}^{[TT]}, \boldsymbol{D}^{[Tw]}, \boldsymbol{D}^{[wT]}$ and $\boldsymbol{D}^{[ww]}$ can be replaced with scalars, and equations (4.29), (4.30) reduce to

$$\boldsymbol{q}^{[T]} = -d^{[TT]}\boldsymbol{g}^{[T]} - d^{[Tw]}\boldsymbol{g}^{[w]}, \tag{4.31}$$

$$\boldsymbol{q}^{[w]} = -d^{[wT]}\boldsymbol{g}^{[T]} - d^{[ww]}\boldsymbol{g}^{[w]}. \tag{4.32}$$

The scalar conductivities have the following units: $d^{[TT]}$ [J/K/m/s], $d^{[Tw]}$ [J/m/s], $d^{[wT]}$ [kg/K/m/s], $d^{[ww]}$ [kg/m/s].

The balance equations of heat and moisture without source terms have the form

$$\frac{\partial \rho u}{\partial t} + \mathrm{div}\boldsymbol{q}^{[T]} = 0, \quad \frac{\partial \rho c}{\partial t} + \mathrm{div}\boldsymbol{q}^{[w]} = 0, \tag{4.33}$$

where $\rho$ is the total density, $u$ is the specific internal energy and $c$ is the mass concentration of moisture [6]. The last quantity is the ratio of the weight of water and the weight of the whole system. After substitution of the fluxes (4.31), (4.32) into (4.33) and rearrangement of the time derivatives, the balance equations for isotropic and homogeneous materials have the form

$$c^{[TT]}\frac{\partial T}{\partial t} + c^{[Tw]}\frac{\partial w}{\partial t} - d^{[TT]}\Delta T - d^{[Tw]}\Delta w = 0, \tag{4.34}$$

$$c^{[wT]}\frac{\partial T}{\partial t} + c^{[ww]}\frac{\partial w}{\partial t} - d^{[wT]}\Delta T - d^{[ww]}\Delta w = 0. \tag{4.35}$$

Coefficients $c^{[TT]}$, $c^{[Tw]}$, $c^{[wT]}$, $c^{[ww]}$ express capacity properties. For example, $c^{[TT]}$ is the specific heat capacity. In the following, we assume that the parameters $c^{[Tw]}$ and $c^{[wT]}$ are zero, which is the usual case. Even if they were nonzero, $\partial w/\partial t$ could be eliminated from (4.34) and $\partial T/\partial t$ from (4.35) using a suitable linear combination of (4.34), (4.35). Thus the final form of the equations to be solved is

$$c^{[TT]}\frac{\partial T}{\partial t} - d^{[TT]}\Delta T - d^{[Tw]}\Delta w = 0, \tag{4.36}$$

$$c^{[ww]}\frac{\partial w}{\partial t} - d^{[wT]}\Delta T - d^{[ww]}\Delta w = 0. \tag{4.37}$$

*Time-dependent boundary conditions:*

The boundary $\partial\Omega$ is split into three disjoint parts: $\Gamma_S$ (axis of symmetry), $\Gamma_R$ (reactor wall), and $\Gamma_E$ (exterior wall). On $\Gamma_S$, one prescribes zero Neumann conditions for both $T$ and $w$:

$$\frac{\partial T}{\partial \boldsymbol{n}} = 0, \quad \frac{\partial w}{\partial \boldsymbol{n}} = 0, \tag{4.38}$$

where $\boldsymbol{n}$ is the unit normal vector to $\partial\Omega$.

On the reactor wall $\Gamma_R$, one prescribes a Dirichlet condition for the temperature $T = \tilde{T}$ and a zero normal moisture flux

$$\boldsymbol{q}^{[w]} \cdot \boldsymbol{n} = \left(-d^{[wT]}\boldsymbol{g}^{[T]} - d^{[ww]}\boldsymbol{g}^{[w]}\right) \cdot \boldsymbol{n} = -d^{[wT]}\frac{\partial T}{\partial \boldsymbol{n}} - d^{[ww]}\frac{\partial w}{\partial \boldsymbol{n}} = 0.$$

Here, $\tilde{T}$ is a prescribed temperature. It is time-dependent and it corresponds to a linear temperature increase from $T_0 = 293.15$ K to $T_{max} = 550$ K in 24 hours.

On the exterior wall $\Gamma_E$, we prescribe Newton boundary conditions

$$\boldsymbol{q}^{[T]} \cdot \boldsymbol{n} = \kappa^{[TT]}(T - T_{ext}) + \kappa^{[Tw]}(w - w_{ext}), \tag{4.39}$$

$$\boldsymbol{q}^{[w]} \cdot \boldsymbol{n} = \kappa^{[wT]}(T - T_{ext}) + \kappa^{[ww]}(w - w_{ext}), \tag{4.40}$$

where $\kappa^{[TT]}$, $\kappa^{[Tw]}$, $\kappa^{[wT]}$ and $\kappa^{[ww]}$ are transmission coefficients, $T_{ext}$ is the temperature of environment surrounding the structure and $w_{ext}$ is relative humidity of environment surrounding the structure. We will assume that $\kappa^{[Tw]} = \kappa^{[wT]} = 0$, which is the common case. Then the conditions (4.39), (4.40) simplify to:

$$\boldsymbol{q}^{[T]} \cdot \boldsymbol{n} = -d^{[TT]}\frac{\partial T}{\partial \boldsymbol{n}} - d^{[Tw]}\frac{\partial w}{\partial \boldsymbol{n}} = \kappa^{[TT]}(T - T_{ext}), \tag{4.41}$$

$$\boldsymbol{q}^{[w]} \cdot \boldsymbol{n} = -d^{[wT]}\frac{\partial T}{\partial \boldsymbol{n}} - d^{[ww]}\frac{\partial w}{\partial \boldsymbol{n}} = \kappa^{[ww]}(w - w_{ext}). \tag{4.42}$$

The condition (4.42) is usually written in the form

$$\boldsymbol{q}^{[w]} \cdot \boldsymbol{n} = \beta(p - p_{ext}), \tag{4.43}$$

where $p$ is the water vapor pressure, $p_{ext}$ the water vapor pressure of the surrounding environment and $\beta$ the convection mass transfer coefficient. The relation between vapor pressure and the relative humidity $w$ has the form

$$w = \frac{p}{p_s(T)}, \tag{4.44}$$

where $p_s(T)$ is the water vapor saturation pressure. Substitution of (4.44) into (4.43) leads to the condition

$$\boldsymbol{q}^{[w]} \cdot \boldsymbol{n} = \beta(p_s(T)w - p_{s,ext}(T_{ext})w_{ext}) = \beta p_s(w - w_{ext}). \tag{4.45}$$

In the application discussed here, the temperature of the structure $T$ and the temperature of the surrounding environment $T_{ext}$ lie within a range where the dependence of $p_s$ on the temperature can be neglected (otherwise, the problem would be nonlinear).

*Initial condition:*

Initially we assume a uniform temperature $T_0 = 293.15$ K and uniform relative humidity $w(0) = 50\%$.

### 4.6.1 Numerical results

Let us compare the performance of three adaptive methods: (a) $h$-adaptive FEM with quadratic elements where the temperature and humidity are approximated on individual meshes, (b) adaptive $hp$-FEM where both fields are approximated on the same

mesh (standard *hp*-FEM), and (c) adaptive *hp*-FEM where both fields are approximated on individual meshes. It should be noted that we dropped *h*-adaptive FEM with linear elements from this comparison due to its excessive CPU times, and also that we dropped non-adaptive computations on fixed meshes.

To the last point, such comparison would be very interesting since non-adaptive computations with fixed uniform fine meshes still prevail in practical engineering computations of transient processes. However, in practice these are virtually never accompanied with an a-posteriori error estimate, and we believe that it is impossible to compare methods that use a-posteriori error control with methods that do not. Even if a non-adaptive computation was accompanied with an a-posteriori error information, in our opinion it still cannot be compared to an adaptive method since the former does not invest any effort into controlling the error while the latter does, and work and CPU time used by a method to control the error should not be counted to its disadvantage.

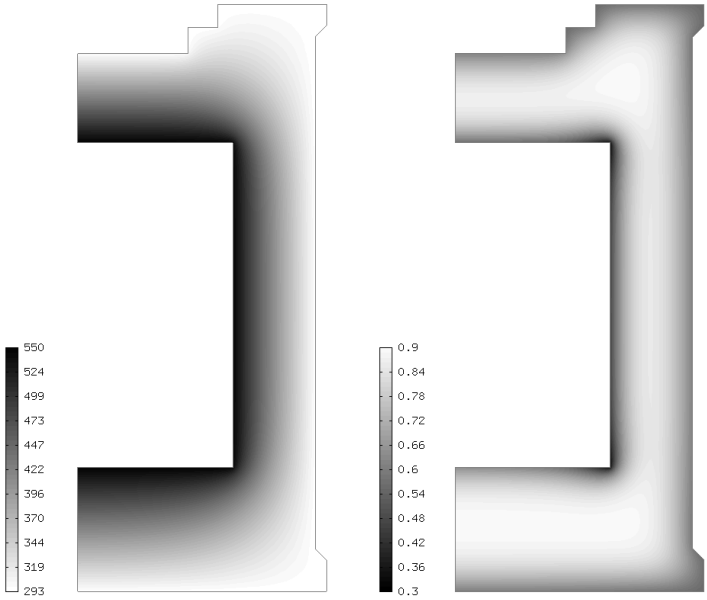



Figure 4.24: Temperature and moisture distribution in the vessel after 30 years.

We use the following material parameters (see [6]): $d^{[TT]} = 2.1$ [J/K/m/s], $d^{[Tw]} = 2.37\,10^{-2}$ [J/m/s], $d^{[wT]} = 1.78\,10^{-10}$ [kg/K/m/s], $d^{[ww]} = 3.02\,10^{-8}$ [kg/m/s], $c^{[TT]} = 2.18\ 10^{6}$ [J/K/m$^3$], $c^{[Tw]} = 0$, $c^{[wT]} = 0$, $c^{[ww]} = 2.49\ 10^{1}$ [kg/m$^3$]. The transition coefficients have the values $\kappa^{[TT]} = 25$ [J/K/m$^2$/s], $\kappa^{[Tw]} = 0$, $\kappa^{[wT]} = 0$, $\kappa^{[ww]} = 1.84\ 10^{-7}$ [kg/m$^2$/s].

Let us begin with showing the temperature and moisture distribution in the vessel after 30 years in Fig. 4.24.

*Adaptive h-FEM with quadratic elements (with individual meshes for temperature and moisture)*

Fig. 4.25 shows a series of finite element meshes generated by the *h*-adaptive FEM with quadratic elements. The reader can see that both mesh refinement and coarsening took place over the 30 year period.

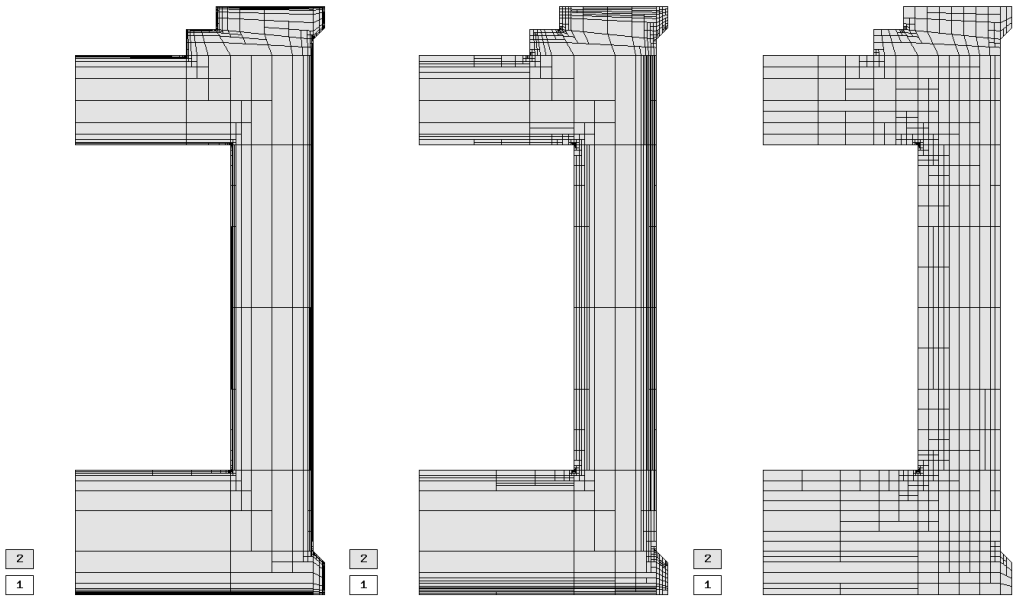



Figure 4.25: Adaptive *h*-FEM with quadratic elements (mesh after 1 month, 1 year, and 30 years)

*Standard adaptive hp-FEM (same meshes for temperature and moisture)*

Fig. 4.26 shows an analogous series of meshes generated by the standard (single-mesh) *hp*-adaptive FEM.

*Adaptive multimesh hp-FEM (with individual meshes for temperature and moisture)*

In the multimesh FEM we allow the meshes for the temperature and moisture fields to be different and to evolve in time independently of each other (to be described in detail in Chapter 5). If the two fields exhibit significantly different behaviors, then this approach can save many degrees of freedom. Figs. 4.27 and 4.28 show the corresponding series of temperature and moisture meshes, respectively.

*Comparison in terms of DOF and CPU time requirements*

Last let us compare the performance of the three adaptive methods in terms of degrees of freedom (DOF) and CPU time requirements. Let us remind the reader that the three

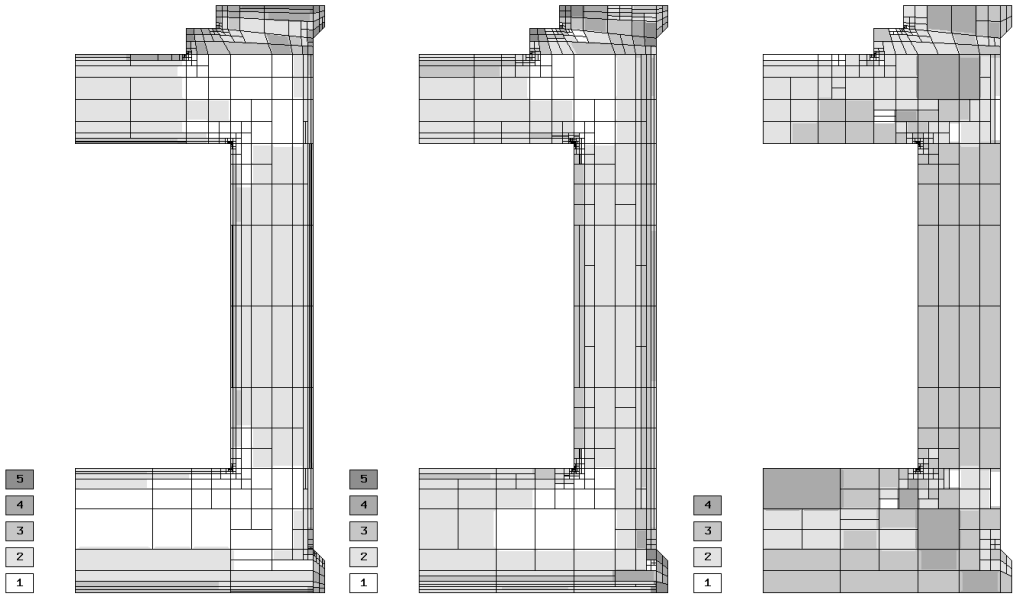



Figure 4.26: Single-mesh *hp*-FEM (mesh after 1 month, 1 year, and 30 years)

comparisons are fair in the sense that all three methods achieved the same accuracy in a given quantity of interest (total contents of moisture in the vessel after 30 years).

The reader can see in Fig. 4.29 that the *h*-FEM with quadratic elements consumes many more DOF than both versions of the adaptive *hp*-FEM. This is a standard observation that will not surprise anyone. It is more interesting to see that the difference between the standard and multimesh *hp*-FEM is more significant during the first approx. 15 years than in the second half of the time interval. This is due to the fact that in the early stage of the computation, the moisture develops a thin boundary layer which is not present in the temperature fields (see Figs. 4.27 and 4.28). Therefore the meshes for temperature and moisture are very different during the initial stage of the computation. Later, as the moisture layers become smeared, these differences vanish, and after 30 years both meshes become very similar (compare the right-most parts of Figs. 4.27 and 4.28).

The differences in the discrete problem sizes from Fig. 4.29 yield different CPU time requirements of the three methods, as shown in Fig. 4.30. Here, the reader may notice that the difference between the standard and multimesh *hp*-FEM is less significant than their difference in terms of DOF. This is due to numerical integration that is more involved in the multimesh case [43].

In Fig. 4.31 we compare the order of the convergence in time for two different numerical schemes. First, the problem was computed on an uniformly refined mesh using the simplest operator-splitting scheme, in which the values of the solution from the

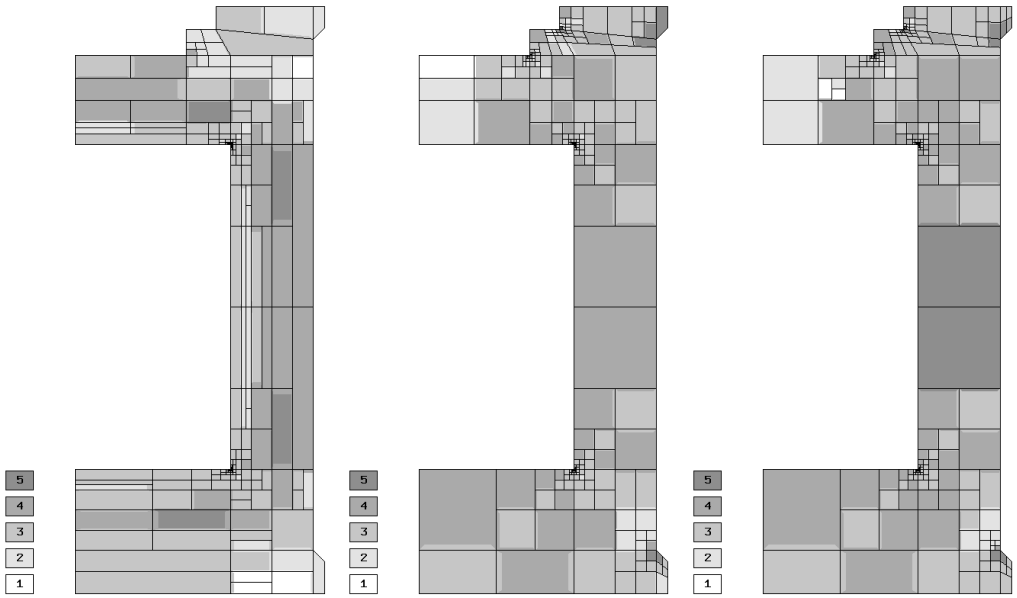



Figure 4.27: Multimesh *hp*-FEM (temperature mesh after 1 month, 1 year, 30 years)

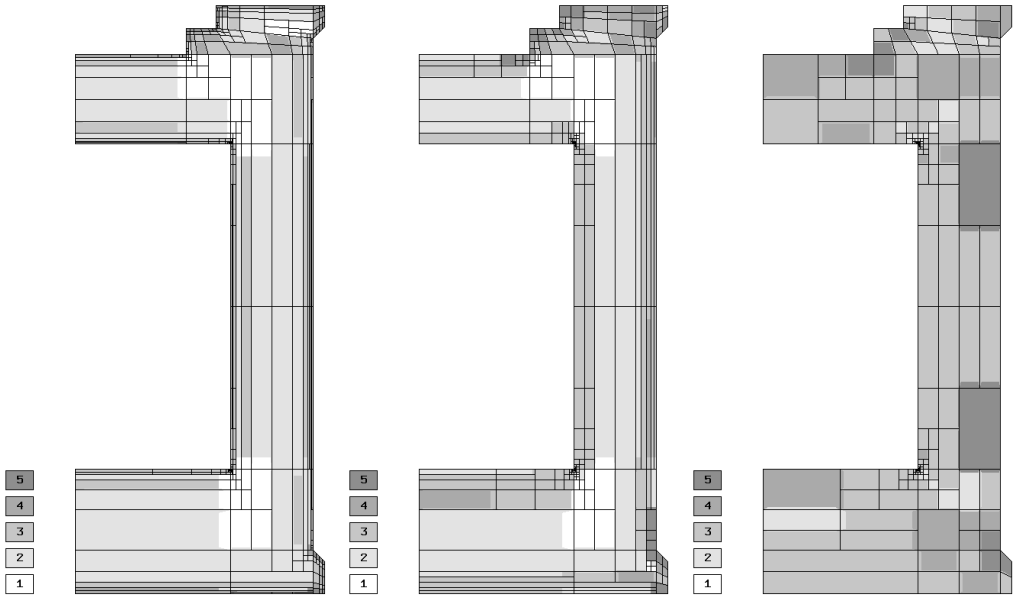



Figure 4.28: Multimesh *hp*-FEM (moisture mesh after 1 month, 1 year, 30 years)

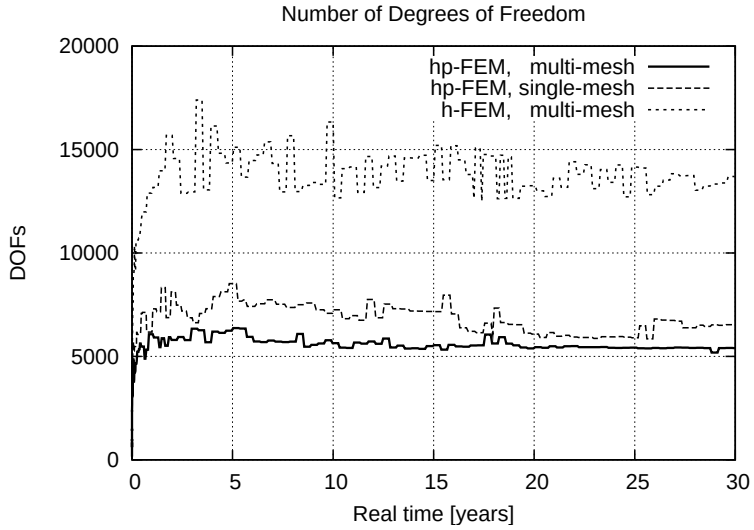



Figure 4.29: Comparison of the three methods in terms of degrees of freedom (DOF).

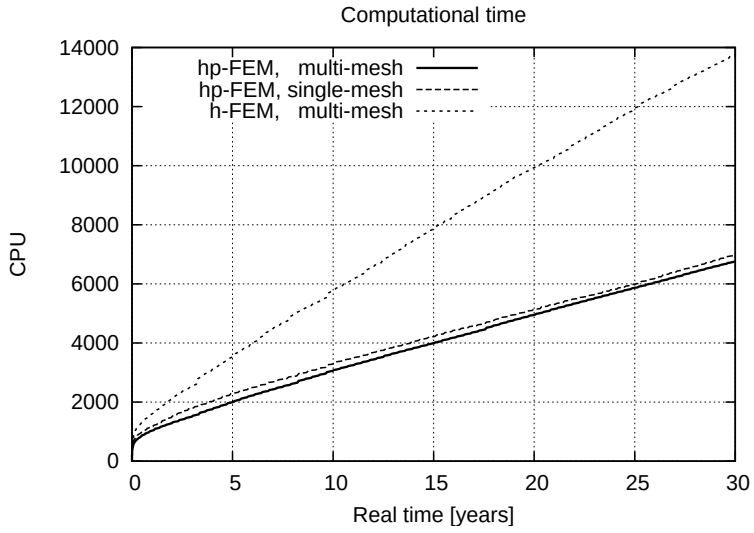



Figure 4.30: Comparison of the three methods in terms of CPU time.

previous time level are used in coupling terms. Then, on the same mesh the problem was solved as fully coupled. In both computations the second order time discretization was used. The experiment was obtained over the time period $(0, 1 \text{ month})$, where the temperature is rapidly changing, having influence on the moisture. From Fig. 4.31 we see that using simple linearization of coupling terms, the order of convergence for the moisture is reduced, while fully coupled problem achieved expected second order accuracy.

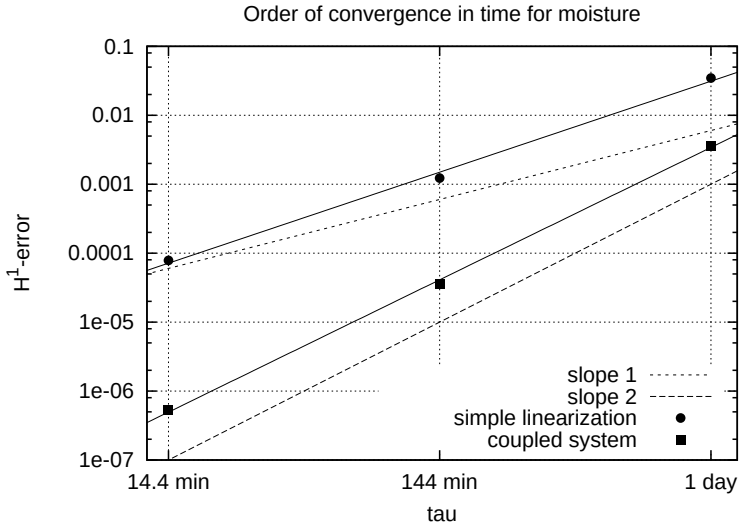



Figure 4.31: Comparison of the order of the convergence in time for operator splitting scheme and monolithical approach.

Chapter 5

# Automatic adaptivity for time-dependent problems

In the previous chapters we were concerned with the stationary problems. Let us now discuss automatic adaptivity for transient problems. Time-dependent coupled problems are challenging since one has to capture transient phenomena with sufficient accuracy, while the size of the problem must stay reasonably small. This leads to an obvious need for dynamically changing meshes for such problems. In most scientific computations, where dynamical meshes are used for time-dependent problems, data-transfer methods are necessary to move solution values between meshes for different time levels. As shown in Section 4.2, data-transfer methods cause additional error and in case of time-dependent problems repeated transfers (usually simple interpolation) between meshes can have disastrous consequences.

In this thesis we propose a novel approach for the solution of transient problems using dynamical *hp*-meshes obtained fully automatically by the *hp*-adaptive algorithm. In this method no transfers between non-matching meshes are necessary as a result of the multimesh approach described in Section 4.4. This technique allows us to integrate weak forms with integrands from different finite element spaces (i.e. discretized on geometrically different meshes). With the automatic *hp*-adaptivity and dynamical meshes the question of coarsening meshes between time levels arises. Mesh derefinement is particularly important in problems where sharp fronts (internal layers) move through the domain leaving smooth solutions behind them. We propose an original coarsening algorithm suitable for *hp*-FEM based on the *super-coarse solution*, which results in substantially fewer adaptive iterations.

The adaptive *hp*-FEM algorithm for time-dependent problems we use is obtained by combining the classical Rothe's method for time discretization with adaptive *hp*-FEM for the space discretization. The Rothe's method is a natural counterpart of the widely used Method of Lines (MOL). Recall that the MOL performs discretization in space while keeping the time variable continuous, which leads to a system of ODEs in time.

The Rothe's method, on the contrary, preserves the continuity of the spatial variable while discretizing time. In every time step, an evolutionary PDE is approximated by means of one or more time-independent ones. For one step methods (such as implicit Runge-Kutta), the number of the time-independent equations per time step is proportional to the order of accuracy of the time discretization method. For example, when employing the implicit Euler method, one has to solve one time-independent PDE per time step:

$$\frac{\partial u}{\partial t} = F(t,u) \quad \Rightarrow \quad \frac{u^{n+1}-u^n}{\Delta t} = F(t^{n+1},u^{n+1}) \tag{5.1}$$

The Rothe's method is fully equivalent to the MOL if no adaptivity in space or time takes place, but it provides a better setting for the application of spatially adaptive algorithms. The spatial discretization error can be controlled by solving the time-independent equations adaptively, and the size of the time step can be adjusted using standard ODE techniques [12, 24, 25].

The chapter is organized as follows: In Section 5.1, the multimesh technique is discussed from the point of view of time-dependent problems. In Section 5.2 we present the refining and coarsening algorithms for such problems and Section 5.3 demonstrates our approach on a numerical example.

## 5.1 Dynamical meshes and the multimesh technique

By $\mathcal{T}_m$ let us denote a uniform coarse mesh covering the computational domain $\Omega$. This mesh (called *master mesh*) is shared by all solution components at all time levels, in other words all meshes can be obtained from this mesh by elementary refinements. At each time instant $t_n$ an optimal mesh is found to suit the best the solution $\boldsymbol{u}^n(\boldsymbol{x})$. On the $(n+1)$st time level, the approximated solution $\boldsymbol{u}^n(\boldsymbol{x})$, that has been obtained in the previous time step, is used as data. Note, however, that $\boldsymbol{u}^n$ is defined on a locally refined mesh that was created automatically during the $n$th time step, while the unknown $\boldsymbol{u}^{n+1}$ is solved adaptively starting from a coarser mesh. As a result, the meshes obtained on each time level are different, i.e., the mesh changes dynamically in time.

In order to evaluate exactly the integrals in the discrete formulation of the problem (5.1) when the previous solution $u^n(\boldsymbol{x})$ and the test function $v^{n+1}(\boldsymbol{x})$ are defined on geometrically different meshes, we use the *multimesh hp*-FEM described in Section 4.4. Example of a master mesh $\mathcal{T}_m$, meshes $\mathcal{T}_n, \mathcal{T}_{n+1}$ and the union mesh $\mathcal{T}_u$ (mesh containing refinements from both subsequent time levels) for time-dependent problem are depicted in Fig. 5.1.

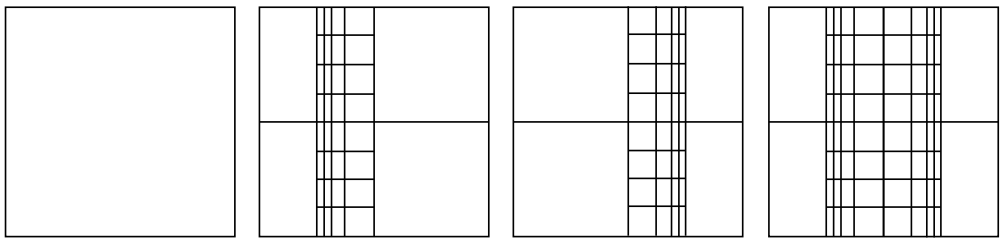


Figure 5.1: Example of a master mesh $\mathcal{T}_m$, meshes $\mathcal{T}_n, \mathcal{T}_{n+1}$ and the union mesh $\mathcal{T}_u$.

Note, that by using multimesh technology no additional error arises since no transfer of information between the meshes takes place. In other words, all integrals in the weak formulation are evaluated exactly up to the error in the numerical quadrature.

## 5.2 $hp$-adaptivity for transient problems

As mentioned above, in transient problems the optimal meshes are on each time level obtained automatically and they can change from one time step to another, which induces need for both refining and coarsening strategies.

### 5.2.1 Refining algorithm

In Chapter 3 the goal of adaptive algorithm was to decrease the error as low as possible. On the other hand for time-dependent problems we want to sustain the space error on approximately the same level, which would result into meshes with smoothly changing number of degrees of freedom. Thus the stopping criterion for $hp$-adaptivity will slightly differ. In adaptivity for time-independent problems we refined in each adaptive step fixed amount of elements (e.g., 30%). For time-dependent problems the amount of elements refined in one adaptive iteration will also depend on the global solution error estimate in that iteration. When refining elements, we sum the element errors of already processed (refined) elements. This sum indicates approximately how much the global error will decrease. And when this number substantially exceeds the difference between the original global error estimate and the prescribed tolerance

$$\sum_{j=0}^{M} err_j > c\ (err_{est} - TOL), \tag{5.2}$$

we stop the procedure and continue with the next adaptive iteration even though we did not refined prescribed fixed amount of elements. In equation (5.2) $c \geq 1$ is a suitable constant, $err_i$ are element contributions to the global error estimate sorted in decreasing order and $M$ is number of already processed elements. In this way in the

next adaptive iteration the global tolerance is satisfied with as less degrees of freedom as possible and we proceed to the next time level.

### 5.2.2 Coarsening algorithm

The most primitive strategy to coarsen the mesh on the next time level is to start on each level from the very coarse (master) mesh and perform the automatic adaptivity to find the optimal mesh for particular solution $\boldsymbol{u}^n(\boldsymbol{x})$. Although this would result in the most optimal meshes in each iteration, it is not virtually possible due to the immense computational time demands. Moreover, a lot of work would be wasted since the solutions from two adjacent time steps usually diverge mildly even though the solution changes significantly in the whole time domain. Global derefinement would result in similar difficulties.

We present a coarsening algorithm that prepares the mesh for the adaptive algorithm on the next time level by local derefinements. Thus, we remove only unnecessary refinements from previous time levels. In this way the meshes for particular time steps are suboptimal, but the number of adaptive iterations performed in each time step significantly decreases.

In higher-order finite element method we have two questions. First question is which elements can be coarsened and second how to coarsen them. In *hp*-FEM we can either decrease the polynomial order of the element or if it has four active sons we can geometrically coarsen the element with various choices for assigned polynomial order. Situation is depicted in Fig. 5.2.

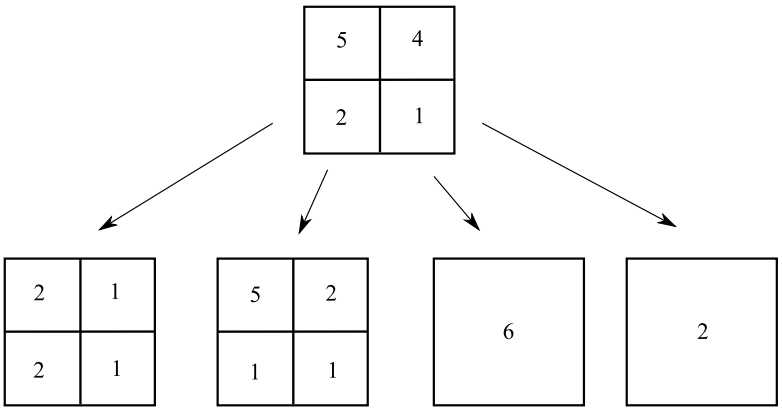



Figure 5.2: Coarsening choices for an element.

However, the coarsening serves only to prepare the mesh for the next adaptive process, thus we do not have to search for the best unrefinement as in case of refinements. It is perfectly sufficient to perform any coarsening that does not cause error increase and

the mesh will optimize itself in the next adaptive loop. So far, to keep the algorithm reasonably simple we allow only coarsening as reversing of previous refinements of isotropically refined elements (both triangles and quadrilaterals). In this way the tree-like structure of meshes starting from the *master mesh* is preserved. Unfortunately, the coarsening algorithm cannot be just an opposite to the refining. While with the refining algorithm we are seeking for the best candidate in the sense of error decrease, here we are looking for a candidate whose error will not exceed some tolerance (so that it would not be subsequently refined).

Let us denote by $e_{max}$ the error of the element with maximal error from the last adaptive iteration. From Alg. 3 from Chapter 3 we know that errors of all elements are below $e_{max}$ and that it should be satisfied as well after the coarsening procedure. First we calculate so called *super-coarse solution* for polynomial orders. By that we mean that polynomial order on all elements is decreased by one. We calculate error estimates for all elements in the super-coarse mesh with respect to the reference solution and lower the polynomial order on those elements whose error after coarsening is less than $k * e_{max}$, where parameter $k \in (0, 1)$ is chosen to ensure that element will not be subsequently refined in the adaptive procedure on the next time level. Similar procedure is run for a spatial coarsening – *super-coarse solution* is calculated on the mesh where all elements with four active subelements are coarsened and polynomial order on such elements is assigned to be the maximum of orders on four subelements. In a similar way as before we determine which elements can be also spatially unrefined without significant increase of the error.

The whole procedure for time-dependent problems with the refining and coarsening strategy is described in Alg. 5. Effectivity of the approach is demonstrated on the flame propagation problem in the next section.

## 5.3 Flame propagation problem

Let us recall Example 4.2 from Section 4.3. All parameters' values are taken from [37]. The behavior of both solution components, the temperature and concentration, rapidly changes throughout the time as the laminar flame propagates through the domain. Function $\omega(\theta, Y)$ representing the reaction rate at different time instants is depicted in Fig. 5.3.

These problems are typically solved on an uniformly refined mesh resulting in huge computational and memory requirements. The automatic $hp$-adaptive Alg. 5 with both refining and coarsening was applied resulting in an optimal discretization with as few degrees of freedom as possible. Optimal $hp$-meshes for four different time levels are shown in Fig. 5.4. Notice that very small elements on the flame front are often adjacent to very large elements. This is possible due to the technique of arbitrary-level hanging nodes described in Section 3.1, and for problems with sharp fronts or

**Algorithm 5**: Adaptive algorithm for time-dependent problems with improved stopping criterion.

```
foreach time level do
    repeat
        compute solution on current mesh u;
        compute reference solution u_ref;
        evaluate global error estimate err_gl and local errors on elements err_i;
        if err_gl < TOL then
            done = true;
            break;
        else
            sort all elements by their error;
            processed error = 0;
            foreach element do
                if (processed error > c * (err_gl - TOL)) or (err_i < k * err_max) then
                    done = true;
                    break;
                else
                    find optimal refinement;
                    processed error += err_i;
    until not done ;
    calculate super-coarse solution for polynomial orders;
    decrease polynomial orders when possible;
    calculate super-coarse solution for spatial refinements;
    geometrically coarsen elements when possible;
```

curvilinear material interfaces, this saves large amounts of degrees of freedom which otherwise would be needed to keep the mesh regular.

For illustration purposes we also show in Fig 5.5 underlying low-order FEM-meshes where biquadratic elements and the *h*-adaptive strategy are used. Comparison of the number of DOFs for both approaches is in Fig. 5.6. It can be seen that low-order FEM required on average 4-5 times more degrees of freedom than *hp*-FEM. It is worth mentioning that with lowest-order (bilinear) *h*-FEM the size of the problem would either exceeds memory capabilities or we would not reach prescribed accuracy.

The effectiveness of the adaptive process with refinement and derefinement is demonstrated in Fig. 5.7, where the number of adaptive iterations per one time iteration is shown. As a result of optimized adaptive algorithm in almost two thirds of all time

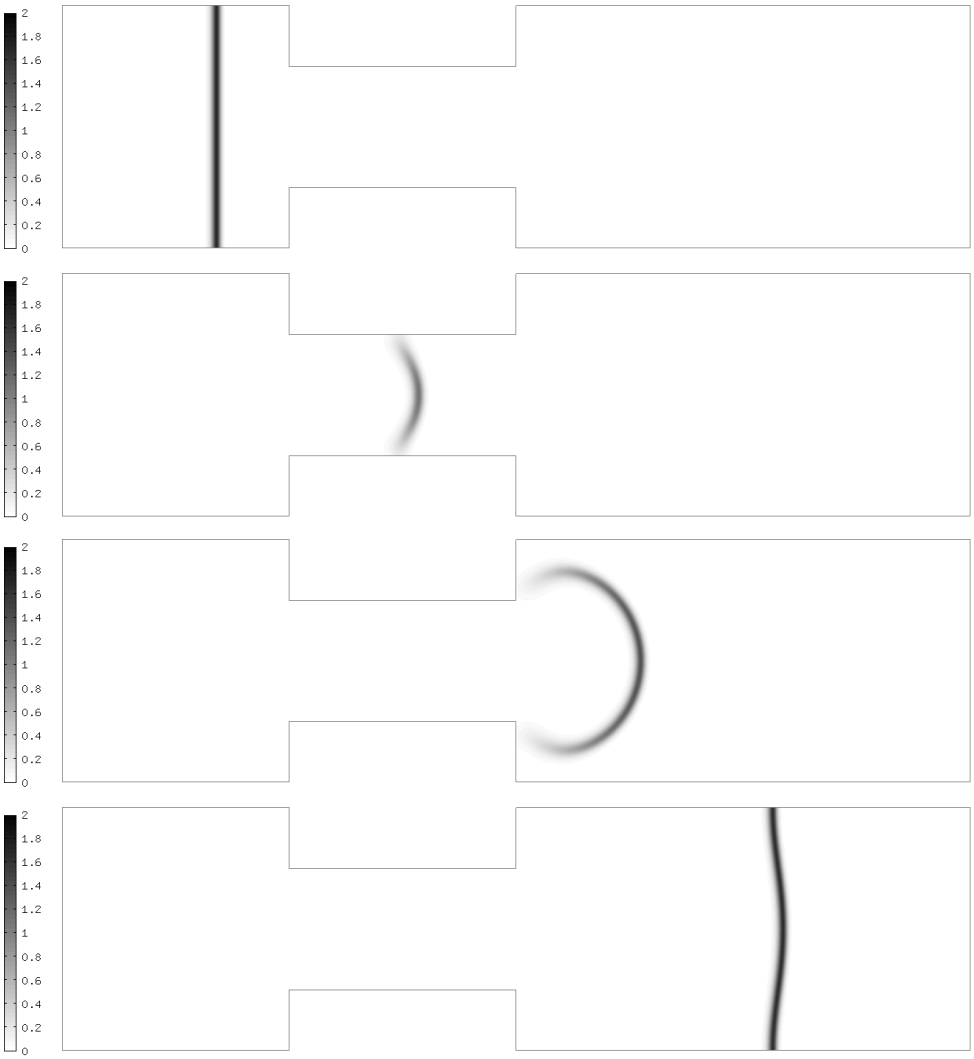



Figure 5.3: Reaction rate $\omega(\theta, Y)$ at $t = 1.37,\ 19,\ 47.4,\ 59$.

levels no adaptivity is necessary and the solution satisfies prescribed tolerance immediately. In about one quarter of all time steps one adaptive loop is done (problem is solved twice) and in negligible number of steps more adaptive iterations have to be applied.

As was said at the beginning of this chapter the multimesh assembling technology allows us to solve problems on dynamically changing meshes without need for the data-transfer methods, thus on each time level the discrete formulation is evaluated accurately up to the error of the numerical quadrature. Appling data-transfer methods such as interpolation on each time level would cause an additional error which can after repeated transfers exceed the error caused by space or time discretization.

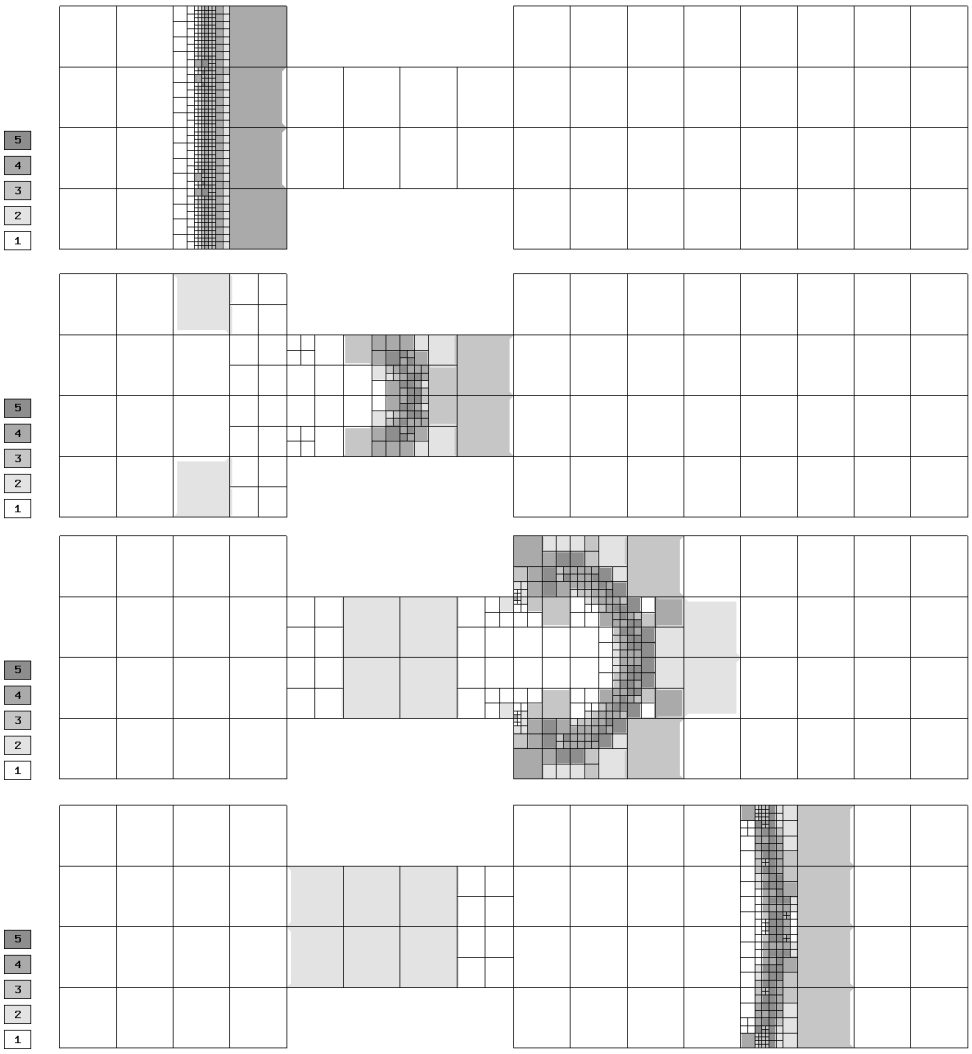



Figure 5.4: $hp$-meshes for time levels $t = 1.37,\ 19,\ 47.4,\ 59$.

For illustration, we computed the problem on dynamically changing meshes with bilinear elements using both, bilinear interpolation between time steps and the multimesh assembling technique. In both computations the meshes were adapted using $h$-adaptivity to achieve accuracy 4% in relative $H^1$-error in $\omega(\theta, Y)$ (the error tolerance was lowered in comparison to the previous computations). In the first case, the solution from the level $t^n$ was transferred from the mesh $\mathcal{T}_n$ onto the mesh $\mathcal{T}_{n+1}$ and then solution was sought without need for additional treatment - all functions (basis functions and external data) were defined on the same mesh. In the second case, no interpolation took place, but integrals containing solution $(\theta^n, Y^n)$ were assembled using the multimesh technique. Both computations were compared with "reference

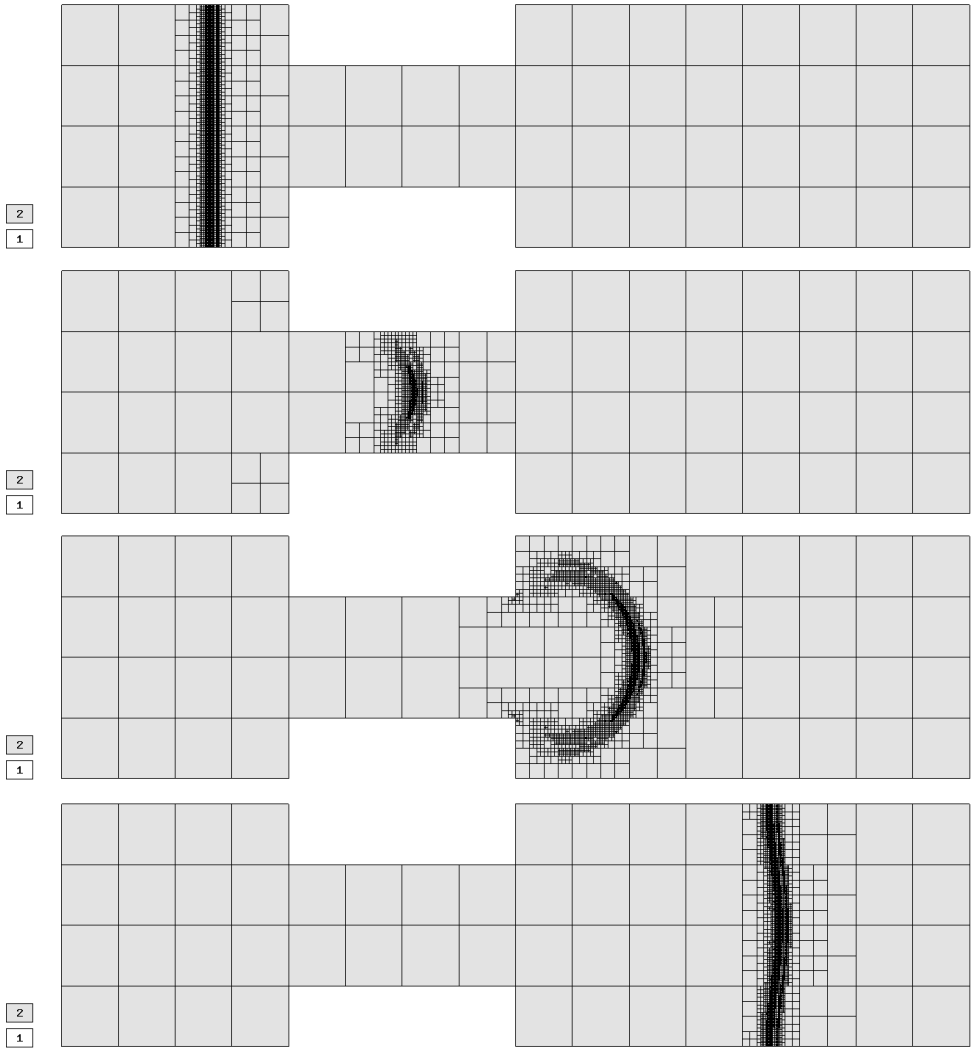



Figure 5.5: $hp$-meshes for time levels $t = 1.37,\ 19,\ 47.4,\ 59$.

computation", when very fine meshes were used. Fig. 5.8 shows the impact of data-transfers on the approximation error

$$e_n = \frac{||\omega(\theta^n, Y^n) - \omega_{ref}||_{H^1(\Omega)}}{||\omega^n_{ref}||_{H^1(\Omega)}}.$$

With increasing length of the time interval the error caused by interpolation accumulates, while the approximation error of the solution obtained by the multimesh stays on the same level.

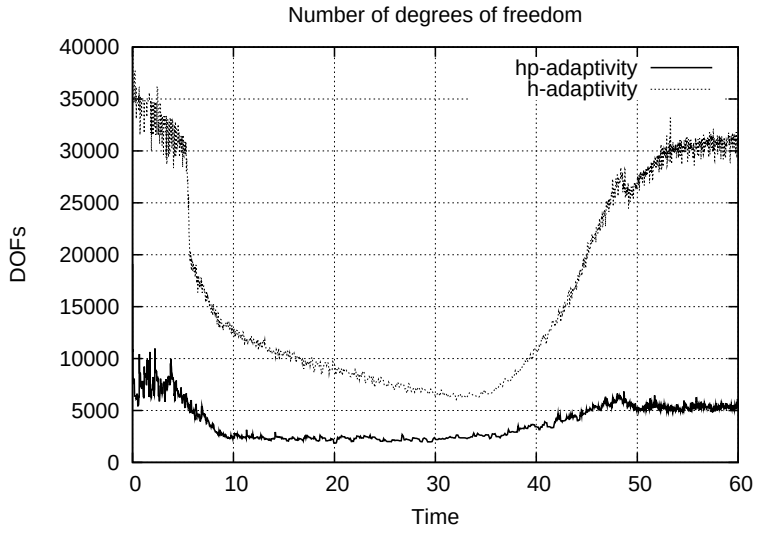



Figure 5.6: Comparison of discrete problem size as a function of time.

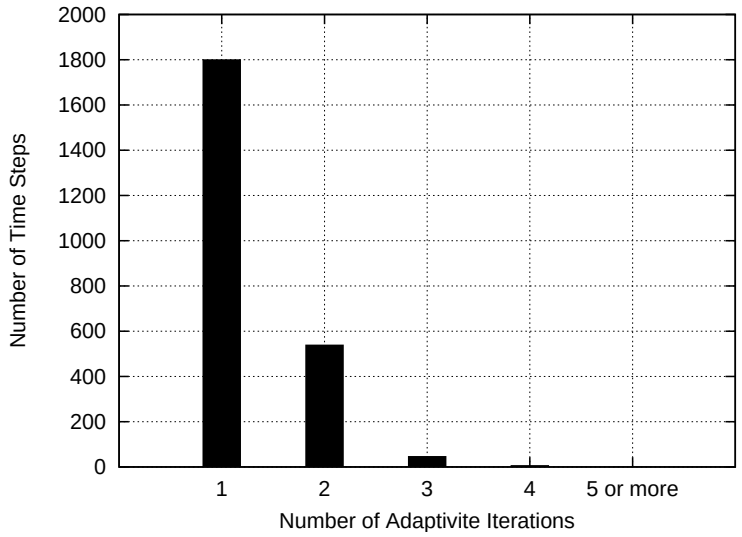



Figure 5.7: Number of adaptive iterations per time step.

Reaction rate $\omega(\theta(t), Y(t))$ at $t = 45$ for both computations and also the "reference" solution $\omega_{ref}$ is depicted in Fig. 5.9. With correspondence to Fig. 5.8 the data-transfers between non-matching meshes cause slowdown of the propagating flame (reference vertical line is inserted for easier comparison).

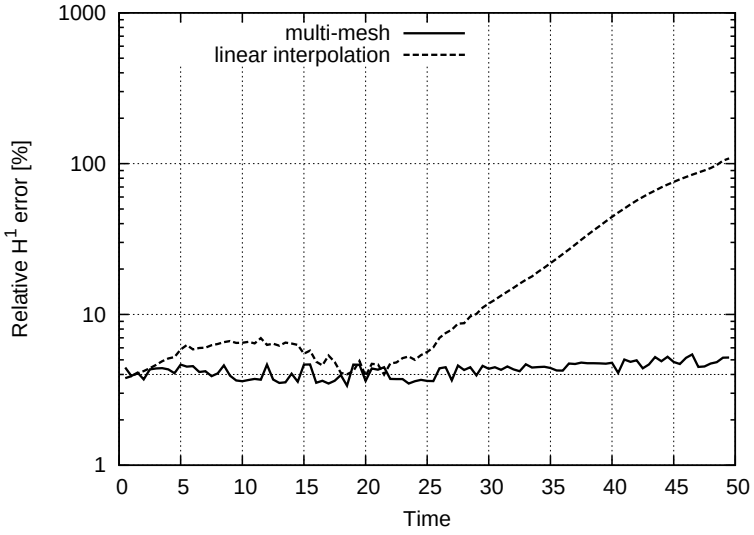



Figure 5.8: Effect of linear interpolation between meshes $\mathcal{T}_n$ and $\mathcal{T}_{n+1}$ on approximation error.

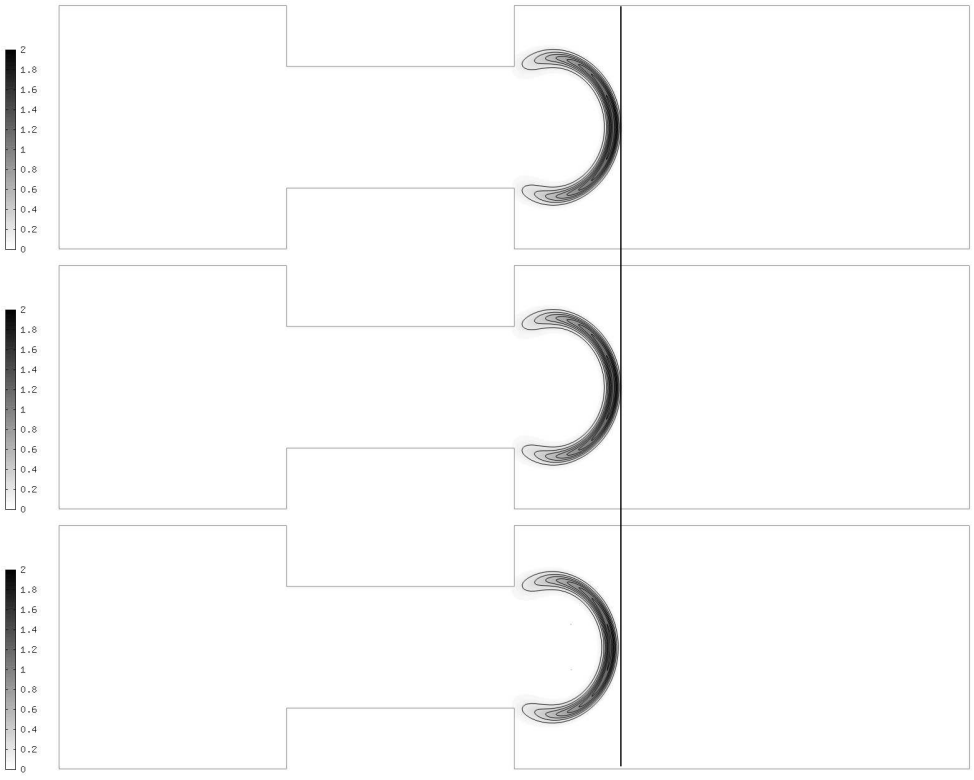


Figure 5.9: Reaction rate $\omega$ – "reference" (top), multimesh (middle), interpolation (bottom).

Chapter 6

# Inductively heated incompressible flow of electrically conductive liquid in pipe

Although the laws of electromagnetism and fluid flow were known already in the nineteenth century, the complete mathematical description of the magnetofluiddynamic (MFD) phenomena comes only from the 1930s, when these effects started to be studied in connection with astrophysics (particularly the behavior of plasma in the universe) and a little later also with geophysics (origin of the Earth's magnetic field). Since then, a lot of systematic effort has been exerted to solve the model, first analytically for geometrically very simple arrangements and later, since 1970s, also numerically, even for very sophisticated cases.

From the physical and mathematical points of view probably nothing new can be added to the model that is considered a very good approximation of the physical reality and its validity has been confirmed many times [1, 9]. On the other hand, its solution is a real problem. The model can usually be reduced to three nonstationary and nonlinear partial differential equations describing the time evolution of electromagnetic, temperature, and flow fields, and one equation of continuity. However, these equations are coupled, because their coefficients (expressed in terms of the physical parameters of involved materials and media) are functions of the temperature, pressure, and other possible state variables.

Numerous industrial technologies working with electrically conductive liquids (e.g., molten metals, acids or solutions of salts) are based on the force and thermal effects of the electromagnetic field. As such processes we can mention pumping, dosing, stirring or heating. The computer modeling of such processes is still a challenge and in engineering community these problems are usually solved by the standard low-order finite element methods (FEM) or finite volume methods (FVM) with one fixed

common mesh (usually uniformly refined) for all components of the problem and for all time levels [1, 27, 33].

In this chapter we are dealing with the problem of heating of a conductive liquid flow in a ceramic pipe. Fig. 6.1 shows the axisymmetric settings of the problem, where the electrically conductive liquid flows from the left to the right and is warmed up by a time-variable magnetic field generated by a harmonic current carrying inductor.

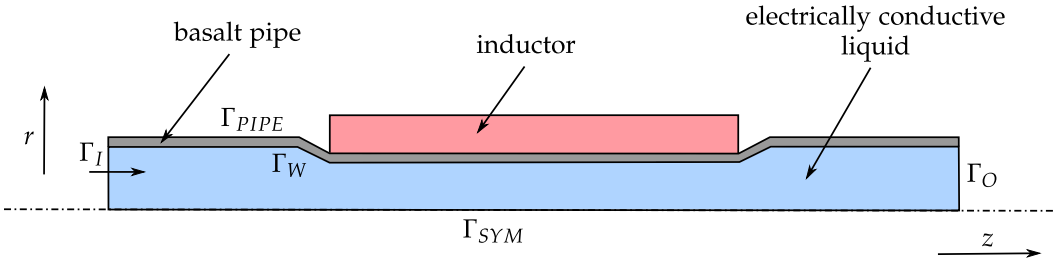



Figure 6.1: Computational domain $\Omega$.

## 6.1 Mathematical model

The mathematical model of the problem is given by three partial differential equations describing the distribution of the magnetic field, the temperature field and the flow field represented by radial and axial velocities and the pressure.

### 6.1.1 Magnetic field in conducting fluid

First, let us describe the movement of the conducting fluid in external magnetic field. Conducting fluid moving at the velocity $\boldsymbol{u}$ in external magnetic field $\boldsymbol{B}$ produces in it (in accordance with Faraday's law) electric field.

If there exists, moreover, an additional external electric field $\boldsymbol{E}$ acting on the fluid, then the electric field caused by the motion of the fluid $\boldsymbol{E}_m$ superimposes to it, thus producing total electric field

$$\boldsymbol{E}_t = \boldsymbol{E}_m + \boldsymbol{E} = \boldsymbol{u} \times \boldsymbol{B} + \boldsymbol{E}.$$

This total field makes charged particles of the fluid move in its direction, which leads to the generation of currents. Denoting electrical conductivity of the fluid by symbol $\gamma$, the current density at a point is given by the formula

$$\boldsymbol{J} = \gamma \boldsymbol{E}_t = \gamma(\boldsymbol{u} \times \boldsymbol{B} + \boldsymbol{E}). \tag{6.1}$$

These currents generate secondary magnetic field that adds to the original external magnetic field $\boldsymbol{B}$. Thus, every particle of the fluid is affected by the Lorentz' force, whose volume density vector $\boldsymbol{f}$ is given by

$$\boldsymbol{f} = \boldsymbol{J} \times \boldsymbol{B} = \gamma(\boldsymbol{u} \times \boldsymbol{B} + \boldsymbol{E}) \times \boldsymbol{B}, \tag{6.2}$$

and local Joule's losses are produced, whose volume value $p_J$ is

$$p_J = \frac{\boldsymbol{J} \cdot \boldsymbol{J}}{\gamma}.$$

The best way of describing magnetic field in the conducting fluid is to use the magnetic vector potential $\boldsymbol{A}$ defined by

$$\boldsymbol{B} = \mathrm{curl}\boldsymbol{A},$$

with Coulomb's calibration condition

$$\mathrm{div}\boldsymbol{A} = 0.$$

Now the first Maxwell's equation

$$\mathrm{curl}\boldsymbol{H} = \boldsymbol{J} + \frac{\partial \boldsymbol{D}}{\partial t},$$

where $\boldsymbol{H}$ is magnetic field intensity and $\boldsymbol{D}$ electric induction, may be after neglecting the displacement currents (the second term on the right-hand side) rewritten into the form

$$\mathrm{curl}(\mu^{-1}\mathrm{curl}\boldsymbol{A}) = \boldsymbol{J}, \tag{6.3}$$

where $\mu$ denotes the magnetic permeability. From the second Maxwell's equation and equation (6.1) we obtain

$$\mathrm{curl}\boldsymbol{E} = -\frac{\partial \boldsymbol{B}}{\partial t} \quad \Rightarrow \quad \mathrm{curl}\left(\frac{\boldsymbol{J}}{\gamma} - \boldsymbol{u} \times \boldsymbol{B}\right) = -\frac{\partial \mathrm{curl}\boldsymbol{A}}{\partial t} = -\,\mathrm{curl}\left(\frac{\partial \boldsymbol{A}}{\partial t}\right),$$

and hence,

$$\frac{\boldsymbol{J}}{\gamma} - \boldsymbol{u} \times \boldsymbol{B} = -\frac{\partial \boldsymbol{A}}{\partial t} - \nabla\varphi,$$

where $\varphi$ is a scalar function (mostly interpreted as the scalar electric potential).

Multiplying this expression by electrical conductivity $\gamma$ we get

$$\boldsymbol{J} = -\gamma\frac{\partial \boldsymbol{A}}{\partial t} - \gamma\nabla\varphi + \gamma\boldsymbol{u} \times \mathrm{curl}\boldsymbol{A}. \tag{6.4}$$

Substituting (6.4) into (6.3) finally provides us with

$$\mathrm{curl}(\mu^{-1}\mathrm{curl}\boldsymbol{A}) = -\gamma\nabla\varphi - \gamma\left(-\frac{\partial \boldsymbol{A}}{\partial t} - \boldsymbol{u}\times\mathrm{curl}\boldsymbol{A}\right). \tag{6.5}$$

Now it remains to explain the physical significance of the individual terms in (6.5):

- $-\gamma\nabla\varphi = \boldsymbol{J}_{\mathrm{ext}}$ represents external current density delivered to the fluid from the external source,
- $-\gamma\dfrac{\partial \boldsymbol{A}}{\partial t}$ stands for the eddy current density due to time variation of the magnetic field,
- $\gamma\boldsymbol{u}\times\mathrm{curl}\boldsymbol{A}$ is the eddy current density due to the motion of the fluid.

It is clear that only external current density is known beforehand. Both eddy current densities are functions of the magnetic vector potential $\boldsymbol{A}$ (and the second one also a function of the velocity $\boldsymbol{u}$). That is the reason why the corresponding terms are put on the left-hand side

$$\mathrm{curl}(\mu^{-1}\mathrm{curl}\boldsymbol{A}) + \gamma\left(\frac{\partial \boldsymbol{A}}{\partial t} - \boldsymbol{u}\times\mathrm{curl}\boldsymbol{A}\right) = \boldsymbol{J}_{\mathrm{ext}}\,. \tag{6.6}$$

If the system under investigation contains no ferromagnetic part, then

$$\mathrm{curl}(\mathrm{curl}\boldsymbol{A}) + \gamma\mu\left(\frac{\partial \boldsymbol{A}}{\partial t} - \boldsymbol{u}\times\mathrm{curl}\boldsymbol{A}\right) = \mu\boldsymbol{J}_{\mathrm{ext}}\,. \tag{6.7}$$

Considering the magnetic vector potential $\boldsymbol{A}$ harmonic in time with the angular frequency $\omega$

$$\boldsymbol{A}(\boldsymbol{x},t) = Re(\underline{\boldsymbol{A}}(\boldsymbol{x})e^{-i\omega t}),$$

where $i$ is an imaginary unit and the underlined $\underline{\boldsymbol{A}}$ is the phasor (generally complex vector), it follows that

$$\frac{\partial \boldsymbol{A}}{\partial t} = -i\omega\underline{\boldsymbol{A}}.$$

Thus

$$\mathrm{curl}(\mathrm{curl}\underline{\boldsymbol{A}}) + i\omega\gamma\mu\underline{\boldsymbol{A}} - \gamma\mu\,\boldsymbol{u}\times\mathrm{curl}\underline{\boldsymbol{A}} = \mu\underline{\boldsymbol{J}}_{\mathrm{ext}}\,, \tag{6.8}$$

where $\underline{\boldsymbol{J}}_{\mathrm{ext}}$ stands for the phasor of the external harmonic current density in the inductor. The boundary conditions on the axis of symmetry and on sufficiently distant artificial boundary are characterized by the Dirichlet boundary condition

$$\underline{\boldsymbol{A}} = \boldsymbol{0}. \tag{6.9}$$

### 6.1.2 Temperature field in conducting fluid

The distribution of the temperature in the fluid is generally a function of the position and time. In other words we can write $T = T(\boldsymbol{r}, t)$, where $\boldsymbol{r}$ denotes the position vector. Provided that $\lambda$ is specific thermal conductivity, the heat flux $\boldsymbol{q}(\boldsymbol{r}, t)$ is defined as

$$\boldsymbol{q}(\boldsymbol{r}, t) = \lambda \nabla T(\boldsymbol{r}, t). \tag{6.10}$$

Let us consider a small part of the conducting incompressible fluid of a volume $V$ closed by surface $S$. The power balance for this volume (according to the first law of thermodynamics) reads

$$P(t) = \frac{\mathrm{d}U}{\mathrm{d}t} + Q(t), \tag{6.11}$$

where $\mathrm{d}U/\,\mathrm{d}t$ denotes the time variation of the internal energy in the volume that is produced by the heat flow $Q(t)$ penetrating to the volume $V$ through its boundary $S$ and by the heat power $P(t)$ produced by the Joule losses. The internal energy $U$ in the volume $V$ can be expressed by the formula

$$U = \int_V \rho c T \,\mathrm{d}V, \tag{6.12}$$

where $\rho$ is the specific mass of the fluid and $c$ denotes its specific heat. Since $\rho$ and $c$ may be declared constants (incompressible flow with small variations of the temperature), we can write

$$\frac{\mathrm{d}U}{\mathrm{d}t} = \rho c \int_V \frac{\mathrm{d}T}{\mathrm{d}t} \,\mathrm{d}V. \tag{6.13}$$

The total heat flow $Q(t)$ may be determined from the heat flux $\boldsymbol{q}(\boldsymbol{r}, t)$ integrated over the whole surface $S$. Taking advantage of the Gauss theorem, it holds that

$$Q(t) = \int_S \boldsymbol{q} \,\mathrm{d}S = \int_S \boldsymbol{q} \cdot \boldsymbol{n} \,\mathrm{d}S = \int_V \mathrm{div}\boldsymbol{q} \,\mathrm{d}V. \tag{6.14}$$

Finally, the total Joule losses in the volume $V$ are given by the integral

$$P(t) = \frac{1}{\gamma} \int_V |\boldsymbol{J}|^2 \,\mathrm{d}V, \tag{6.15}$$

where the current density $J$ is given by (6.1). Substituting (6.13), (6.14) and (6.15) into (6.11) gives

$$\frac{1}{\gamma} \int_V |\boldsymbol{J}|^2 \,\mathrm{d}V = \rho c \int_V \frac{\mathrm{d}T}{\mathrm{d}t} \,\mathrm{d}V + \int_V \mathrm{div}\boldsymbol{q} \,\mathrm{d}V.$$

Since this equation must hold for any volume $V$, there must hold also the equality of

the relevant integrands

$$\frac{|\boldsymbol{J}|^2}{\gamma} = \rho c \frac{\mathrm{d}T}{\mathrm{d}t} + \mathrm{div}\boldsymbol{q},$$

and after substituting for $\boldsymbol{q}$ from (6.10) and some rearrangements we obtain

$$\mathrm{div}(\lambda \nabla T) - \rho c \frac{\mathrm{d}T}{\mathrm{d}t} = -\frac{|\boldsymbol{J}|^2}{\gamma}.$$

The remaining operation to be carried out is to perform the full derivative $\mathrm{d}T/\,\mathrm{d}t$. We easily get

$$\frac{\mathrm{d}T}{\mathrm{d}t} = \frac{\partial T}{\partial t} + \frac{\partial T}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial T}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial T}{\partial z}\frac{\partial z}{\partial t} = \frac{\partial T}{\partial t} + \boldsymbol{u} \cdot \nabla T.$$

Thus, the equation describing the nonstationary temperature field in the fluid moving at the velocity $\boldsymbol{u}$ reads

$$\mathrm{div}(\lambda \nabla T) - \rho c \left( \frac{\partial T}{\partial t} + \boldsymbol{u} \cdot \nabla T \right) = -\frac{|\boldsymbol{J}|^2}{\gamma}, \tag{6.16}$$

where time average internal source of heat (the specific Joule losses) is determined by the formula

$$\frac{|\boldsymbol{J}|^2}{\gamma} = \omega^2 \gamma \, |\underline{\boldsymbol{A}}|^2.$$

Note that in the ceramic pipe the term $\boldsymbol{u} \cdot \nabla T$ vanishes. The boundary conditions (see Fig. 6.1 for notation) equipped with equation (6.16) are as follows

$$\begin{aligned} T &= T_D && \text{on } \Gamma_I, \\ \frac{\partial T}{\partial \boldsymbol{n}} &= \alpha(T - T_{ext}) && \text{on } \Gamma_{PIPE}, \\ \frac{\partial T}{\partial \boldsymbol{n}} &= 0 && \text{elsewhere,} \end{aligned}$$

where $\alpha$ denotes the heat transfer coefficient.

### 6.1.3 Flow field in conducting fluid

Liquid metals may be with practically negligible error considered incompressible, i.e., their density can be supposed constant, and thus the flow is governed by Navier-Stokes equations

$$\rho \left[ \frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} \right] = -\nabla p + \nu \Delta \boldsymbol{u} + \rho \boldsymbol{g} + \boldsymbol{f}, \tag{6.17}$$

together with the continuity equation

$$\mathrm{div}\boldsymbol{u} = 0, \tag{6.18}$$

where $p$ is the pressure, $\nu$ dynamic viscosity and $\boldsymbol{f}$ denotes the internal volume Lorentz forces produced by the time variable magnetic field and defined by (6.2). Navier-Stokes equations have been derived in detail in numerous monographs, we refer the reader for instance to [18]. For numerical computations we neglect the gravitational term $\rho\,\boldsymbol{g}$ in order to preserve axial symmetry of the problem. The equations are equipped with the following boundary conditions

$$\begin{aligned} v_r = 0,\ v_z = v_I \qquad & \text{on } \Gamma_I, \\ (p - p_{ref})\,\boldsymbol{n} + \frac{1}{2}(\boldsymbol{u}\cdot\boldsymbol{n})^-\,\boldsymbol{u} = \nu\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} \qquad & \text{on } \Gamma_O, \\ \boldsymbol{u} = \boldsymbol{0} \qquad & \text{on } \Gamma_W, \\ v_r = 0,\ \frac{\partial v_z}{\partial r} = 0 \qquad & \text{on } \Gamma_{SYM}. \end{aligned}$$

## 6.2 Rothe's method for time discretization

Let us first describe the discretization in time. For simplicity, we consider uniform partition of the time domain $(0,T)$ with time step $\tau$, hence $(0,T) = (0 = t_0, t_1, \ldots, t_N = T),\ t_n = n\tau$. Then we approximate all components of the solution

$$\begin{aligned} \underline{\boldsymbol{A}}(t_n) &\approx \underline{\boldsymbol{A}}^n, & \boldsymbol{u}(t_n) &\approx \boldsymbol{u}^n, \\ T(t_n) &\approx T^n, & p(t_n) &\approx p^n, \end{aligned}$$

and we use the implicit second order scheme for discretization of time derivatives

$$\begin{aligned} \frac{\partial \boldsymbol{u}}{\partial t} &\approx \frac{3\boldsymbol{u}^n - 4\boldsymbol{u}^{n-1} + \boldsymbol{u}^{n-2}}{2\tau}, \\ \frac{\partial T}{\partial t} &\approx \frac{3T^n - 4T^{n-1} + T^{n-2}}{2\tau}. \end{aligned}$$

Resulting time-independent coupled problem on time level $t_n$ reads

$$\mathrm{curl}(\mathrm{curl}\underline{\boldsymbol{A}}^n) + i\omega\gamma\mu\underline{\boldsymbol{A}}^n - \gamma\mu\,\boldsymbol{u}^n \times \mathrm{curl}\underline{\boldsymbol{A}}^n = \mu\underline{\boldsymbol{J}}_{\mathrm{ext}}, \tag{6.19}$$

$$\mathrm{div}(\lambda\nabla T^n) - \rho c\left(\frac{3T^n - 4T^{n-1} + T^{n-2}}{2\tau} + \boldsymbol{u}^n\cdot\nabla T^n\right) = -\frac{|\boldsymbol{J}^n|^2}{\gamma}, \tag{6.20}$$

$$\rho\left[\frac{3\boldsymbol{u}^n-4\boldsymbol{u}^{n-1}+\boldsymbol{u}^{n-2}}{2\tau}+\boldsymbol{u}^n\cdot\nabla\boldsymbol{u}^n\right]=-\nabla p^n+\nu\Delta\boldsymbol{u}^n+\boldsymbol{f}^n, \tag{6.21}$$

$$\mathrm{div}\boldsymbol{u}^n=0. \tag{6.22}$$

## 6.3 Weak formulation of the problem

According to Chapter 2, the starting point for the discrete formulation of the problem is the weak formulation. Let us derive the weak formulation of the problem (6.19) – (6.22) in the axisymmetric arrangement. Let us introduce simplifying notation $\boldsymbol{A}=\underline{\boldsymbol{A}}^n$, $T=T^n$, $\boldsymbol{u}=\boldsymbol{u}^n$ and $p=p^n$.

### 6.3.1 Magnetic potential

First, we rewrite the vector equation (6.19) as a scalar equation for $A_\varphi$ under the assumption that $A_r=A_z=0$. Let us start with cylindrical formulas for necessary operators.

$$\begin{aligned}\mathrm{curl}\,\boldsymbol{A} &= \frac{1}{r}\begin{vmatrix}\vec{r}_0 & r\vec{\varphi}_0 & \vec{z}_0\\ \frac{\partial}{\partial r} & \frac{\partial}{\partial\varphi} & \frac{\partial}{\partial z}\\ 0 & rA_\varphi & 0\end{vmatrix}=\frac{1}{r}\left[\vec{z}_0\frac{\partial}{\partial r}(rA_\varphi)-\vec{r}_0\frac{\partial}{\partial z}(rA_\varphi)\right]=\\ &= \vec{z}_0(\frac{A_\varphi}{r}+\frac{\partial A_\varphi}{\partial r})-\vec{r}_0\frac{\partial A_\varphi}{\partial z}=\left(-\frac{\partial A_\varphi}{\partial z},\,0,\,\frac{A_\varphi}{r}+\frac{\partial A_\varphi}{\partial r}\right).\end{aligned} \tag{6.23}$$

Therefore by plugging (6.23) into curl $(\mathrm{curl}\,\boldsymbol{A})$ we get

$$\begin{aligned}\mathrm{curl}\,(\mathrm{curl}\,\boldsymbol{A}) &= \frac{1}{r}\begin{vmatrix}\vec{r}_0 & r\vec{\varphi}_0 & \vec{z}_0\\ \frac{\partial}{\partial r} & 0 & \frac{\partial}{\partial z}\\ -\frac{\partial A_\varphi}{\partial z} & 0 & \left(\frac{A_\varphi}{r}+\frac{\partial A_\varphi}{\partial r}\right)\end{vmatrix}\\ &= \frac{1}{r}\left[r\vec{\varphi}_0\frac{\partial}{\partial z}\left(-\frac{\partial A_\varphi}{\partial z}\right)-r\vec{\varphi}_0\frac{\partial}{\partial r}\left(\frac{A_\varphi}{r}+\frac{\partial A_\varphi}{\partial r}\right)\right]\\ &= \left(0,\,-\frac{\partial}{\partial z}\left(\frac{\partial A_\varphi}{\partial z}\right)-\frac{\partial}{\partial r}\left(\frac{\partial A_\varphi}{\partial r}\right)-\frac{\partial}{\partial r}\left(\frac{A_\varphi}{r}\right),\,0\right),\end{aligned} \tag{6.24}$$

and using (6.23) and assuming that $u_\varphi = 0$ we obtain

$$\begin{aligned} \boldsymbol{u} \times \operatorname{curl} \boldsymbol{A} &= (u_r,\, 0,\, u_z) \times \left( -\frac{\partial A_\varphi}{\partial z},\, 0,\, \frac{A_\varphi}{r} + \frac{\partial A_\varphi}{\partial r} \right) \\ &= \left( 0,\, -u_r(\frac{A_\varphi}{r} + \frac{\partial A_\varphi}{\partial r}) - u_z \frac{\partial A_\varphi}{\partial z},\, 0 \right). \end{aligned} \tag{6.25}$$

By substituting (6.24) and (6.25) into the original vector equation for the magnetic potential (6.19) we get the following scalar equation

$$\begin{aligned} &-\frac{\partial}{\partial z}\left(\frac{\partial A_\varphi}{\partial z}\right) - \frac{\partial}{\partial r}\left(\frac{\partial A_\varphi}{\partial r}\right) - \frac{\partial}{\partial r}\left(\frac{A_\varphi}{r}\right) \\ &\qquad + i\omega\mu\gamma A_\varphi + \gamma\,\mu u_r(\frac{A_\varphi}{r} + \frac{\partial A_\varphi}{\partial r}) + \gamma\,\mu u_z \frac{\partial A_\varphi}{\partial z} = \mu J_{ext,\varphi} \quad \text{in } \Omega. \end{aligned} \tag{6.26}$$

Multiplying by a test function $v_A \in H_0^1(\Omega)$, integrating over the domain $\Omega$ and using Green's theorem and boundary condition (6.9) we get the weak formulation for the magnetic potential:

Find $A_\varphi \in H_0^1(\Omega)$, such that

$$\begin{aligned} &\int_\Omega \left[ \frac{\partial A_\varphi}{\partial z}\frac{\partial v_A}{\partial z} + \frac{\partial A_\varphi}{\partial r}\frac{\partial v_A}{\partial r} + \frac{1}{r}\,A_\varphi \frac{\partial v_A}{\partial r} + i\,\omega\,\mu\,\gamma\,A_\varphi v_A \right. \\ &\left. + \gamma\,\mu\,u_r \left( \frac{A_\varphi}{r} + \frac{\partial A_\varphi}{\partial r} \right) v_A + \gamma\,\mu\,u_z \frac{\partial A_\varphi}{\partial z}\, v_A \right] \mathrm{d}(r,z) = \int_\Omega \mu\, J_{ext,\varphi}\, v_A\, \mathrm{d}(r,z), \end{aligned} \tag{6.27}$$

$$\text{for all } v_A \in H_0^1(\Omega). \tag{6.28}$$

**Remark 6.1** *The equation (6.27) is defined in the complex plane and its weak solution, phasor $\underline{A}_\varphi$ is generally also a complex-valued function. On the other hand, temperature $T$, velocities $u_r, u_z$ and pressure $p$ are real-valued functions and the numerical simulation of the whole coupled problem in the complex plane would be severe waste of the computational time. Instead, the magnetic potential is written as $\underline{A}_\varphi = A_\varphi^R + iA_\varphi^I$ and the equation (6.27) is splitted into two coupled equations for $A_\varphi^R$ and $A_\varphi^I$ in the following way. Coupling terms between real and imaginary parts of the magnetic potential are shown in red.*

$$\begin{aligned} &\int_\Omega \left[ \frac{\partial A_\varphi^R}{\partial z}\frac{\partial v_A^R}{\partial z} + \frac{\partial A_\varphi^R}{\partial r}\frac{\partial v_A^R}{\partial r} + \frac{1}{r}\,A_\varphi^R \frac{\partial v_A^R}{\partial r} - \omega\,\mu\,\gamma\,A_\varphi^I v_A^R \right. \\ &\left. + \gamma\,\mu\,u_r \left( \frac{A_\varphi^R}{r} + \frac{\partial A_\varphi^R}{\partial r} \right) v_A^R + \gamma\,\mu\,u_z \frac{\partial A_\varphi^R}{\partial z}\, v_A^R \right] d(r,z) = \int_\Omega \mu\, J_{ext,\varphi}^R\, v_A^R\, d(r,z), \end{aligned} \tag{6.29}$$

$$\int_\Omega \left[ \frac{\partial A_\varphi^I}{\partial z}\frac{\partial v_A^I}{\partial z} + \frac{\partial A_\varphi^I}{\partial r}\frac{\partial v_A^I}{\partial r} + \frac{1}{r}A_\varphi^I\frac{\partial v_A^I}{\partial r} + \omega\,\mu\,\gamma\,A_\varphi^R v_A^I \right. \tag{6.30}$$

$$\left. + \gamma\,\mu\,u_r\left(\frac{A_\varphi^I}{r} + \frac{\partial A_\varphi^I}{\partial r}\right)v_A^I + \gamma\,\mu\,u_z\frac{\partial A_\varphi^I}{\partial z}\,v_A^I \right] d(r,z) = \int_\Omega \mu\,J_{ext,\varphi}^I\,v_A^I\,d(r,z).$$

### 6.3.2 Temperature

In order to derive the weak formulation for the temperature field, let us state relations for gradient and divergence in cylindrical coordinates

$$\nabla T = \left(\frac{\partial T}{\partial r}, \frac{1}{r}\frac{\partial T}{\partial \varphi}, \frac{\partial T}{\partial z}\right), \tag{6.31}$$

$$\nabla \cdot \boldsymbol{u} = \frac{1}{r}\frac{\partial (r u_r)}{\partial r} + \frac{1}{r}\frac{\partial u_\varphi}{\partial \varphi} + \frac{\partial u_z}{\partial z}. \tag{6.32}$$

Using relations (6.31) and (6.32) and assuming that temperature $T$ is independent on $\varphi$, i.e. $\frac{\partial T}{\partial \varphi} = 0$ and $u_\varphi = 0$, equation (6.20) can be written as follows

$$\lambda\left(\frac{\partial^2 T}{\partial r^2} + \frac{1}{r}\frac{\partial T}{\partial r} + \frac{\partial^2 T}{\partial z^2}\right) = \rho\,C\left(\frac{3T - 4T^{n-1} + T^{n-2}}{2\tau} + u_r\frac{\partial T}{\partial r} + u_z\frac{\partial T}{\partial z}\right) - p_J. \tag{6.33}$$

To derive the weak formulation of (6.33) we first multiply equation (6.33) by $r$ and then by a test function $v_T \in V_T$, where

$$V_T = \{v \in H^1(\Omega), v = 0 \text{ on } \Gamma_I\},$$

and integrate over the two-dimensional domain $\Omega$.

$$-\lambda\int_\Omega \left(r\frac{\partial^2 T}{\partial r^2} + r\frac{\partial^2 T}{\partial z^2} + \frac{\partial T}{\partial r}\right) v_T\,\mathrm{d}(r,z) +$$

$$\rho C\int_\Omega \left(r\,\frac{3T - 4T^{n-1} + T^{n-2}}{2\tau} + r\,u_r\,\frac{\partial T}{\partial r} + r\,u_z\,\frac{\partial T}{\partial z}\right) v_T\,\mathrm{d}(r,z) =$$

$$\int_\Omega r\,p_J\,v_T\,\mathrm{d}(r,z).$$

Now, the Green's theorem applied on the first two terms gives

$$-\lambda\int_\Omega \left(r\frac{\partial^2 T}{\partial r^2} + r\frac{\partial^2 T}{\partial z^2}\right) v_T\,\mathrm{d}(r,z) =$$

$$= \lambda \int_\Omega \left( \frac{\partial T}{\partial r} \frac{\partial r v_T}{\partial r} + \frac{\partial T}{\partial z} \frac{\partial r v_T}{\partial z} \right) \mathrm{d}\boldsymbol{x} - \lambda \int_{\partial\Omega} \nabla T \cdot \boldsymbol{n}\, r\, v_T \,\mathrm{d}S =$$

$$= \lambda \int_\Omega \left( r \frac{\partial T}{\partial r} \frac{\partial v_T}{\partial r} + \frac{\partial T}{\partial r} v_T + r \frac{\partial T}{\partial z} \frac{\partial v_T}{\partial z} \right) \mathrm{d}(r,z) - \lambda \int_{\partial\Omega} \nabla T \cdot \boldsymbol{n}\, r\, v_T \,\mathrm{d}S.$$

Therefore we can cancel integrals $\lambda \int_\Omega \frac{\partial T}{\partial r} v_1 \mathrm{d}(r,z)$ and the weak formulations for the temperature field is as follows

$$\lambda \int_\Omega \left( r \frac{\partial T}{\partial r} \frac{\partial v_T}{\partial r} + r \frac{\partial T}{\partial z} \frac{\partial v_T}{\partial z} \right) \mathrm{d}(r,z) +$$

$$\rho C \int_\Omega \left( \frac{3T - 4T^{n-1} + T^{n-2}}{2\tau} + u_r \frac{\partial T}{\partial r} + u_z \frac{\partial T}{\partial z} \right) r\, v_T \,\mathrm{d}(r,z) -$$

$$\lambda \int_{\partial\Omega} \nabla T \cdot \boldsymbol{n}\, r\, v_T \;\mathrm{d}S = \int_\Omega r\, p_J\, v_T \,\mathrm{d}(r,z). \tag{6.34}$$

### 6.3.3 Flow field

Finally we derive the cylindrical weak formulation of the Navier-Stokes equations (6.21). We start from the weak formulation in three dimensions, where $v_i,\ i = 1,2,3$ and $q$ are test functions

$$v_i \in W = \{v \in H^1(\Omega_{R^3}), v = 0 \text{ on } \Gamma_I \cup \Gamma_W\}, \qquad\qquad q \in L^2(\Omega_{R^3}),$$

where $\Omega_{R^3}$ represents corresponding 3D computational domain and $\Gamma_I, \Gamma_O$ and $\Gamma_W$ corresponding 2D surfaces, the inlet, the outlet and the wall, respectively.

The Green's theorem implies

$$\int_{\Omega_{R^3}} \rho \left[ \frac{3u_i - 4u_i^{n-1} + u_i^{n-2}}{2\tau} v_i + (\boldsymbol{u} \cdot \nabla)\, u_i\, v_i \right] + \nu\, \nabla u_i \cdot \nabla v_i - p \frac{\partial v_i}{\partial x_i} \,\mathrm{d}\boldsymbol{x} =$$

$$\int_{\partial\Omega_{R^3}} p v_i n_i - \nu \nabla u_i \cdot \boldsymbol{n}\, v_i \,\mathrm{d}S = \int_{\Omega_{R^3}} f_i\, v_i \,\mathrm{d}\boldsymbol{x}, \qquad i = 1,2,3, \tag{6.35}$$

$$\int_{\Omega_{R^3}} \left[ \frac{\partial u_1}{\partial x} + \frac{\partial u_2}{\partial y} + \frac{\partial u_3}{\partial z} \right] q \,\mathrm{d}\boldsymbol{x} = 0. \tag{6.36}$$

The convection term in (6.35) can be rewritten using green's theorem as follows

$$\int_{\Omega_{R^3}} (\boldsymbol{u} \cdot \nabla)\, u_i\, v_i = \frac{1}{2} \int_{\Omega_{R^3}} (\boldsymbol{u} \cdot \nabla)\, u_i\, v_i + \frac{1}{2} \int_{\Omega_{R^3}} \sum_j u_j \frac{\partial u_i}{\partial x_j} v_i$$

$$= \frac{1}{2}\int_{\Omega_{R^3}} (\boldsymbol{u}\cdot\nabla)\, u_i\, v_i - \frac{1}{2}\int_{\Omega_{R^3}} \sum_j u_i \frac{\partial (u_j\, v_i)}{\partial x_j} + \frac{1}{2}\int_{\partial\Omega_{R^3}} \sum_j u_i\, v_i\, u_j\, n_j$$

$$= \frac{1}{2}\int_{\Omega_{R^3}} (\boldsymbol{u}\cdot\nabla)\, u_i\, v_i - \frac{1}{2}\int_{\Omega_{R^3}} \sum_j u_i \frac{\partial u_j}{\partial x_j}\, v_i - \frac{1}{2}\int_{\Omega_{R^3}} \sum_j u_i \frac{\partial v_i}{\partial x_j}\, u_j + \frac{1}{2}\int_{\partial\Omega_{R^3}} \sum_j u_i\, v_i\, u_j\, n_j$$

$$= \frac{1}{2}\int_{\Omega_{R^3}} (\boldsymbol{u}\cdot\nabla)\, u_i\, v_i - \frac{1}{2}\int_{\Omega_{R^3}} (\boldsymbol{u}\cdot\nabla)\, v_i\, u_i + \frac{1}{2}\int_{\partial\Omega_{R^3}} (\boldsymbol{u}\cdot\boldsymbol{n})\, u_i\, v_i$$

because

$$\operatorname{div}\boldsymbol{u} = \sum_j \frac{\partial u_j}{\partial x_j} = 0.$$

Hence, the weak formulation of the Navier-Stokes equations can be written as

$$\int_{\Omega_{R^3}} \rho\left[\frac{3}{2\tau}u_i v_i + \frac{1}{2}(\boldsymbol{u}\cdot\nabla)\, u_i\, v_i - \frac{1}{2}(\boldsymbol{u}\cdot\nabla)\, v_i\, u_i\right] + \nu\nabla u_i\cdot\nabla v_i - p\frac{\partial v_i}{\partial x_i}\mathrm{d}\boldsymbol{x} + \tag{6.37}$$

$$\frac{1}{2}\int_{\Gamma_O} (\boldsymbol{u}\cdot\boldsymbol{n})^+ u_i\, v_i\,\mathrm{d}S = \int_{\Omega_{R^3}} \frac{4{u_i}^{n-1} - {u_i}^{n-2}}{2\tau}\, v_i + f_i\, v_i \mathrm{d}\boldsymbol{x} - \int_{\Gamma_{IO}} p_{ref}\, v_i\, n_i\,\mathrm{d}S.$$

The vector velocity field $\boldsymbol{u}$ takes in cylindrical coordinates the following form

$$\begin{aligned} u_1 &= u_r\cos\varphi, \\ u_2 &= u_r\sin\varphi, \\ u_3 &= u_z. \end{aligned} \tag{6.38}$$

Under the assumption that

$$\frac{\partial u_r}{\partial\varphi} = \frac{\partial u_z}{\partial\varphi} = 0, \tag{6.39}$$

the relations for derivatives of the vector $\boldsymbol{u}$ in cylindrical coordinates can be summarized as follows

| | $\frac{\partial u_1}{\partial x_i}$ | $\frac{\partial u_2}{\partial x_i}$ | $\frac{\partial u_3}{\partial x_i}$ |
|---|---|---|---|
| $\partial x_1$ | $\frac{\partial u_r}{\partial r}\cos^2\varphi + \frac{1}{r}u_r\sin^2\varphi$ | $\frac{\partial u_r}{\partial r}\cos\varphi\sin\varphi - \frac{1}{r}u_r\cos\varphi\sin\varphi$ | $\frac{\partial u_z}{\partial r}\cos\varphi$ |
| $\partial x_2$ | $\frac{\partial u_r}{\partial r}\cos\varphi\sin\varphi - \frac{1}{r}u_r\cos\varphi\sin\varphi$ | $\frac{\partial u_r}{\partial r}\sin^2\varphi + \frac{1}{r}u_r\cos^2\varphi$ | $\frac{\partial u_z}{\partial r}\sin\varphi$ |
| $\partial x_3$ | $\frac{\partial u_r}{\partial z}\cos\varphi$ | $\frac{\partial u_r}{\partial z}\sin\varphi$ | $\frac{\partial u_z}{\partial z}$ |

Thus the formula for $\nabla u_i \cdot \nabla v_i$ in (6.35) can be derived as follows

$$\nabla u_1 \cdot \nabla v_1 = \frac{\partial u_1}{\partial x_1}\frac{\partial v_1}{\partial x_1} + \frac{\partial u_1}{\partial x_2}\frac{\partial v_1}{\partial x_2} + \frac{\partial u_1}{\partial x_3}\frac{\partial v_1}{\partial x_3} =$$

$$\frac{\partial u_r}{\partial r}\frac{\partial v_r}{\partial r}\cos^4\varphi + \frac{1}{r^2}u_r v_r \sin^4\varphi + \frac{1}{r}u_r\frac{\partial v_r}{\partial r}\cos^2\varphi\sin^2\varphi + \frac{1}{r}\frac{\partial u_r}{\partial r}v_r\cos^2\varphi\sin^2\varphi +$$
$$\frac{\partial u_r}{\partial r}\frac{\partial v_r}{\partial r}\sin^2\varphi\cos^2\varphi + \frac{1}{r^2}u_r v_r\cos^2\varphi\sin^2\varphi - \frac{1}{r}u_r\frac{\partial v_r}{\partial r}\cos^2\varphi\sin^2\varphi - \frac{1}{r}\frac{\partial u_r}{\partial r}v_r\cos^2\varphi\sin^2\varphi$$
$$+\frac{\partial u_r}{\partial z}\frac{\partial v_r}{\partial z}\cos^2\varphi.$$

Similarly, the formula for $\nabla u_i \cdot \nabla v_i,\ i = 2, 3$ can be derived, thus

$$\begin{aligned}
\nabla u_1 \cdot \nabla v_1 &= \frac{\partial u_r}{\partial r}\frac{\partial v_r}{\partial r}\cos^2\varphi + \frac{1}{r^2}u_r v_r\sin^2\varphi + \frac{\partial u_r}{\partial z}\frac{\partial v_r}{\partial z}\cos^2\varphi,\\
\nabla u_2 \cdot \nabla v_2 &= \frac{\partial u_r}{\partial r}\frac{\partial v_r}{\partial r}\sin^2\varphi + \frac{1}{r^2}u_r v_r\cos^2\varphi + \frac{\partial u_r}{\partial z}\frac{\partial v_r}{\partial z}\sin^2\varphi,\\
\nabla u_3 \cdot \nabla v_3 &= \frac{\partial u_z}{\partial r}\frac{\partial v_z}{\partial r} + \frac{\partial u_z}{\partial z}\frac{\partial v_z}{\partial z}.
\end{aligned}$$

Now let us derive convective terms

$$(\boldsymbol{u}\cdot\nabla)u_1 = u_1\frac{\partial u_1}{\partial x_1} + u_2\frac{\partial u_1}{\partial x_2} + u_3\frac{\partial u_1}{\partial x_3} = u_z\frac{\partial u_r}{\partial z}\cos\varphi +$$
$$u_r\cos\varphi\left(\frac{\partial u_r}{\partial r}\cos^2\varphi + \frac{1}{r}u_r\sin^2\varphi\right) + u_r\sin\varphi\left(\frac{\partial u_r}{\partial r}\cos\varphi\sin\varphi - \frac{1}{r}u_r\cos\varphi\sin\varphi\right)$$
$$= u_r\frac{\partial u_r}{\partial r}\cos\varphi + u_z\frac{\partial u_r}{\partial z}\cos\varphi.$$

Similarly, the other convective terms can be obtained, thus

$$\begin{aligned}
(\boldsymbol{u}\cdot\nabla)u_1 &= u_r\frac{\partial u_r}{\partial r}\cos\varphi + u_z\frac{\partial u_r}{\partial z}\cos\varphi,\\
(\boldsymbol{u}\cdot\nabla)u_2 &= u_r\frac{\partial u_r}{\partial r}\sin\varphi + u_z\frac{\partial u_r}{\partial z}\sin\varphi,\\
(\boldsymbol{u}\cdot\nabla)u_3 &= u_r\frac{\partial u_z}{\partial r} + u_z\frac{\partial u_z}{\partial z}.
\end{aligned}$$

Finally the formula (6.32) for the divergence $\nabla\cdot\boldsymbol{u}$ becomes (under the assumption (6.39))

$$\nabla\cdot\boldsymbol{u} = \frac{\partial u_r}{\partial r} + \frac{1}{r}u_r + \frac{\partial u_z}{\partial z}.$$

Now let us substitute all these derived formulas into (6.35) using substitution theorem.

First equation in (6.35) in cylindrical coordinates reads

$$\iiint \left[\rho\frac{3u_r}{2\tau}v_r\cos^2\varphi+\nu\left(\frac{\partial u_r}{\partial r}\frac{\partial v_r}{\partial r}\cos^2\varphi+\frac{1}{r^2}u_rv_r\sin^2\varphi+\frac{\partial u_r}{\partial z}\frac{\partial v_r}{\partial z}\cos^2\varphi\right)\right.$$
$$\left.+\rho\left(u_r\frac{\partial u_r}{\partial r}v_r\cos^2\varphi+u_z\frac{\partial u_r}{\partial z}v_r\cos^2\varphi\right)-p\left(\frac{\partial v_r}{\partial r}\cos^2\varphi+\frac{1}{r}v_r\sin^2\varphi\right)\right]\,r\,\mathrm{d}(r,\varphi,z)$$
$$=\iiint\left(\frac{4u_r^{n-1}-u_r^{n-2}}{2\tau}v_r\cos^2\varphi+f_rv_r\cos^2\varphi\right)\,r\,\mathrm{d}(r,\varphi,z).$$

Similarly second N-S equation

$$\iiint \left[\rho\frac{3u_r}{2\tau}v_r\sin^2\varphi+\nu\left(\frac{\partial u_r}{\partial r}\frac{\partial v_r}{\partial r}\sin^2\varphi+\frac{1}{r^2}u_rv_r\cos^2\varphi+\frac{\partial u_r}{\partial z}\frac{\partial v_r}{\partial z}\sin^2\varphi\right)\right.$$
$$\left.+\rho\left(u_r\frac{\partial u_r}{\partial r}v_r\sin^2\varphi+u_z\frac{\partial u_r}{\partial z}v_r\sin^2\varphi\right)-p\left(\frac{\partial v_r}{\partial r}\sin^2\varphi+\frac{1}{r}v_r\cos^2\varphi\right)\right]\,r\,\mathrm{d}(r,\varphi,z)$$
$$=\iiint\left(\frac{4u_r^{n-1}-u_r^{n-2}}{2\tau}v_r\sin^2\varphi+f_rv_r\sin^2\varphi\right)\,r\,\mathrm{d}(r,\varphi,z).$$

By adding these two equations together we get the equation for $u_r$. Since the integrands no longer depend on $\varphi$, we can divide the equation by $2\pi$ and simplify as follows

$$\iint\rho\left(\frac{3u_r}{2\tau}v_r+u_r\frac{\partial u_r}{\partial r}v_r+u_z\frac{\partial u_r}{\partial z}v_r\right)+\nu\left(\frac{\partial u_r}{\partial r}\frac{\partial v_r}{\partial r}+\frac{1}{r^2}u_rv_r+\frac{\partial u_r}{\partial z}\frac{\partial v_r}{\partial z}\right)$$
$$-p\left(\frac{\partial v_r}{\partial r}+\frac{1}{r}v_r\right)\,r\,\mathrm{d}(r,z)=\iint\left(\frac{4u_r^{n-1}-u_r^{n-2}}{2\tau}v_r+f_rv_r\right)\,r\,\mathrm{d}(r,z).\qquad(6.40)$$

Finally, the third N-S equation, in other words the equation for $u_z$

$$\iint\rho\left(\frac{3u_z}{2\tau}v_z+u_r\frac{\partial u_z}{\partial r}v_z+u_z\frac{\partial u_z}{\partial z}v_z\right)+\nu\left(\frac{\partial u_z}{\partial r}\frac{\partial v_z}{\partial r}+\frac{\partial u_z}{\partial z}\frac{\partial v_z}{\partial z}\right)$$
$$-p\frac{\partial v_z}{\partial z}\,r\,\mathrm{d}(r,z)=\iint\left(\frac{4u_z^{n-1}-u_z^{n-2}}{2\tau}v_z+f_zv_z\right)\,r\,\mathrm{d}(r,z).\qquad(6.41)$$

The continuity equation in the weak form reads

$$\iint(\frac{\partial u_r}{\partial r}+\frac{1}{r}u_r+\frac{\partial u_z}{\partial z})\,q\,r\,\mathrm{d}(r,z)=0.\qquad(6.42)$$

The formula for the internal Lorentz force in (6.40) and (6.41) written in the axisym-

metric arrangement is as follows

$$\boldsymbol{f} = \boldsymbol{J} \times \boldsymbol{B} = (0, J_\varphi, 0) \times (B_r, 0, B_z) = (J_\varphi B_z, 0, -J_\varphi B_r).$$

Thus

$$\begin{aligned} f_r &= |J_\varphi|\,|B_z| \cos(\arg J_\varphi + \arg B_z), \\ f_z &= -|J_\varphi|\,|B_r| \cos(\arg J_\varphi + \arg B_r). \end{aligned}$$

## 6.4 Numerical method

The numerical solution at each time level is defined in means of finite element method. The computational domain is first covered by a coarse partition consisting of generally unstructured quadrilaterals. Since the particular physical fields are defined in different parts of the domain, the meshes $\mathcal{T}_a$ (for $A_\varphi^R, A_\varphi^I$), $\mathcal{T}_t$ (for $T$) and $\mathcal{T}_f$ (for $u_r$, $u_z$, $p$) can cover only parts of the whole domain to better accommodate individual needs of the solution components. Therefore the flow field is sought only in the interior of the pipe, temperature field also in the pipe itself and the magnetic field has to be solved also in the inductor and in the vicinity of the arrangement. Computational domains with initial coarse meshes for all three physics are depicted in Fig. 6.2.

Standard continuous higher-order finite elements are used for the discretization of the magnetic potential $A_\varphi^R$, $A_\varphi^I$ and temperature $T$. Thus

$$A_\varphi^R,\, A_\varphi^I \in V_{hp}^A = \left\{ v \in H_0^1(\Omega) \cap C(\overline{\Omega});\; v|_K \in P^{p_K}(K),\; \forall K \in \mathcal{T}_a \right\},$$

$$T \in V_{hp}^T = \left\{ v \in V_T \cap C(\overline{\Omega});\; v|_K \in P^{p_K}(K),\; \forall K \in \mathcal{T}_t \right\}.$$

For the flow field, well-known Taylor-Hood $Q_{k+1}/Q_k$ elements satisfying Babuška-Brezzi condition are used. Here velocities $u_r$, $u_z$ are approximated by continuous elements, while pressure $p$ is discretized using discontinuous $L_2$ finite elements.

$$\boldsymbol{u} \in \boldsymbol{V}_{hp}^F = \left\{ \boldsymbol{v} \in V_F \cap (C(\overline{\Omega}))^2;\; \boldsymbol{v}|_K \in (P^{k+1}(K))^2,\; \forall K \in \mathcal{T}_f \right\},$$

$$V_F = \{ \boldsymbol{v} \in (H^1(\Omega))^2,\, \boldsymbol{v} = 0 \text{ on } \Gamma_I \cup \Gamma_W \},$$

$$p \in Q_{hp} = \left\{ q \in L^2(\Omega) \cap C(\overline{\Omega});\; q|_K \in P^k(K),\; \forall K \in \mathcal{T}_f \right\}.$$

**Remark 6.2** *We have tested Navier-Stokes equations derived in axisymmetric scheme on a flow through a narrow pipe. Since the quadrature rules [48] used in our computations do not have points on the boundary of elements, we did not experienced any numerical instabilities in our computations.*

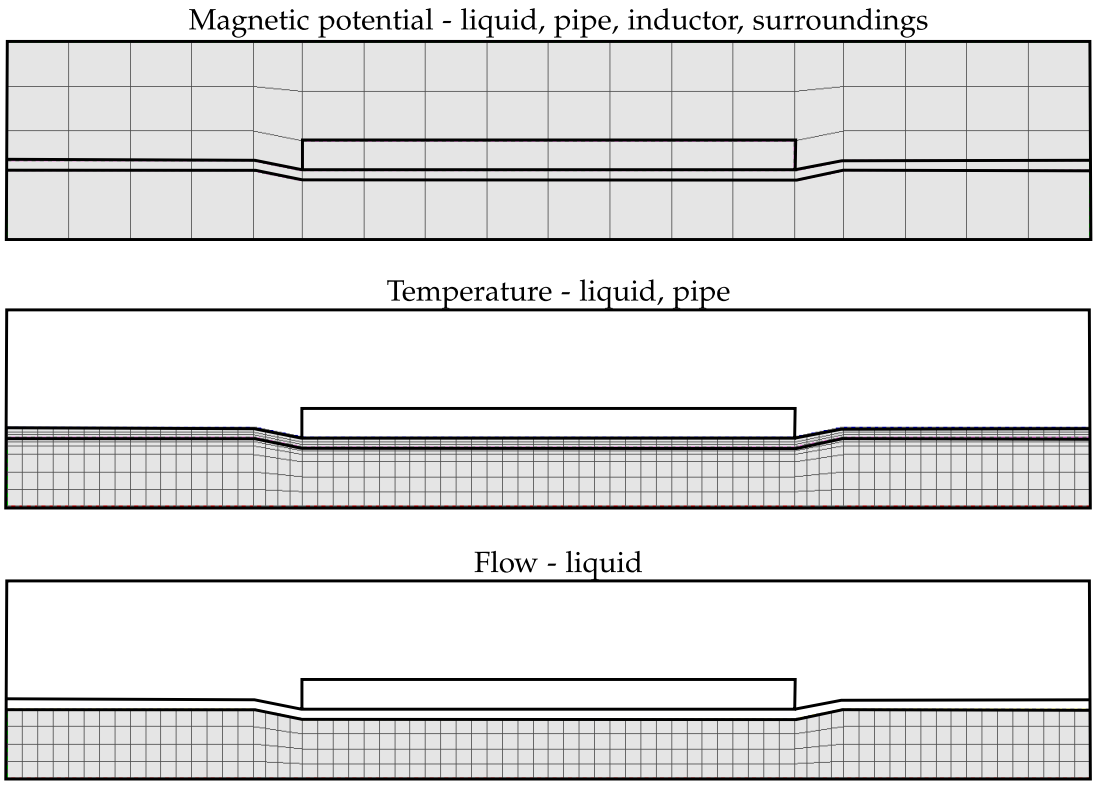



Figure 6.2: Initial meshes $\mathcal{T}_a$ (for $A_\varphi^R, A_\varphi^I$), $\mathcal{T}_t$ (for $T$) and $\mathcal{T}_f$ (for $u_r$, $u_z$, $p$).

Since the Reynolds number for our computation is large (up to $10^5$), the numerical solution can contain nonphysical spurious oscillations. In order to suppress them the finite element method is stabilized using the streamline-diffusion and grad–div techniques.

### 6.4.1 Stabilization of FEM

We define stabilization terms

$$\begin{aligned}
\mathcal{L}_h &= \int \delta_K \left( \rho \frac{1}{\tau} \boldsymbol{u} - \nu \Delta \boldsymbol{u} + (\boldsymbol{u}^{n-1} \cdot \nabla)\, \boldsymbol{u} + \nabla p \right) \cdot (\boldsymbol{u}^{n-1} \cdot \nabla)\, \boldsymbol{v} \, \mathrm{d}\boldsymbol{x}, \\
\mathcal{P}_h &= \int \tau_K \nabla \cdot \boldsymbol{u} \, \nabla \cdot \boldsymbol{v} \, \mathrm{d}\boldsymbol{x}, \\
\mathcal{F}_h &= \int \delta_K \left( \rho \frac{1}{\tau} \boldsymbol{u}^{n-1} + \boldsymbol{f} \right) (\boldsymbol{u}^{n-1} \cdot \nabla)\, \boldsymbol{v} \, \mathrm{d}\boldsymbol{x}.
\end{aligned}$$

In a similar way as equations (6.40), (6.41) these equations can be written in cylindrical coordinates as follows

$$\int \delta_K \left( \rho \frac{1}{\tau} u_r \cos\varphi - \nu \left( \frac{\partial^2 u_r}{\partial r^2} \cos\varphi + \frac{1}{r} \frac{\partial u_r}{\partial r} \cos\varphi - \frac{1}{r^2} u_r \cos\varphi + \frac{\partial^2 u_r}{\partial z^2} \cos\varphi \right) \right.$$

$$\left. + u_r^{n-1} \frac{\partial u_r}{\partial r} \cos\varphi + u_z^{n-1} \frac{\partial u_r}{\partial z} \cos\varphi + \frac{\partial p}{\partial r} \cos\varphi \right) \left( u_r^{n-1} \frac{\partial v_r}{\partial r} \cos\varphi + u_z^{n-1} \frac{\partial v_r}{\partial z} \cos\varphi \right)$$

$$+ \int \delta_K \left( \rho \frac{1}{\tau} u_r \sin\varphi - \nu \left( \frac{\partial^2 u_r}{\partial r^2} \sin\varphi + \frac{1}{r} \frac{\partial u_r}{\partial r} \sin\varphi - \frac{1}{r^2} u_r \sin\varphi + \frac{\partial^2 u_r}{\partial z^2} \sin\varphi \right) \right.$$

$$\left. + u_r^{n-1} \frac{\partial u_r}{\partial r} \sin\varphi + u_z^{n-1} \frac{\partial u_r}{\partial z} \sin\varphi + \frac{\partial p}{\partial r} \sin\varphi \right) \left( u_r^{n-1} \frac{\partial v_r}{\partial r} \sin\varphi + u_z^{n-1} \frac{\partial v_r}{\partial z} \sin\varphi \right)$$

$$= \int \delta_K \left( \rho \frac{1}{\tau} u_r - \nu \left( \frac{\partial^2 u_r}{\partial r^2} + \frac{1}{r} \frac{\partial u_r}{\partial r} - \frac{1}{r^2} u_r + \frac{\partial^2 u_r}{\partial z^2} \right) \right.$$

$$\left. + u_r^{n-1} \frac{\partial u_r}{\partial r} + u_z^{n-1} \frac{\partial u_r}{\partial z} + \frac{\partial p}{\partial r} \right) \left( u_r^{n-1} \frac{\partial v_r}{\partial r} + u_z^{n-1} \frac{\partial v_r}{\partial z} \right) (\cos^2\varphi + \sin^2\varphi) r \, \mathrm{d}(r,z) =$$

$$= \int \delta_K \left( \rho \frac{1}{\tau} u_r - \nu \left( \frac{\partial^2 u_r}{\partial r^2} + \frac{1}{r} \frac{\partial u_r}{\partial r} - \frac{1}{r^2} u_r + \frac{\partial^2 u_r}{\partial z^2} \right) \right.$$

$$\left. + u_r^{n-1} \frac{\partial u_r}{\partial r} + u_z^{n-1} \frac{\partial u_r}{\partial z} + \frac{\partial p}{\partial r} \right) \left( u_r^{n-1} \frac{\partial v_r}{\partial r} + u_z^{n-1} \frac{\partial v_r}{\partial z} \right) r \, \mathrm{d}(r,z).$$

The third equation in $\mathcal{L}_h$ is

$$\int \delta_K \left( \rho \frac{1}{\tau} u_z - \nu \left( \frac{\partial^2 u_z}{\partial r^2} + \frac{1}{r} \frac{\partial u_z}{\partial r} + \frac{\partial^2 u_z}{\partial z^2} \right) \right.$$

$$\left. + u_r^{n-1} \frac{\partial u_z}{\partial r} + u_z^{n-1} \frac{\partial u_z}{\partial z} + \frac{\partial p}{\partial z} \right) \left( u_r^{n-1} \frac{\partial v_z}{\partial r} + u_z^{n-1} \frac{\partial v_z}{\partial z} \right) r \mathrm{d}(r,z).$$

The second stabilization term reads

$$\mathcal{P}_h = \int \tau_K \left( \frac{\partial u_r}{\partial r} + \frac{1}{r} u_r + \frac{\partial u_z}{\partial z} \right) \left( \frac{\partial v_r}{\partial r} + \frac{1}{r} v_r + \frac{\partial v_z}{\partial z} \right) r \, \mathrm{d}(r,z).$$

Finally, the right hand side stabilization term

$$\mathcal{F}_h = \int \delta_K \left( \rho \frac{1}{\tau} u_r^{n-1} + f_r \right) \left( u_r^{n-1} \frac{\partial v_r}{\partial r} + u_z^{n-1} \frac{\partial v_r}{\partial z} \right) +$$

$$\left( \rho \frac{1}{\tau} u_z^{n-1} + f_z \right) \left( u_r^{n-1} \frac{\partial v_z}{\partial r} + u_z^{n-1} \frac{\partial v_z}{\partial z} \right) r \, \mathrm{d}(r,z).$$

### 6.4.2 Adaptivity and individual meshes

As was described in Chapters 3, 4 and 5 the *hp*-adaptive process is run on each time level for all three physics individually, i.e. the magnetic potential, temperature and flow field have their own mesh automatically adapted to capture their individual behavior. For the magnetic potential and temperature *hp*-adaptivity was used, for the flow field *h*-adaptive algorithm was applied. Thus on the time level $t_n$ we obtain three independent meshes $\mathcal{T}_a^n$, $\mathcal{T}_t^n$, $\mathcal{T}_f^n$ and to evaluate integrals in the weak formulation the multimesh technique described in Sec. 4.4 is used. However, in contrast to the standard multimesh, here, the meshes not only differ in their refinements but also cover different parts of the computational domain. Hence, the standard element-by-element assembly procedure needs little modification. First, we assume that all meshes come from one original master mesh and that in some meshes some elements are "missing". In our particular problem, the coarse master mesh covers the computational domain for the magnetic field and in meshes for the temperature and flow fields some elements are "deleted". The element-by-element procedure assembling the stiffness matrix then skips integrals involving functions from spaces connected with meshes with missing element (see Alg. 6).

**Algorithm 6**: Element-by-element assembling with missing elements.

**forall** *elements* $Q_k$ *of union mesh* $\mathcal{T}_u$ **do**
  **forall** *equations:* $m = 1 \dots 6$ **do**
    **if** $Q_k \subset \mathcal{T}_m$ **then**
      $L_m =$ list of basis functions whose $\text{supp}(v_i) \cap Q_k \neq \emptyset$;
    **else** // missing element
      $L_m = \emptyset$;
  **for** $m = 1 \dots 6$ **do**
    **for** $n = 1 \dots 6$ **do**
      **for** $i \in L_m$ **do**
        **for** $j \in L_n$ **do**
          $S_{ij} = S_{ij} + a_{mn}(v_i, v_j)|_{Q_k}$;

Our approach results in smaller linear systems since no degrees of freedom are wasted in areas where some components of the solution are not defined. Note that even though the meshes are mutually independent, the coupled problem is still solved monolithically, thus all components are advanced in time simultaneously and nonlinearities can be resolved more accurately.

**Remark 6.3** *Since the nonlinearities in our example are very weak and the solution changes very slowly we treat the nonlinearities by fixed-point iterations which turned out to be sufficient for this particular problem. But let us point out that Newton's method could be used instead without any difficulty.*

## 6.5 Numerical experiment

### 6.5.1 Input data

The pipe in Fig. 6.1 carries molten sodium whose inlet temperature is $150°C$. At the time $t_0$ (when the transport is considered steady-state) the inductor is switched on in order to warm up the melt. The distributions of the velocity in the melt and the temperature in the whole system melt-pipe for time $t_0$ are known (they follow from the solution of the steady-state process before the inductor was switched on). The average value of the velocity of the melt at the inlet is 0.2 $\mathrm{ms}^{-1}$. The inductor parameters are $\underline{J}_{\mathrm{ext}} = 10^6$ $\mathrm{Am}^{-2}$, and $f = 50$ Hz. The physical parameters of the molten sodium and basalt are given in Tabs. 6.1 and 6.2.

Table 6.1: Physical parameters of molten sodium.

| **quantity** | **unit** | **value** |
|---|---|---|
| specific mass | $\mathrm{kg\ m}^{-3}$ | 927 |
| temperature of melting | °C | 97.72 |
| electrical conductivity | $\mathrm{S\ m}^{-1}$ | $2.0964 \times 10^6$ |
| thermal conductivity | $\mathrm{W\ m}^{-1}\ \mathrm{K}^{-1}$ | 142 |
| thermal capacity | $\mathrm{J\ kg}^{-1}\ \mathrm{K}^{-1}$ | 1230 |
| dynamic viscosity | Pa s | for 98°C 0.000688<br>for 127°C 0.000599<br>for 227°C 0.000415 |

Table 6.2: Physical parameters of basalt.

| **quantity** | **unit** | **value** |
|---|---|---|
| specific mass | $\mathrm{kg\ m}^{-3}$ | 2900 |
| electrical conductivity | $\mathrm{S\ m}^{-1}$ | 0 |
| thermal conductivity | $\mathrm{W\ m}^{-1}\ \mathrm{K}^{-1}$ | 2.0 |
| thermal capacity | $\mathrm{J\ kg}^{-1}\ \mathrm{K}^{-1}$ | 840 |

### 6.5.2 Results

First, we checked the convergence of the results related to the distribution of the magnetic field with respect to the position of its artificial boundary. It was proved

that particularly the position of its external part (see Fig. 6.2) does not need to be too far from the inductor. Although the magnetic field near this part of the boundary is strongly distorted (see Fig. 6.3), its values in the melt do not change any longer.

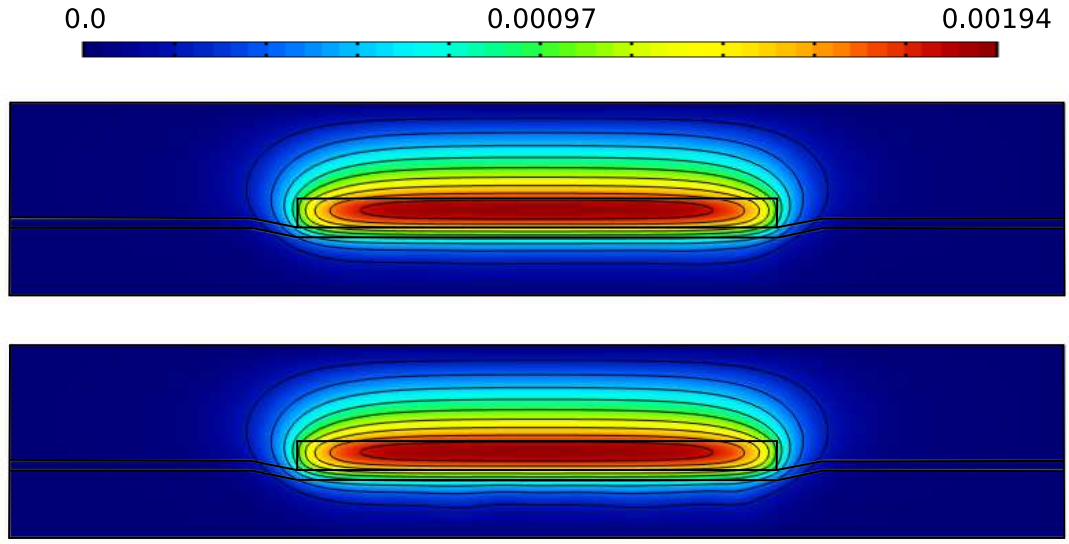



Figure 6.3: Distribution of the magnetic vector potential $A_\varphi$ (Wb m$^{-1}$) at two time instants ($t = 2.0$, 4.0s).

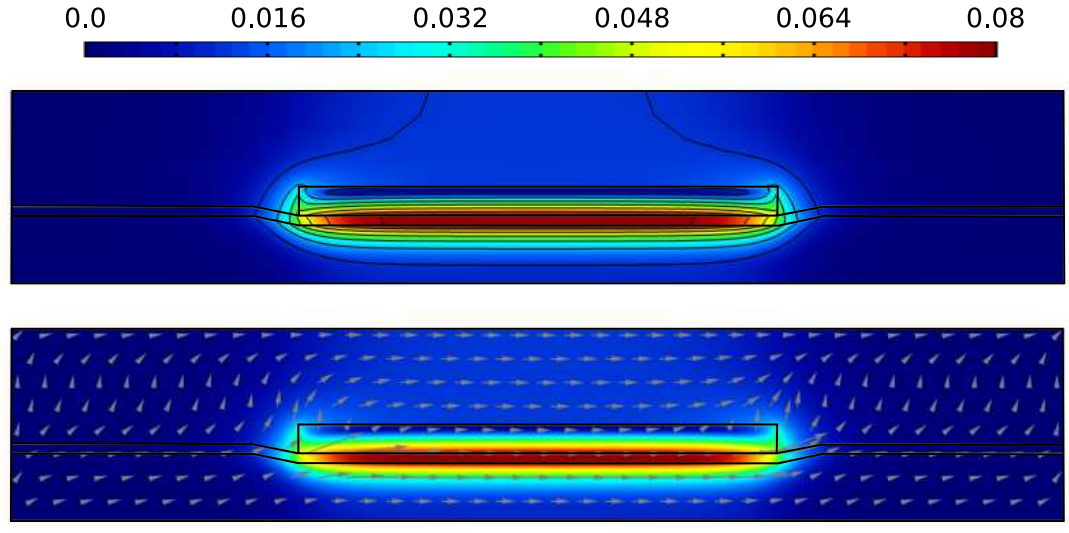



Figure 6.4: Distribution of the module of magnetic flux density $\boldsymbol{B}$ (T) with isolines (top). Magnetic flux density as a vector field (bottom).

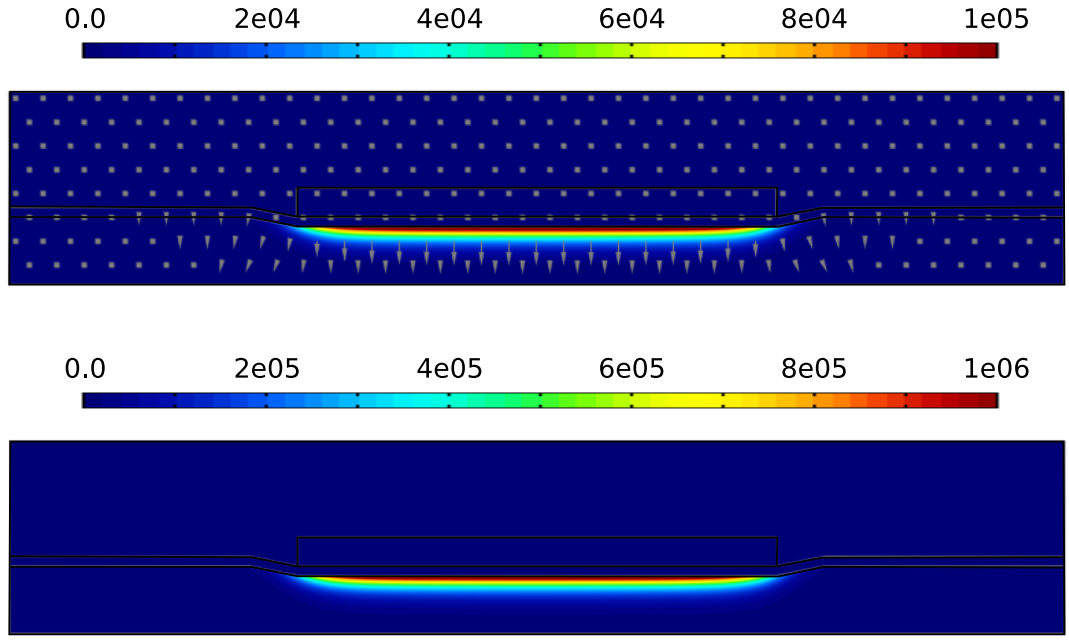



Figure 6.5: Lorentz forces acting on the fluid (top). Joule's looses heating up the fluid (bottom).

Fig. 6.3 shows the distribution of the magnetic field $\underline{A}_\varphi$ in the system at two time levels $t = 2.0, 4.0$s. For low velocities of the melt its values are practically the same as if we would omit in (6.8) the velocity term, while with increasing velocity we can observe perturbances in the magnetic field caused by the velocity term in (6.8). Distribution of the magnetic flux density $\boldsymbol{B}$ in the domain is depicted in Fig. 6.4. Even in this case, distribution practically does not depend on the velocity term in (6.8). Joule's losses causing heating of the fluid and Lorentz forces accelerating the fluid are shown in Fig. 6.5. The temperature field does not change too much with time since the external current $\underline{J}_{\text{ext}}$ is small and thus resulting heating effect is small. Its distribution starts from the steady state before switching on the inductor. At time $t_0$ this distribution is practically uniform both in the melt and pipe and its value lies at maximum several hundredths of a degree below 150°C. Fig. 6.6 shows that even 6s after switching the inductor on its distribution does not change by more than about 0.2°C due to the thermal inertia of the system. From Lorentz forces in Fig. 6.5 we see that the conducting fluid is repelled away from the inductor which results in formation of vortices in the velocity field at the end of the narrow part of the pipe. Distribution of the magnitude of the velocity at several time instants is depicted in Fig. 6.7, and corresponding pressure distributions in Fig. 6.8. The underlying meshes for the potential, temperature and flow fields are shown in Figs. 6.10, 6.11 and 6.12, respectively. Notice mutually independent dynamical meshes for all physical fields. Magnetic field and

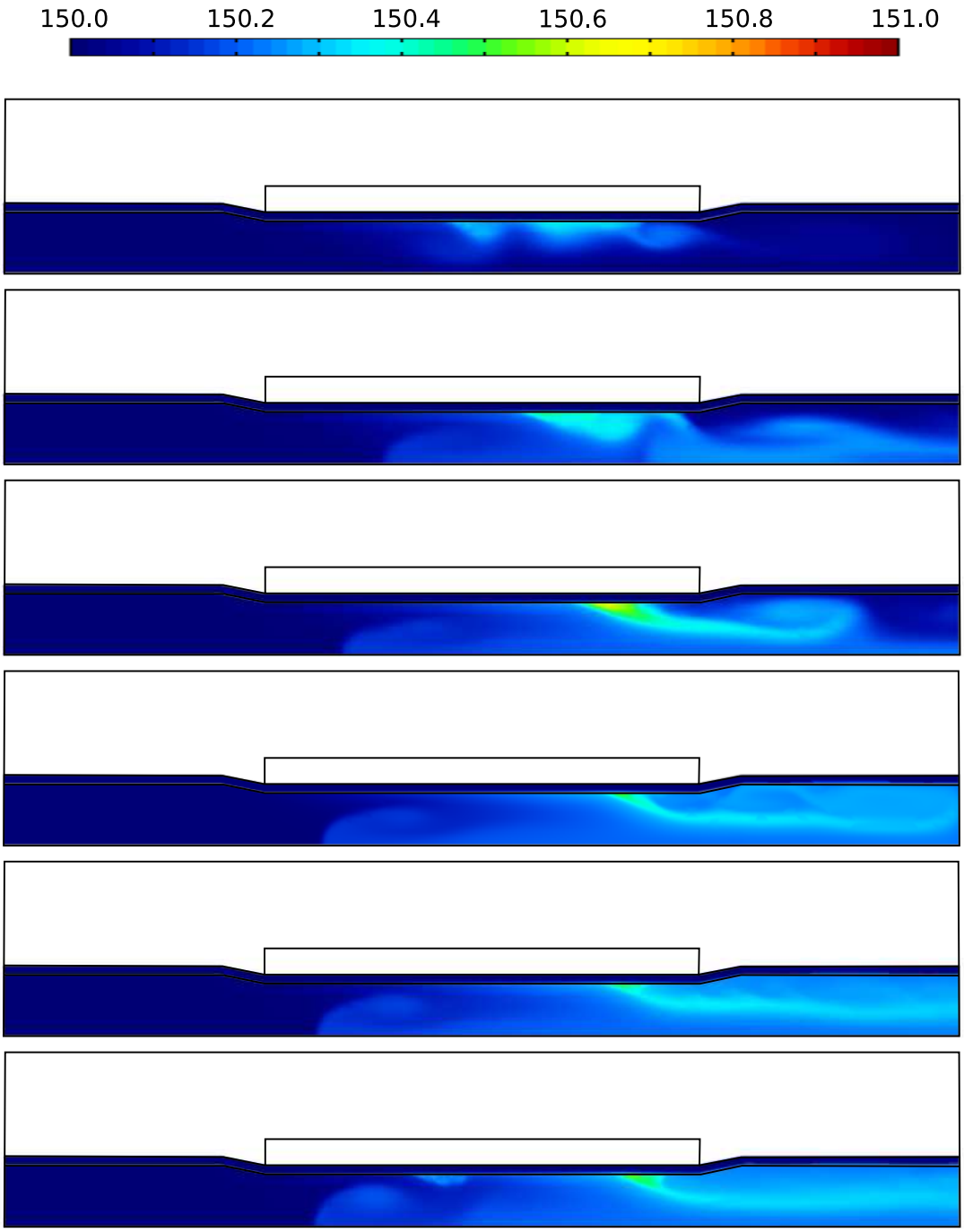



Figure 6.6: Temperature field $T$ (°C) at $t = 0.5$, 1.0, 1.5, 2.0, 2.5, 3.0s.

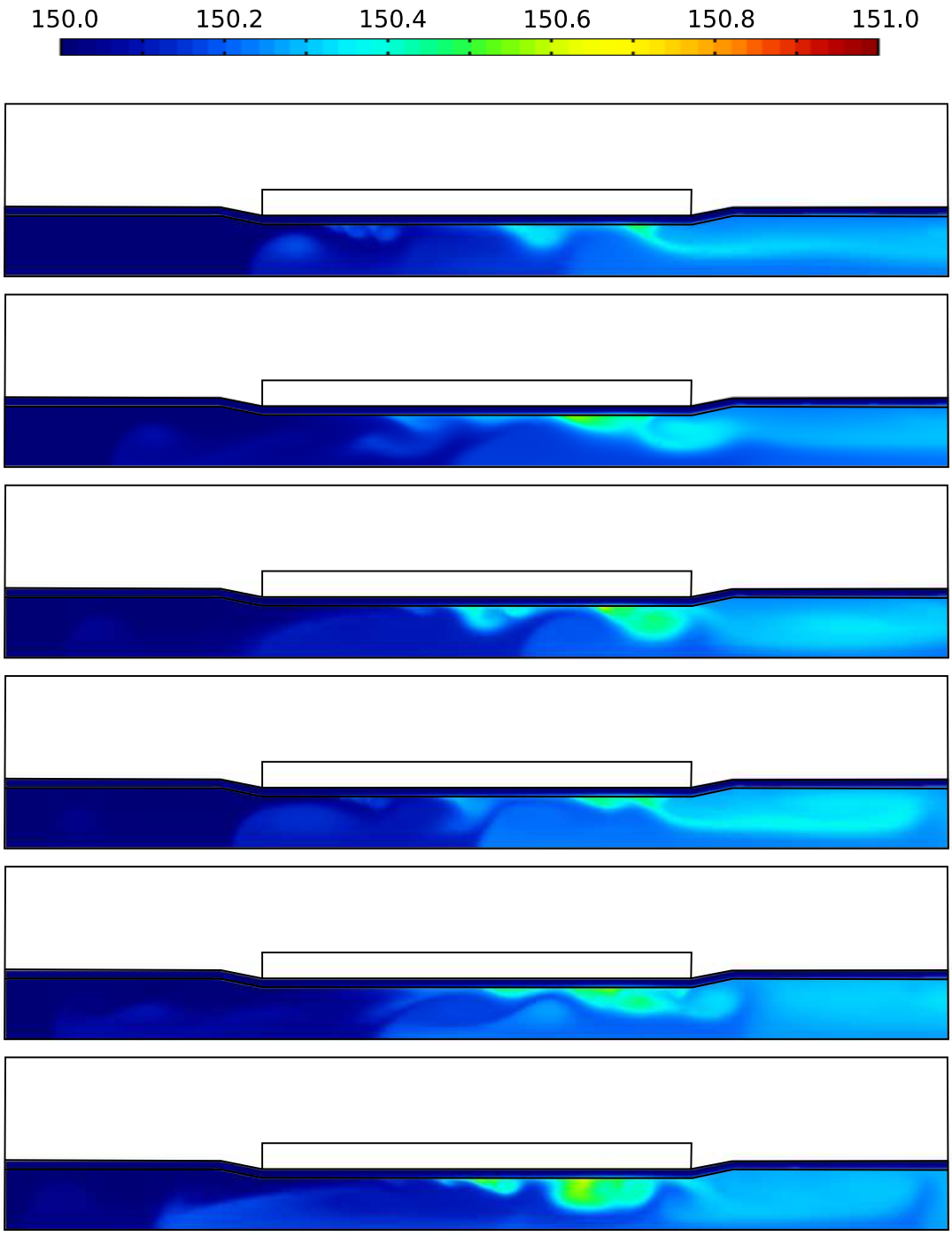



Figure 6.6: *(continued)* Temperature field $T$ (°C) at $t = 3.5$, 4.0, 4.5, 5.0, 5.5, 6.0s.

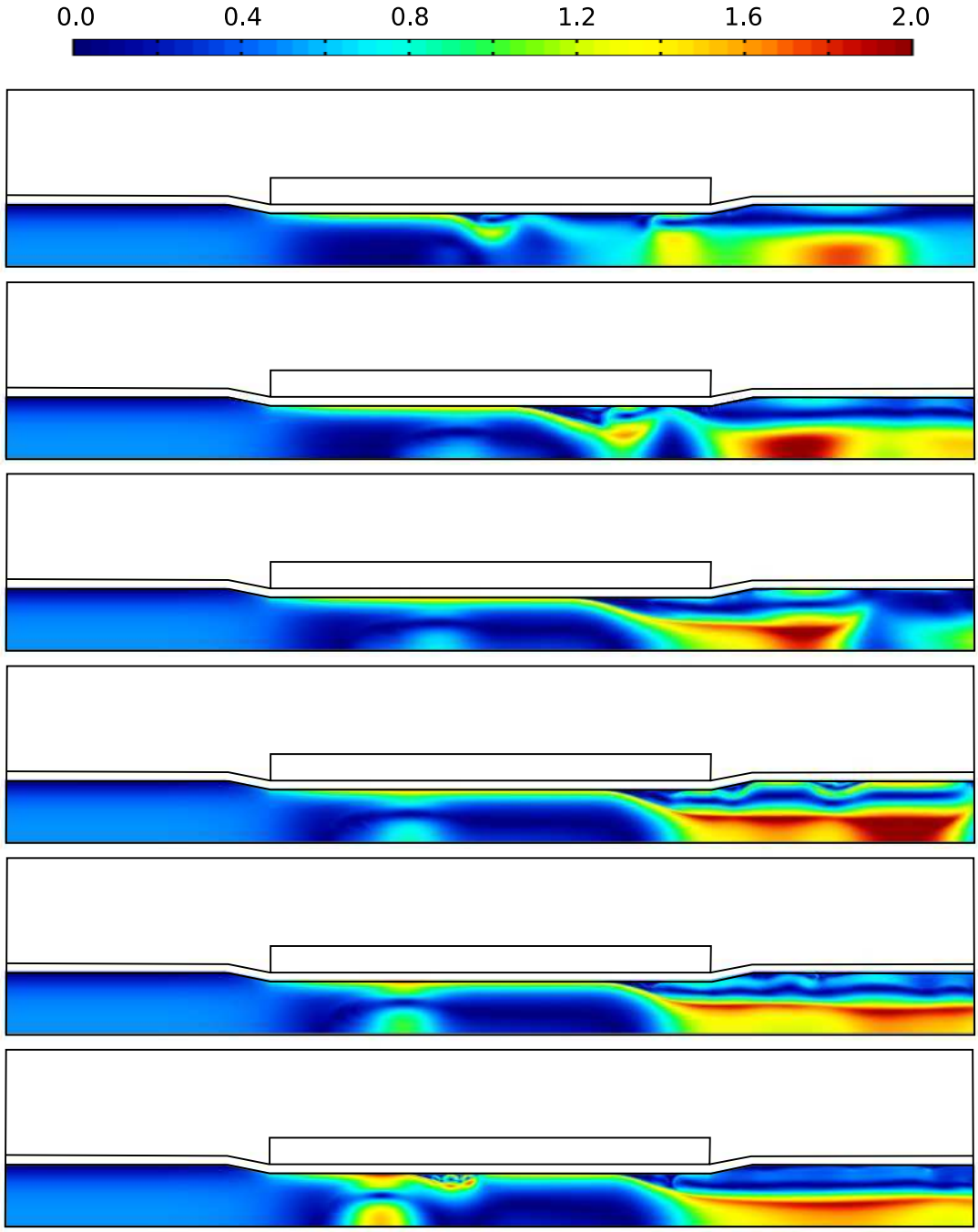



Figure 6.7: Magnitude of velocity $\boldsymbol{u}$ (ms$^{-1}$) at $t = 0.5, 1.0, 1.5, 2.0, 2.5, 3.0$s.

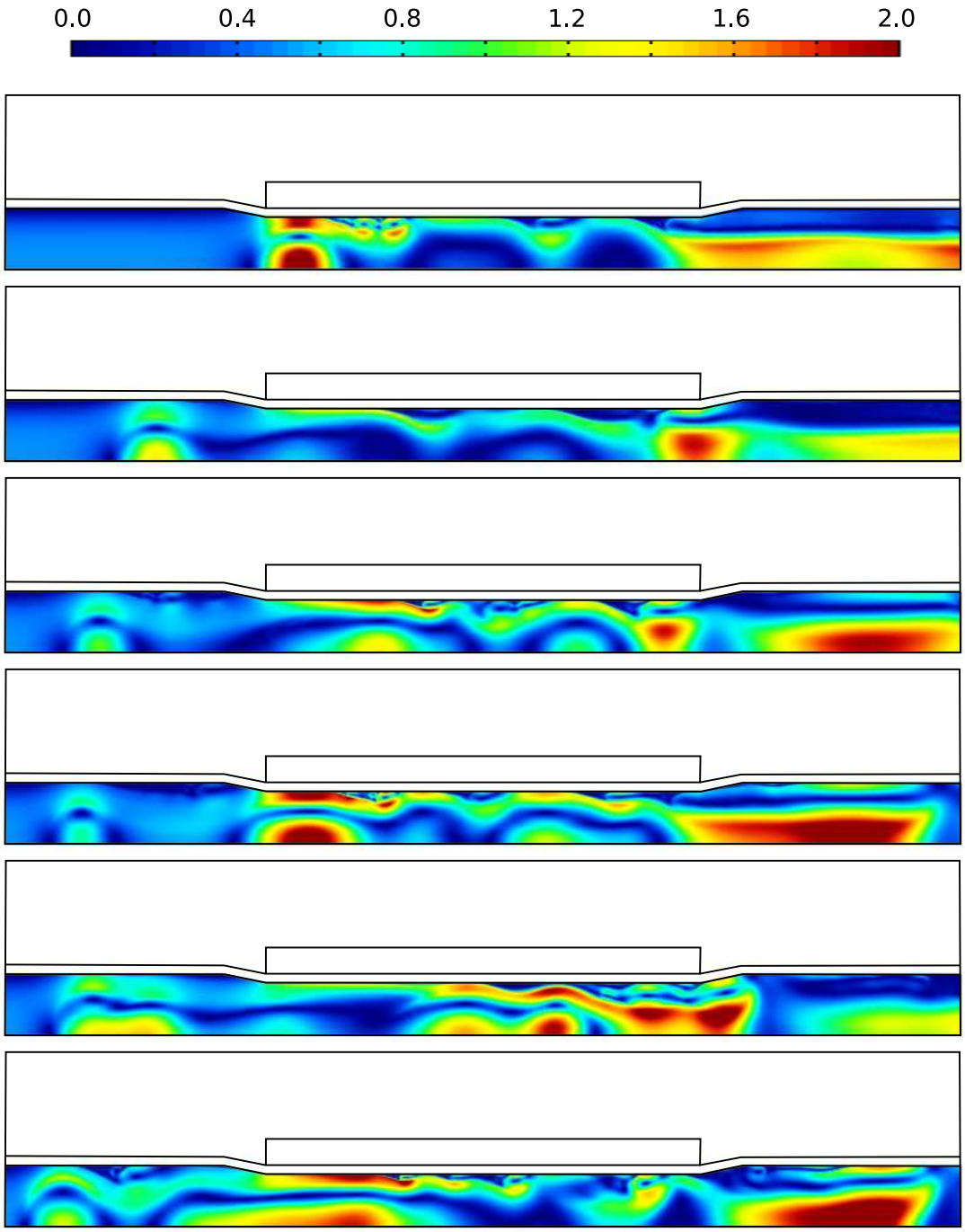



Figure 6.7: *(continued)* Magnitude of velocity $\boldsymbol{u}$ (ms$^{-1}$) at $t = 3.5$, 4.0, 4.5, 5.0, 5.5, 6.0s.

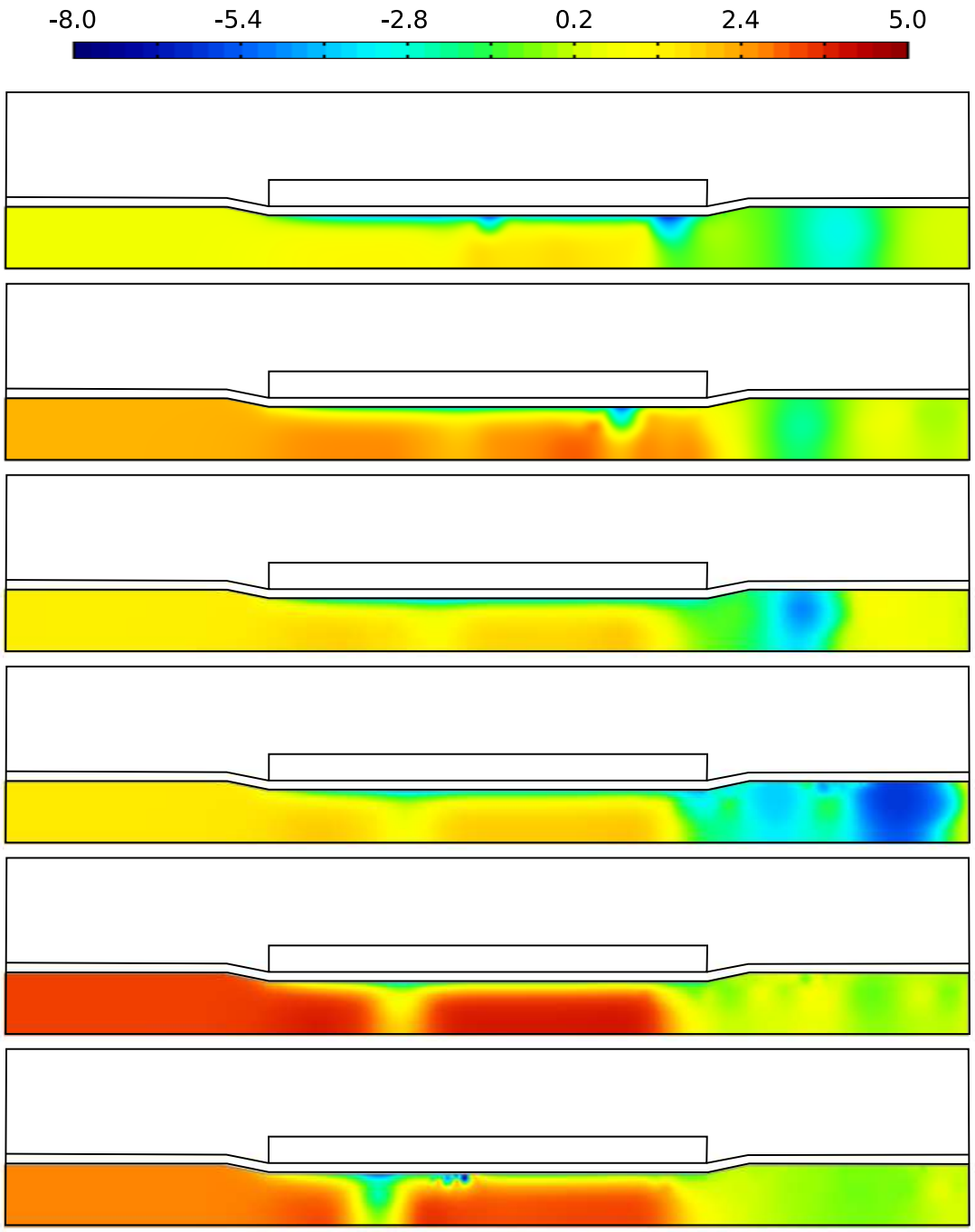



Figure 6.8: Pressure distribution $p$ (Pa) at $t = 0.5, 1.0, 1.5, 2.0, 2.5, 3.0$s.

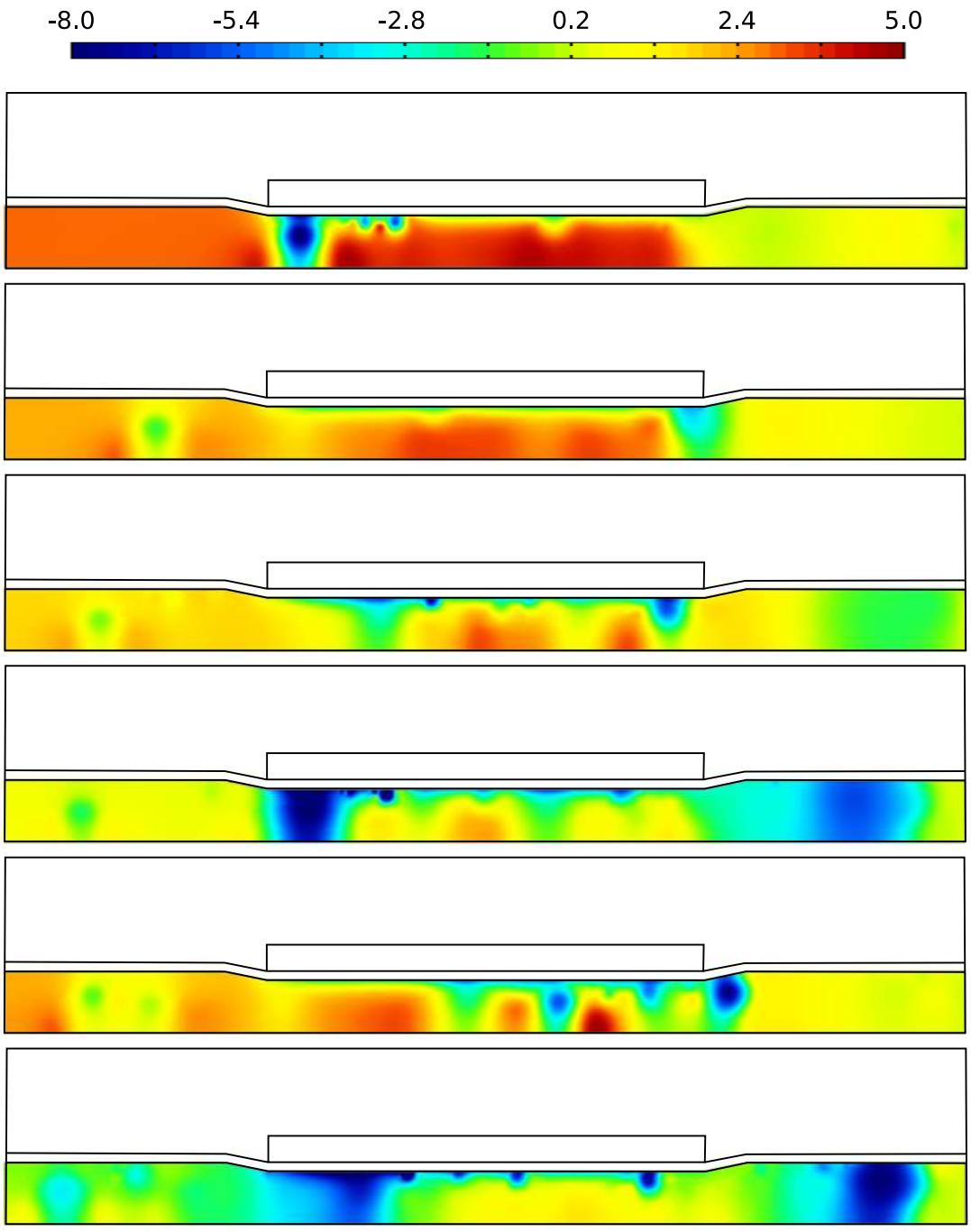



Figure 6.8: *(continued)* Pressure distribution $p$ (Pa) at $t = 3.5$, 4.0, 4.5, 5.0, 5.5, 6.0s.

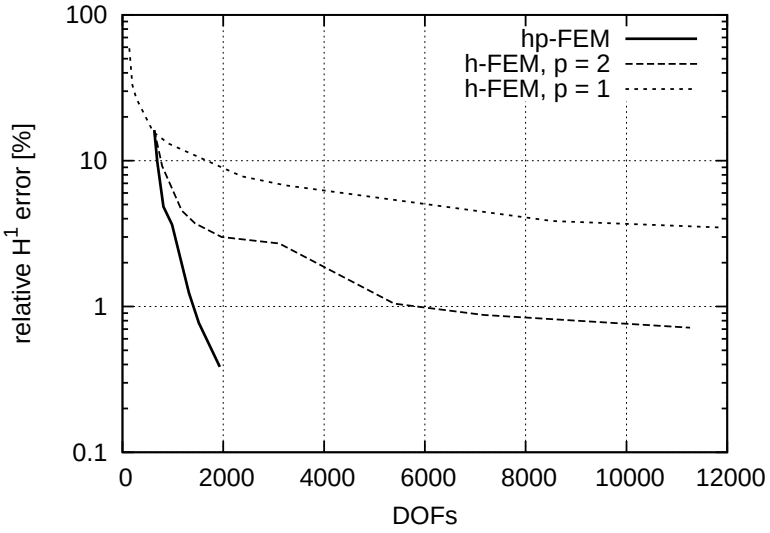



Figure 6.9: Convergence history of *hp*-adaptivity for magnetic potential.

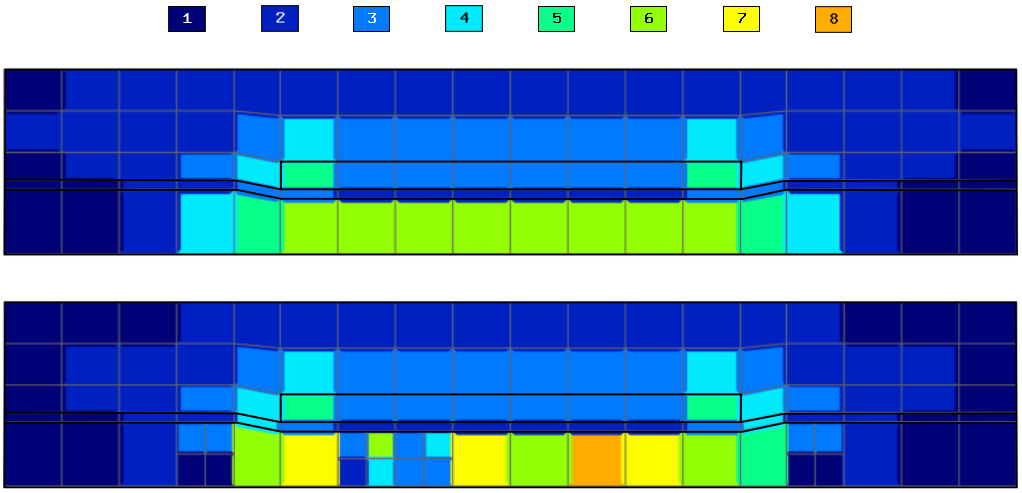



Figure 6.10: *hp*-meshes for magnetic potential at $t = 2.0$s and $t = 4.0$s.

temperature were approximated on *hp*-meshes automatically obtained by the adaptive algorithm. An average size of *hp*-meshes for the magnetic potential is around 2000 degrees of freedom (DOFs), while *h*-adaptivity on biquadratic elements would result in a mesh with around 11000 DOFs (see Fig. 6.9 for comparison of convergence history). Reaching the same level of accuracy with uniform mesh requires more than $10^5$ DOFs. Similar, however not that severe difference could be observed on *hp*/*h*-meshes for the temperature field. For approximation of the flow field standard *h*-FEM

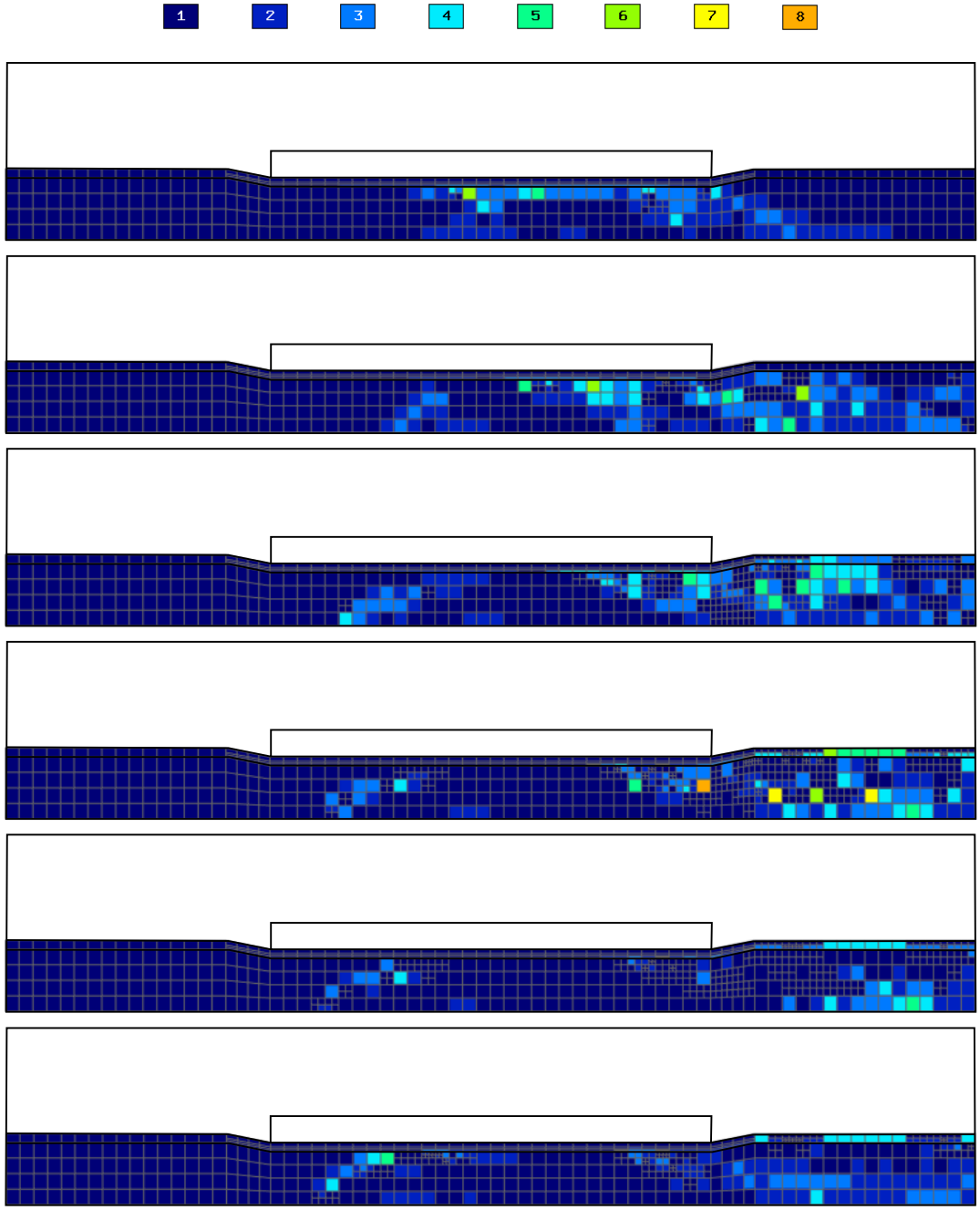



Figure 6.11: *hp*-meshes for temperature at $t = 0.5, 1.0, 1.5, 2.0, 2.5, 3.0$s.

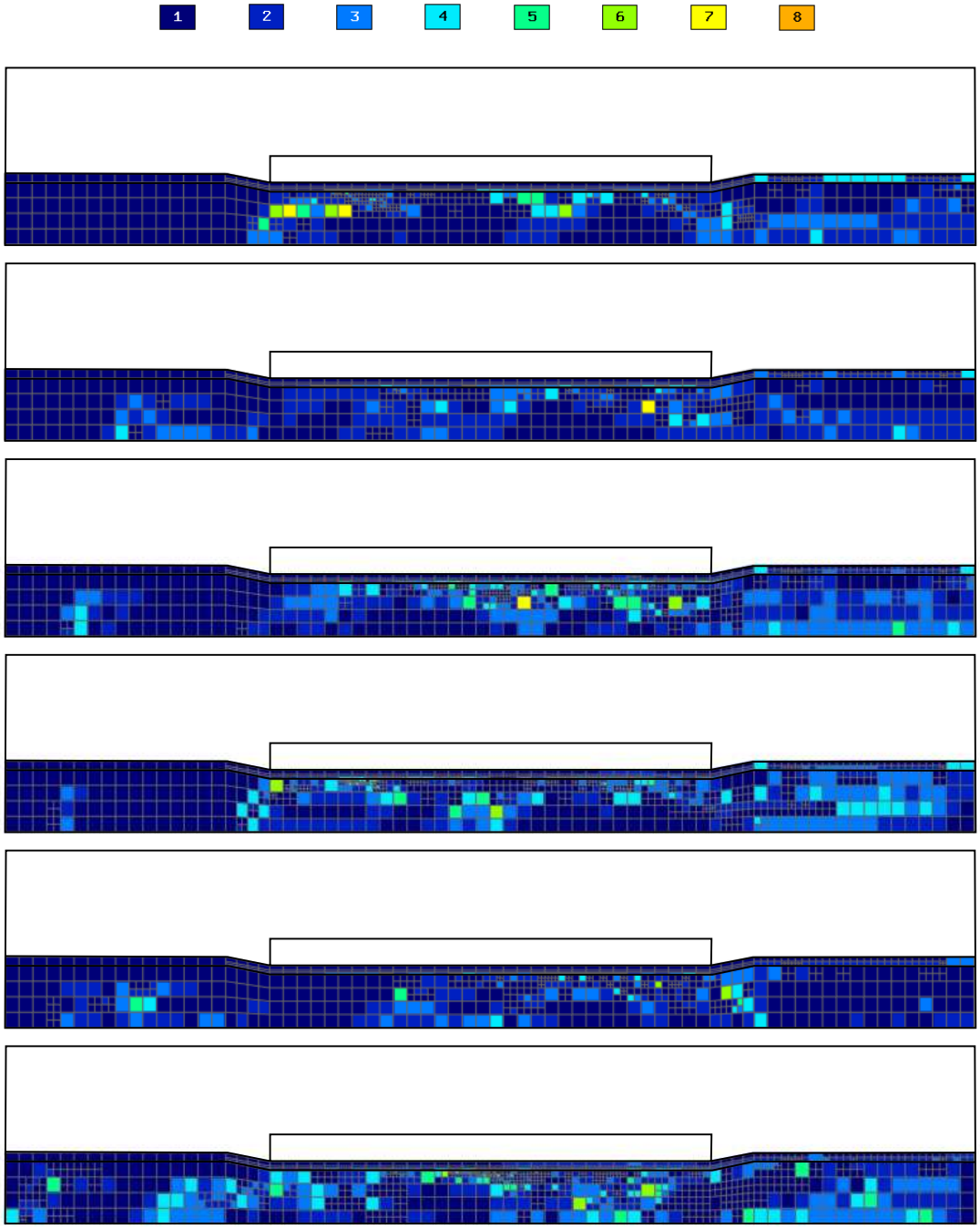



Figure 6.11: *(continued) hp*-meshes for temperature at $t = 3.5$, 4.0, 4.5, 5.0, 5.5, 6.0s.

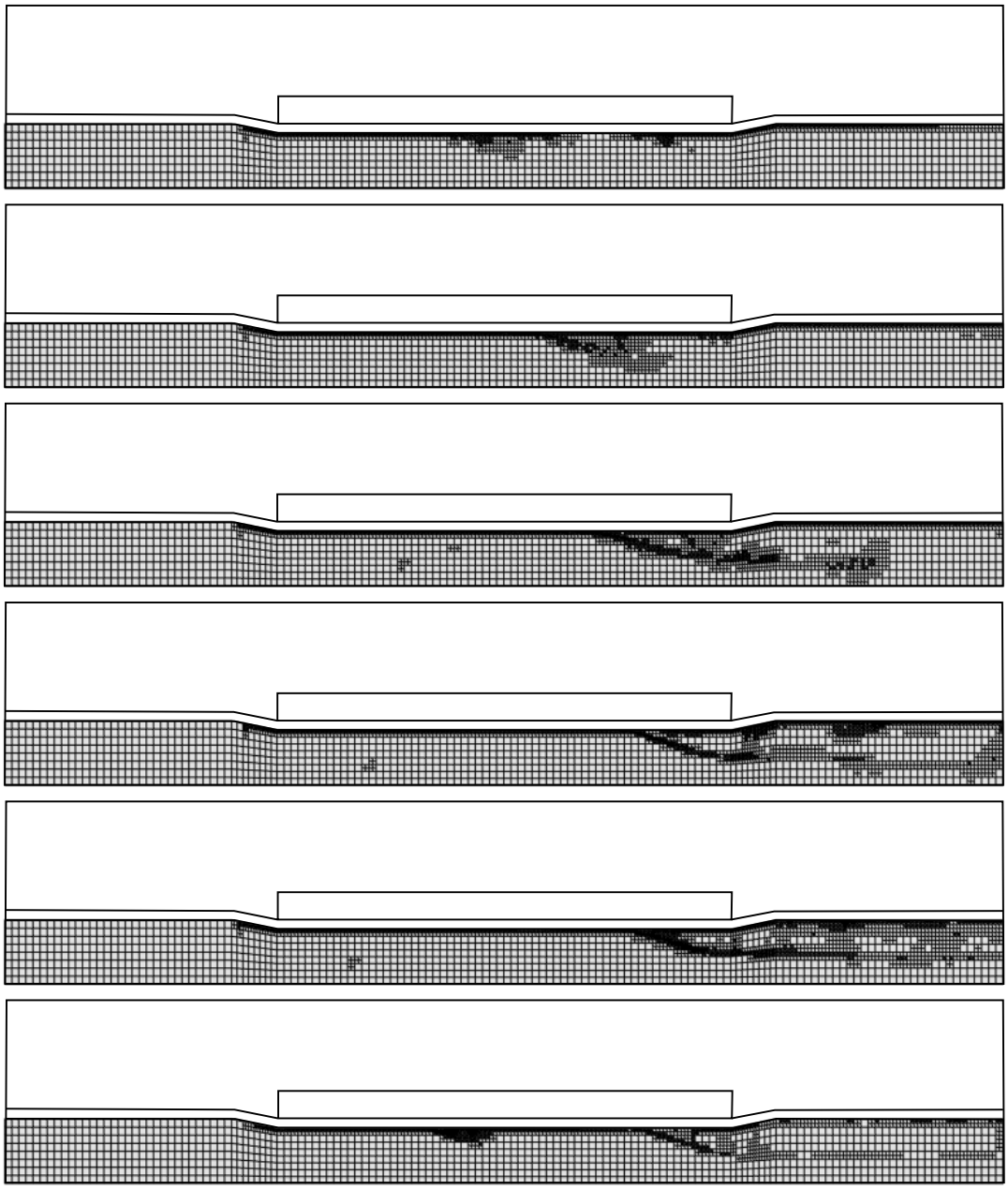


Figure 6.12: Dynamically changing meshes for velocity at time levels $t = 0.5$, 1.0, 1.5, 2.0, 2.5, 3.0s.

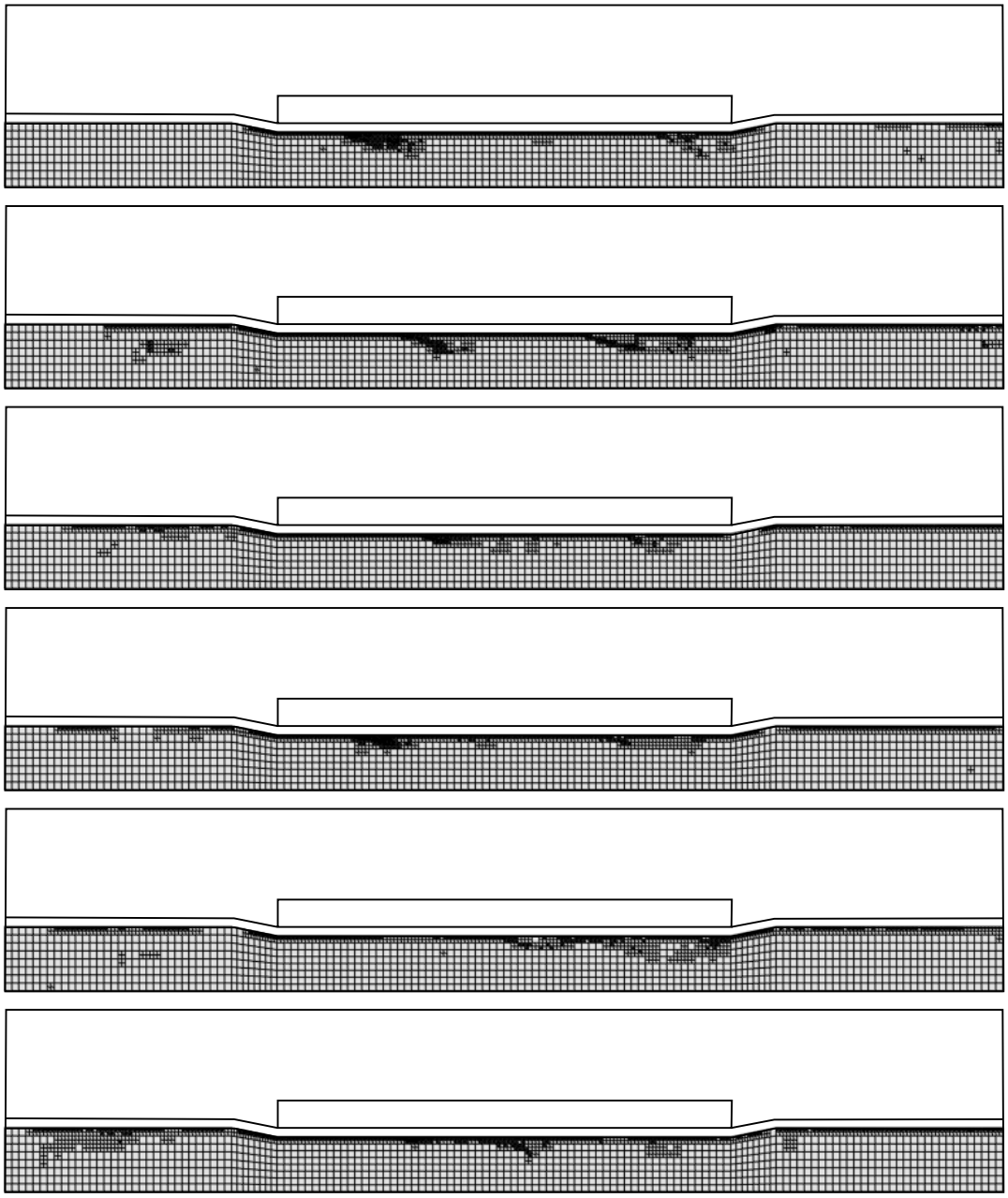


Figure 6.12: *(continued)* Dynamically changing meshes for velocity at time levels $t =$ 3.5, 4.0, 4.5, 5.0, 5.5, 6.0s.

adaptivity was used. Notice that dynamically changing meshes for flow field are traveling together with the moving vortices, resolving the flow features more accurately than by using a fixed uniform mesh. Similar behavior can be observed on dynamically changing $hp$-meshes for temperature, where higher-order elements are used in areas where the fluid is heated by the Joule's looses and the mesh stays unrefined in areas where the temperature of the fluid remains constant. Since the magnetic potential's changes in the fluid are small, we see that only minor changes in the underlying mesh are performed during the time. Notice, that by using the multimesh technology in the problem, we spent much less degrees of freedom in meshes for magnetic potential. They do not have to refine spatially in such a way the flow and temperature fields are.

Chapter 7

# Conclusion

We would like to conclude the thesis with a review of its results already published in international journals. Several remarks and possible ways how to improve our approach and how to continue in the future are discussed.

In Chapter 2 we gave a short introduction to the higher-order finite element discretization and described it also from the implementation point of view. Chapter 3 was concerned with the automatic $hp$-adaptivity for a single partial differential equation and the basic idea of meshes with arbitrary level hanging nodes was given. More details on $hp$-adaptivity, meshes with hanging nodes and extension of this strategy to electromagnetic problems solved in the space $H(\mathrm{curl})$ can be found in [46].

The main emphasis of the thesis lies in $hp$-adaptivity applied to coupled systems of PDEs. In such problems each component of the solution displays different behavior, and thus it requires the mesh to be refined in different parts of the computational domain. In [43, 14, 47] and in Chapter 4 we proposed the multimesh technology that allows us to discretize each physical field in the coupled problem on its own mesh, while the whole system is still solved monolithically. Utilizing the multimesh technology, there is no need for data-transfers between non-matching meshes. We compared our approach with traditionally used methods of operator-splitting and data-transfer. These comparisons will also be published in [15]. Comparison with the so-called single mesh approach, where all physical fields share the same mesh containing all necessary refinements, was demonstrated on a problem from civil-engineering that was published in [47]. Even though the multimesh approach has its drawbacks, such as the computational cost of numerical quadrature addressed in Section 4.5.1, we believe that its advantages are stronger.

Very recently researchers from the TU Dresden implemented the multimesh approach for nodal finite elements with application to dendritic growth and solid-solid phase-transitions. Their assembling technique is based on expressing parts of shape functions on subelements of the union mesh as linear combinations of standard shape functions. However, for their approach to work the meshes for all components must

be equipped with the same polynomial orders. In submitted paper [49] they cited our works [43, 47].

Chapter 5 discussed time-dependent coupled problems from the point of view of *hp*-adaptivity and the multimesh technology. An adaptive strategy was used to find an optimal mesh for the solution on each time level and the multimesh technology easily allowed meshes to change between subsequent time steps with no transfer of solution values between meshes. Presented results will be published in [45] in the near future.

In Chapter 6 we succeeded in applying most of the presented methods to a complex multiphysics problem. Inductively heated flow of a liquid metal couples three physical fields – magnetic field, temperature and incompressible flow. In agreement with previous chapters, meshes for all three physics were obtained automatically by our adaptive algorithm, hence each field was discretized on its own mesh which was tailored to the needs of the solution component. To capture the solution evolving in time, dynamically changing meshes were obtained using the multimesh technology. This model, applied to several problems coupling magnetic field, temperature and incompressible flow, was presented in the following conference proceedings: [7, 8, 13].

A significant benefit of this thesis is also in numerous contributions to the open source project Hermes2d, a numerical C++ library for the solution of multiphysics problems described by partial differential equations. Computations performed in this thesis were obtained by this library and result from author's implementations in the Hermes2d core. Since it became an open source software in 2009, students and researchers from all over the world can use it as well as contribute to it. It may be used both for education purposes at universities and for the solution of complex scientific problems.

There are many possible improvements to the presented approach, some of them were mentioned in the thesis. First let us point out the need to speed up the computation of the reference solution. This was already partially studied in [40], where the reference solution was obtained by hierarchically extending the previous reference solution with promissing results in 1D. Another approach that is being studied at University of Nevada at Reno consists in skipping the direct computation of the coarse solution and replacing it by a projection of the reference solution on the coarse mesh. In case the problem is nonlinear, the projection is faster than performing the Newton's method for the coarse problem.

Concerning the multimesh technology the main drawback lies in numerical quadrature in cases where the meshes differ considerably in both geometry and polynomial orders. A possible cure was proposed in Section 4.5.1. It consists in reducing the order of integration when integrating just a small part of the shape function.

There is still a lot of room for improvement in numerical solution of the resulting linear/nonlinear systems of algebraic equations. At this moment direct solvers are used for the solution of linear systems arising from our FEM discretizations since

their performance on matrices obtained by *hp*-FEM was very good. Nevertheless, iterative solvers are a necessity for very complex engineering problems as well as for the 3D version of the Hermes code, where the resulting matrices are too large for direct solvers. More research also can be done in the Jacobian-free Newton Krylov method and physics-based preconditioning for nonlinear coupled problems solved by the multimesh *hp*-FEM, since their performance on traditional linear FEM shows promising results [29, 21, 30].

# References


[1] A. Alemany, P. Marty, and J.P. Thibault: *Transfer Phenomena in Magnetohydrodynamic and Electroconducting Flow*, Kluwer Academic Publishers, 1999.

[2] I. Babuška, W. Gui: The h, p and hp-versions of the finite element method in 1 dimension-Part I. The error analysis of the p-version, Numer. Math., Volume 49, Issue 6, 1986, Pages 577-612.

[3] I. Babuška, W. Gui: The h, p and hp-versions of the finite element method in 1 dimension-Part II. The error analysis of the h and hp-versions, Numer. Math., Volume 49, Issue 6, 1986, Pages 613-657.

[4] I. Babuška, W. Gui: The h, p and hp-versions of the finite element method in 1 dimension-Part III. The adaptive hp-version, Numer. Math., Volume 49, Issue 6, 1986, Pages 659-683.

[5] W. Bangerth, R. Hartmann, and G. Kanschat: deal.II-A general-purpose object-oriented finite element library, ACM Trans. Math. Softw. Volume 33, Issue 4, August 2007, Pages 24.

[6] R. Černý, P. Rovnaníková: *Transport Processes in Concrete*, Spon Press, London, 2002.

[7] J. Červený, L. Dubcová, I. Doležel, P. Karban, J. Barglik: Flow of molten metal in a pipe driven by electromagnetic field, Proceedings of 11th Spanish Portuguese Conference on Electrical Engineering – 11CHLIE, Zaragoza, Spain, CD-ROM, 1-4 July, 2009.

[8] J. Červený, I. Doležel, L. Dubcová, P. Karban, P. Šolín: Higher-order Finite Element Modeling of Electromagnetically Driven Flow of Molten Metal in Pipe, Proceedings of the International Conference on Electrical Machines and Systems – ICEMS, Tokyo, Japan, CD-ROM, November 2009.

[9] P.A. Davidson: *An Introduction to Magnetohydrodynamics*, Cambridge University Press, 2001.

[10] L. Demkowicz: *Computing with hp-adaptive finite elements*, Chapman & HAL-L/CRC, Applied mathematics and nonlinear science series, 2007.

[11] L. Demkowicz, W. Rachowicz, P. Devloo: A Fully Automatic *hp*-Adaptivity, J. Sci. Comput., Volume 17, Issues 1-4, December 2002, Pages 117-142.

[12] P. Deuflhard, F. Bornemann: *Scientific Computing with Ordinary Differential Equations*, Springer, 2001.

[13] I. Doležel, L. Dubcová, P. Karban, J. Červený, P. Šolín: Inductively Heated Incompressible Flow of Electrically Conductive Liquid in Pipe, Proceedings of 17th International Conference on the Computation of Electromagnetics Field – COMPUMAG, Florianopolis, Brazil, CD-ROM, November 2009.

[14] L. Dubcová, P. Šolín, J. Červený, P. Kůs: Space and Time Adaptive Two-Mesh hp-FEM for Transient Microwave Heating Problems, Electromagnetics, Volume 30, Issue 1-2, January 2010, Pages 23-40.

[15] L. Dubcová, P. Šolín, G. Hansen, H. Park: Comparison of Multimesh hp-FEM to Interpolation and Projection Methods for Spatial Coupling of Reactor Thermal and Neutron Diffusion Calculations, submitted to J. Comput. Phys.

[16] E. Erturk: Numerical solutions of 2-D steady incompressible flow over a backward-facing step, Part I: High Reynolds number solutions, Computers & Fluids, Volume 37, Issue 6, July 2008, Pages 633-655.

[17] D. Estep, V. Ginting, D. Ropp, J.N.Shadid, S.Tavener: An A Posteriori-A Priori Analysis of Multiscale Operator Splitting, SIAM Journal on Numerical Analysis, Volume 46, Issue 3, March 2008, Pages 116 - 1146.

[18] M. Feistauer: *Mathematical Methods in Fluid Dynamics*, Longman Scientific & Technical, Harlow, 1993.

[19] C. A. Felippa, K. C. Park, C. Farhat: Partitioned analysis of coupled mechanical systems, Computer Methods in Applied Mechanics and Engineering, Volume 190, Issues 24-25, 2 March 2001, Pages 3247-3270.

[20] P. Frauenfelder, Ch. Lage: Concepts-An Object-Oriented Software Package for Partial Differential Equations, M2AN Volume 36, Issue 5, 2002, Pages 937-951.

[21] D. Gaston, C. Newman, G. Hansen, D. Lebrun-Grandie: MOOSE: A parallel computational framework for coupled systems of nonlinear equations, Nuclear Engineering and Design, Volume 239, Issue 10, October 2009, Pages 1768-1778.

[22] M. W. Gee, C. M. Siefert, J. Hu, R. S. Tuminaro, M. Sala: ML 5.0 Smoothed Aggregation User's Guide, Tech. Report SAND2006-2649, Sandia National Laboratories (May 2006).

[23] W. Gui, I. Babuška: The h, p and h-p versions of the finite element methods in 1 dimension. Part III. The adaptive h-p version, Numerische Mathematik, Volume 49, Issue 6, 1986, Pages 659 - 683.

[24] E. Hairer, S.P. Norsett, G. Wanner: *Solving Ordinary Differential Equations, I: Nonstiff Problems*, Springer Ser. Comput. Math., 8, Springer-Verlag, Heidelberg, 1987.

[25] E. Hairer, G. Wanner: *Solving Ordinary Differential Equations, II: Stiff and Differential-Algebraic Problems*, Springer Ser. Comput. Math., 14, Springer-Verlag, Heidelberg, 1991.

[26] G. Hansen, S. Owen: Mesh generation technology for nuclear reactor simulation; barriers and opportunities, Nuclear Engineering and Design, Volume 238, Issue 10, October 2008, Pages 2590-2605.

[27] M. Hughes, K. A. Pericleous, M. Cross: The numerical modelling of DC electromagnetic pump and brake flow, Applied Mathematical Modelling, Volume 19, Issue 12, December 1995, Pages 713-723.

[28] X. Jiao, M. T. Heath: Common-refinement-based data transfer between nonmatching meshes in multiphysics simulations, Int. J. Numer. Meth Engng, Volume 61, Issue 14, 2004, Pages 2402-2427.

[29] D.A. Knoll, D.E. Keyes: Jacobian-free Newton-Krylov methods: a survey of approaches and applications, Journal of Computational Physics, Volume 193, Issue 2, 2004, Pages 357-397.

[30] V.A. Mouseau, D.A. Knoll, W.J. Rider: Physics based preconditioning and the Newton-Krylov method for nonequilibrium radiation diffusion, J. Comput. Phys., Volume 160, Issue 2, May 2000, Pages 743-765.

[31] D. Pardo, L. Demkowicz: Integration of hp-Adaptivity and a Two Grid Solver for Elliptic Problems, Computer Methods in Applied Mechanics and Engineering, Volume 195, Issues 7-8, January 2006, Pages 674-710.

[32] M. Paszynski, J. Kurtz, L. Demkowicz: Parallel, fully automatic hp-adaptive 2d finite element package, Computer Methods in Applied Mechanics and Engineering, Volume 195, Issues 7-8, January 2006, Pages 711-741.

[33] K. Pericleous, V. Bojarevics, G. Djambazov, R.A. Harding, M. Wickins: Experimental and numerical study of the cold crucible melting process, Applied Mathematical Modelling, Volume 30, Issue 11, November 2006, Pages 1262-1280.

[34] W. Rachowicz, L. Demkowicz: An *hp*-adaptive finite element method for electromagnetics. Part II. A 3D implementation, Internat. J. Numer. Methods Engrg., Volume 53, Issue 1, 2002, Pages 147-180.

[35] J. C. Ragusa, V. S. Mahadevan: Consistent and accurate schemes for coupled neutronics thermal-hydraulics reactor analysis, Nuclear Engineering and Design, Volume 239, Issue 3, March 2009, Pages 566-579.

[36] J. I. Ramos, N. S. Winowich: Finite difference and finite element methods for mhd channel flows, International Journal for Numerical Methods in Fluids, Volume 11, Issue 6, 1990, Pages 907-934.

[37] M. Schmich, B. Vexler: Adaptivity with Dynamic Meshes for Space-Time Finite Element Discretizations of Parabolic Equations, SIAM J. Sci. Comput., Volume 30, Issue 1, November 2007, Pages 369-393.

[38] C. Schwab: *p- and hp-Finite Element Methods, Theory and Applications to Solid and Fluid Mechanics*, Oxford University Press, New York,1998.

[39] P. Šolín: *Partial Differential Equations and the Finite Element Method*, J. Wiley & Sons, 2005.

[40] P. Šolín, D. Andrš, J. Červený, M. Šimko: PDE-independent adaptive hp-FEM based on hierarchic extension of finite element spaces, Journal of Computational and Applied Mathematics, Volume 233, Issue 12, Finite Element Methods in Engineering and Science (FEMTEC 2009), April 2010, Pages 3086-3094.

[41] P. Šolín, J. Červený, I. Doležel: Arbitrary-Level Hanging Nodes and Automatic Adaptivity in the *hp*-FEM, Mathematics and Computers in Simulation, Volume 77, Issue 1, February 2008, Pages 117-132.

[42] P. Šolín, J. Červený, L. Dubcová: Higher-Order Edge Elements Based on Generalized Eigenfunctions of the Curl-Curl Operator, Research Report No. 2007-05, Department of Math. Sciences, University of Texas at El Paso.

[43] P. Šolín, J. Červený, L. Dubcová, D. Andrš: Monolithic discretization of linear thermoelasticity problems via adaptive multimesh hp-FEM, Journal of Computational and Applied Mathematics, Volume 234, Issue 7, Fourth International Conference on Advanced COmputational Method in ENgineering (ACOMEN 2008), August 2010, Pages 2350-2357.

[44] P. Šolín, L. Demkowicz: Goal-oriented hp-adaptivity for elliptic problems, Computer Methods in Applied Mechanics and Engineering, Volume 193, Issues 6-8, February 2004, Pages 449-468.

[45] P. Šolín, L. Dubcová: Space-Time Adaptive hp-FEM on Dynamical Meshes with Application to a Flame Propagation Problem, in preparation.

[46] P. Šolín, L. Dubcová, J. Červený, I. Doležel: Adaptive hp-FEM with Arbitrary-Level Hanging Nodes for Maxwell's Equations, Adv. Appl. Math. Mech., Volume 2, Issue 4, August 2010, Pages 518-532.

[47] P. Šolín, L. Dubcová, J. Kruis: Adaptive hp-FEM with dynamical meshes for transient heat and moisture transfer problems, Journal of Computational and Applied Mathematics, Volume 233, Issue 12, Finite Element Methods in Engineering and Science (FEMTEC 2009), April 2010, Pages 3103-3112.

[48] P. Šolín, K. Segeth, I. Doležel: *Higher-Order Finite Element Methods*, Chapman & Hall/CRC Press, 2003.

[49] T. Witkowski, A. Voigt: A multi-mesh finite element method for Lagrange elements of arbitrary degree, 2010, arXiv:1005.4808v1 [math.NA].

[50] M. Zítka: *On Some Aspects of Adaptive Higher-Order Finite Element Method for Three-Dimensional Elliptic Problems*, Doctoral thesis, Charles University, Prague, 2008.