Available online at www.sciencedirect.com

ScienceDirect

Mathematics and Computers in Simulation 77 (2008) 117–132

MATHEMATICS AND COMPUTERS IN SIMULATION

www.elsevier.com/locate/matcom

Arbitrary-level hanging nodes and automatic adaptivity in the *hp*-FEM

Pavel Šolín [a,b,*], Jakub Červený [a,b], Ivo Doležel [b]

[a] *Department of Mathematical Sciences, University of Texas at El Paso, El Paso, TX 79968-0514, USA*
[b] *Institute of Thermomechanics, Academy of Sciences of the Czech Republic, Dolejškova 5, Prague, CZ 10800*

Received 4 June 2006; received in revised form 20 February 2007; accepted 21 February 2007
Available online 2 March 2007

Abstract

In this paper we present a new automatic adaptivity algorithm for the *hp*-FEM which is based on arbitrary-level hanging nodes and local element projections. The method is very simple to implement compared to other existing *hp*-adaptive strategies, while its performance is comparable or superior. This is demonstrated on several numerical examples which include the L-shape domain problem, a problem with internal layer, and the Girkmann problem of linear elasticity. With appropriate simplifications, the proposed technique can be applied to standard lower-order and spectral finite element methods.
© 2007 IMACS. Published by Elsevier B.V. All rights reserved.

Keywords: Constrained approximation; Hanging nodes; *hp*-FEM; Automatic *hp*-adaptivity

1. Introduction

The *hp*-FEM is a modern version of the Finite Element Method (FEM) which combines finite elements of variable size (h) and polynomial degree (p) in order to obtain fast exponential convergence [3–7,12,14,21,23,24,27]. The outstanding performance of the *hp*-FEM as well as its superiority over standard (low-order) FEM has been proven in a number of convincing non-academic applications during the past decade [9,10,12,23,24,29], and the method is becoming increasingly popular among practitioners.

However, there still are issues that need to be addressed before the *hp*-FEM can become standard in computational engineering software. One of the well-known drawbacks of the method is its algorithmic complexity and relatively high implementation cost. Current research in the *hp*-FEM covers a variety of topics, such as development of fast automatic adaptive strategies and goal-oriented adaptivity [18,20,23,24], design of optimal higher-order shape functions for various types of continuous and vector-valued finite elements required by various types of PDEs [2,19,26], parallelization and automatic load balancing [9,10,15], application to problems with boundary layers [16], preservation of nonnegativity of physically nonnegative quantities (discrete maximum principles) [11,25,28], etc.

In this study we present a feasible algorithm for the treatment of multiple-level hanging nodes and show its positive effects on both the speed of convergence of the *hp*-FEM and simplification of automatic *hp*-adaptivity. The paper is

* Corresponding author. Tel.: +1 915 760 5829.
E-mail addresses: solin@utep.edu (P. Šolín), jcerveny@utep.edu (J. Červený), dolezel@fel.cvut.cz (I. Doležel).
URLs: http://hpfem.math.utep.edu/ (P. Šolín), http://hpfem.math.utep.edu/ (J. Červený).

0378-4754/$32.00 © 2007 IMACS. Published by Elsevier B.V. All rights reserved.
doi:10.1016/j.matcom.2007.02.011

organized as follows: In Section 2 we introduce *irregularity rules* and show their relation to the speed of convergence of adaptive *hp*-FEM. The algorithmic treatment of arbitrary-level hanging nodes is described in Section 3. New automatic adaptive strategy for the *hp*-FEM based on arbitrary-level hanging nodes is presented in Section 4. Numerical examples and comparisons follow in Section 5.

2. Hanging nodes and irregularity rules

When working with regular meshes (where two elements either share a common vertex, common edge, or their intersection is empty), adaptivity often is done using the so-called *red–green refinement strategy* [1]. This technique first subdivides desired elements into geometrically convenient subelements with hanging nodes and then it eliminates the hanging nodes by forcing refinement of additional elements, as illustrated in Fig. 1.

This approach preserves the regularity of the mesh but at the same time it introduces elements with sharp angles which, in general, are not desirable in finite element analysis. Often this happens when repeated refinements occur in some part of the domain, e.g., toward a boundary layer or point singularity.

The “green” refinements can be avoided by introducing *hanging nodes,* i.e., by allowing *irregular meshes* where element vertices can lie in the interior of edges of other elements. To ease the computer implementation, most finite element codes working with hanging nodes limit the maximum difference of refinement levels of adjacent elements to one (1-*irregularity rule*)—see, e.g., Refs. [18,20,23]. In the following, by *k-irregularity rule* (or *k-level hanging nodes*) we mean this type of restriction where the maximum difference of refinement levels of adjacent elements is *k*. In this context, $k=0$ corresponds to adaptivity with regular meshes and $k=\infty$ to adaptivity with arbitrary-level hanging nodes.

It is illustrated in Fig. 2 that even the 1-irregularity rule does not avoid all forced refinements.

The amount of forced refinements in the mesh depends strongly on the level of hanging nodes.

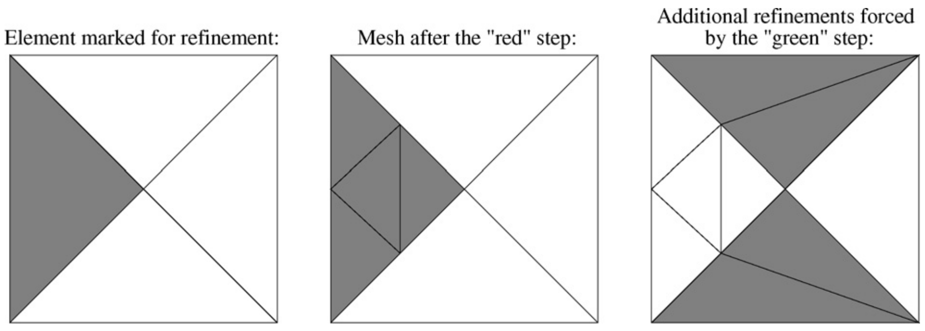
Element marked for refinement:
Mesh after the "red" step:
Additional refinements forced by the "green" step:

Fig. 1. Red–green refinement.

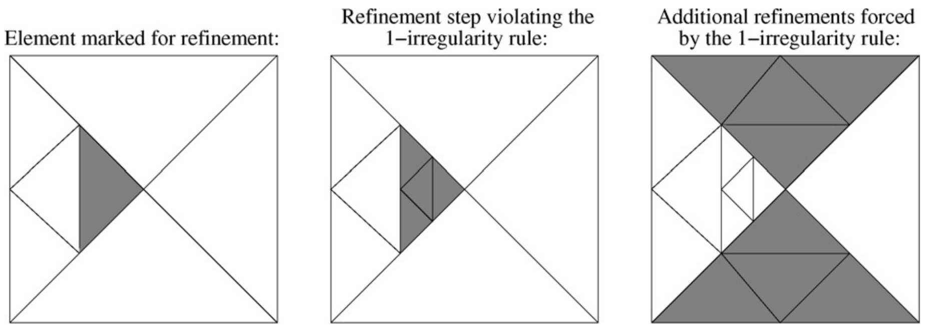
Element marked for refinement:
Refinement step violating the 1−irregularity rule:
Additional refinements forced by the 1−irregularity rule:

Fig. 2. Refinement with 1-irregularity rule.

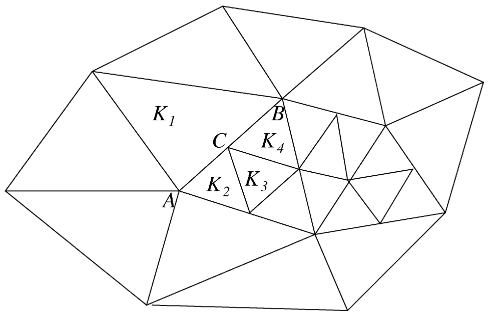

Fig. 3. Example of a mesh with one-level hanging nodes.

3. Arbitrary-level hanging nodes

The requirement of approximation continuity across an edge with hanging nodes is equivalent to a set of linear algebraic relations between the coefficients of the constraining and constrained shape functions. In Ref. [24], these relations were derived explicitly using transition matrices between pairs of constraining and constrained polynomial bases. This approach is very efficient as long as the maximum number of constraint levels is small (say, less than five). When the number of constraint levels becomes larger, the number of various geometrical situations on a constraining edge grows rapidly, and thus it becomes more efficient to calculate the constraining-constrained relations in specific situations on-demand. Therefore, we propose a new algorithm which is presented in Section 3.3. However, before discussing arbitrary-level hanging nodes, first let us explain the structure of one-level and multiple-level constraints in Sections 3.1 and 3.2.

3.1. One-level constraints

Assume a mesh edge AB containing one-level hanging nodes, as shown in Fig. 3. Let the polynomial degree of this edge be $p_{AB} \geq 1$.

There are $p_{AB} + 1$ constraining shape functions on the edge AB which induce three different situations:

I. Vertex function on K_1 corresponding to the vertex A. This shape function constrains the vertex functions on the elements K_2, K_3, K_4 corresponding to the vertex C.
II. Vertex function on K_1 corresponding to the vertex B. This shape function also constrains the vertex functions on the elements K_2, K_3, K_4 corresponding to the vertex C.
III. $p_{AB} - 1$ edge functions of polynomial degrees $p = 2, 3, \ldots, p_{AB}$ on K_1 corresponding to the edge AB. Edge function of degree p constrains
   (a) $p - 1$ edge functions on the element K_2 corresponding to the edge AC,
   (b) vertex functions on the elements K_2, K_3, and K_4 corresponding to the vertex C,
   (c) $p - 1$ edge functions on the element K_4 corresponding to the edge CB.

Notice that interior shape functions (bubble functions) which vanish on element boundaries are neither constraining nor constrained. Fig. 4 illustrates the simplest situation with $p_{AB} = 2$.

3.2. Multiple-level constraints

The case of multiple-level constraints is more interesting. Consider, for example, a mesh with four-level hanging nodes shown in Fig. 5.

Now there are $p_{AB} + 1$ constraining shape functions on K_1 associated with the edge AB. Each of them induces two types of constraints:

- *Direct constraints.* These include constrained vertex functions associated with all vertices in the interior of the edge AB (vertices C_1, C_4, C_7, C_2) and $p_{AB} - 1$ constrained edge functions for each subedge of AB (edges AC_1, C_1C_4, C_4C_7, C_7C_2, C_2B).

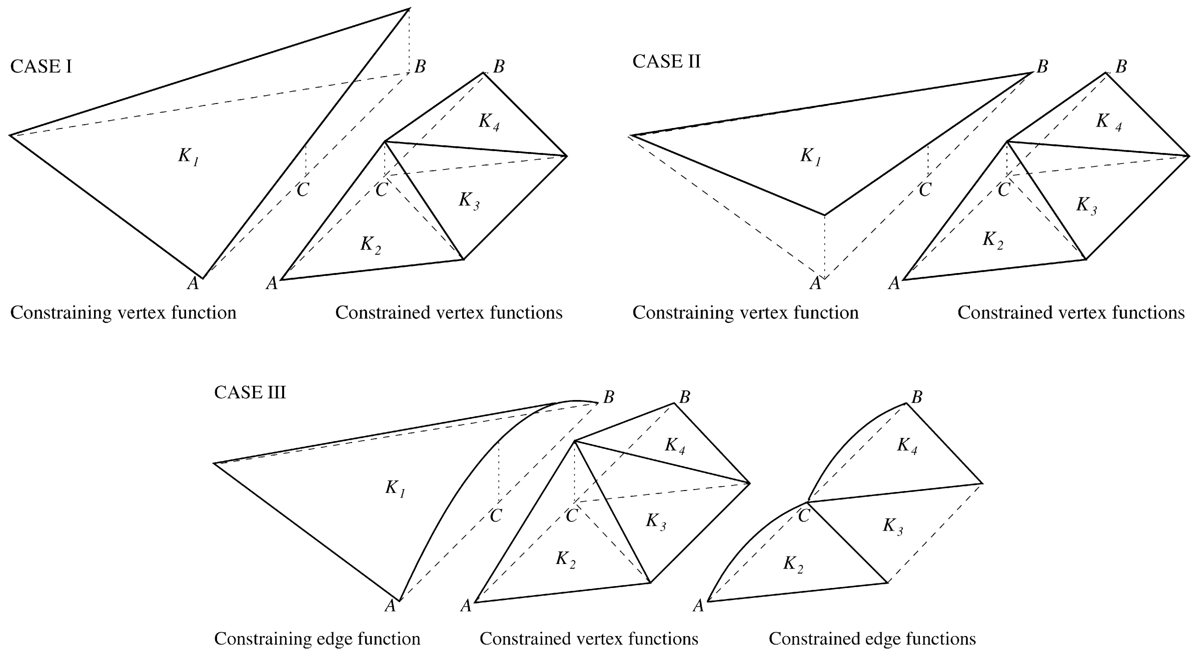
CASE I
CASE II
CASE III
K1
K2
K3
K4
A
B
C
Constraining vertex function
Constrained vertex functions
Constraining edge function
Constrained edge functions

Fig. 4. Constraining and constrained shape functions on a quadratic edge AB with one-level hanging nodes.

- *Implied constraints.* Constrained vertex functions may further constrain vertex functions associated with vertices which do not lie on the edge AB (vertices C_3, C_5, C_6, C_8). Vertex functions subject to implied constraints may produce further implied constraints.

Note that constrained edge functions never imply additional constraints and that edge functions never are subject to implied constraints.

3.3. Algorithmic treatment

Arbitrary-level hanging nodes can be handled via a simple extension of standard connectivity arrays for element-by-element assembling procedures [24].

3.3.1. Extension of standard connectivity arrays

Recall that the element-by-element assembling procedure requires a unique enumeration of the basis functions $v_1, v_2, \ldots, v_N$ of the global finite element space $V_{h,p}$, $\dim(V_{h,p}) = N$ as well as a unique local enumeration of shape functions on the reference domain $\hat{K}$. On a reference triangle, for example, by φ^{v_k}, $k = 1, 2, 3$ we may denote the vertex functions and by $\varphi_2^{e_k}, \varphi_3^{e_k}, \ldots, \varphi_{p_k}^{e_k}$, $k = 1, 2, 3$ the edge functions associated with edge e_k of polynomial degree p_k. Then every element K_i in a regular mesh contains pointers to six nodes:

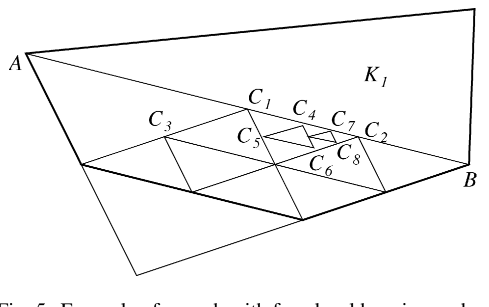

Fig. 5. Example of a mesh with four-level hanging nodes.

- Three vertex nodes

$$\boldsymbol{c}^{v_k} = \{[x_1^{v_k}, x_2^{v_k}], m^{v_k}\}, \quad k = 1, 2, 3, \tag{1}$$

where $[x_1^{v_k}, x_2^{v_k}]$ are the coordinates of v_k and m^{v_k} a connectivity index such that

$$\varphi^{v_k} = v_{m^{v_k}}(x_{K_i}), \quad k = 1, 2, 3.$$

(Here, $\boldsymbol{x}_{K_i} : \hat{K} \to K_i$ is the standard reference map.)

- Three edge nodes

$$\boldsymbol{c}^{e_k} = \{m_2^{e_k}, m_3^{e_k}, \dots, m_{p_k}^{e_k}\}, \quad k = 1, 2, 3, \tag{2}$$

where

$$\varphi_j^{e_k} = v_{m_j^{e_k}}(\boldsymbol{x}_{K_i}), \quad j = 2, 3, \dots, p_k.$$

In multiple-level constrained approximation, a vertex function φ^{v_k} is related to s_k basis functions of the finite element space $V_{h,p}$ via the relations

$$\varphi^{v_k} = \alpha_r^{v_k} v_{m_r^{v_k}}(\boldsymbol{x}_{K_i}), \quad r = 1, 2, \dots, s_j.$$

Accordingly, the index m^{v_k} in (1) is replaced with an array

$$\boldsymbol{m}^{v_k} = \{[\alpha_r^{v_k}, m_1^{v_k}], [\alpha_2^{v_k}, m_2^{v_k}], \dots, [\alpha_{s_k}^{v_k}, m_{s_k}^{v_k}]\}. \tag{3}$$

Edge connectivity arrays (2) are extended analogously.

3.3.2. Calculation of constraint coefficients

Let K_i be an element in the mesh whose edge $e_k = (v_r, v_s)$ is constrained by another mesh edge AB. On K_i we store an index q^{e_k} which uniquely identifies the geometrical position of e_k within AB. Notice that AB is uniquely oriented through the indices of the vertices A and B. Fig. 6 shows the enumeration of various geometrical cases.

The pair of immediate subedges of e_k have the indices $q_1^{e_k} = 2q^{e_k} + 2$ and $q_2^{e_k} = 2q^{e_k} + 3$. The index q^{e_k} identifies uniquely the coordinates of the relative endpoints $-1 \le a < b \le 1$ of e_k within AB.

Assume a polynomial shape function ψ of degree p_k on the edge AB. Restricted to the edge e^k, ψ determines the coefficients

$$\alpha^{v_r} = \psi(a), \qquad \alpha^{v_s} = \psi(b)$$

of the vertex functions φ^{v_r}, φ^{v_s} on K_i as well as the coefficients $\alpha_2^{e_k}, \alpha_3^{e_k}, \dots, \alpha_{p_k}^{e_k}$ of the $p_k - 1$ edge functions $\varphi_2^{e_k}, \varphi_3^{e_k}, \dots, \varphi_{p_k}^{e_k}$ on K_i. These values are obtained by solving a system of $p_k - 1$ linear algebraic equations of the form

$$\sum_{j=2}^{p_k} \alpha_j^{e_k} \varphi_j^{e_k}(y_i^{p_k}) = \psi(y_i^{p_k}) - \alpha^{v_r}\varphi^{v_r}(y_i^{p_k}) - \alpha^{v_s}\varphi^{v_s}(y_i^{p_k}), \quad 2 \le i \le p_k,$$

where $y_i^{p_k}$ are the $p_k - 1$ interior Chebyshev points of degree p_k on the edge e_k.

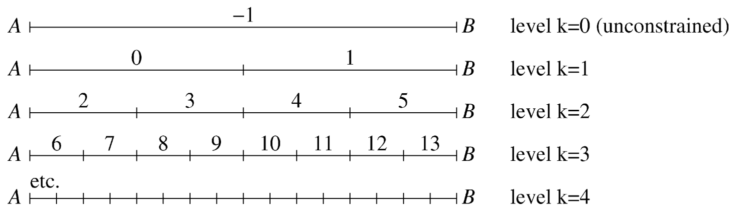

Fig. 6. Geometrical situations on a constraining edge.

3.3.3. *Construction of extended connectivity arrays*

The extended connectivity arrays are constructed in a recursive manner, starting with top-level nodes. Let $e_k = (v_r, v_s)$ be a constraining edge. Consider a vertex node v_k lying on e_k. The extended connectivity array $\boldsymbol{m}^{v_k}$ has the form

$$[\alpha_i^{v_k}, m_i^{v_k}] \in \boldsymbol{m}^{v_k} \Leftrightarrow [\beta, m_i^{v_k}] \in \boldsymbol{m}^{v_r} \text{ or } [\gamma, m_i^{v_k}] \in \boldsymbol{m}^{v_s} \text{ or } [\delta, m_i^{v_k}] \in \boldsymbol{m}_j^{e_k}$$

for some $\alpha_i^{v_k}, \beta, \gamma, \delta \in \mathbb{R}$, $1 \le i \le s_k$ and $2 \le j \le p_k$. Each coefficient $\alpha_i^{v_k}$ is defined by the first applicable rule in the following list:

(1) $\alpha_i^{v_k} = (\beta + \gamma)/2$, if $[\beta, m_i^{v_k}] \in \boldsymbol{m}^{v_r}$ and $[\gamma, m_i^{v_k}] \in \boldsymbol{m}^{v_s}$,
(2) $\alpha_i^{v_k} = \beta/2$, if $[\beta, m_i^{v_k}] \in \boldsymbol{m}^{v_r}$,
(3) $\alpha_i^{v_k} = \gamma/2$, if $[\gamma, m_i^{v_k}] \in \boldsymbol{m}^{v_s}$,
(4) $\alpha_i^{v_k} = \varphi^{e_k}(v_k)$,

where $\varphi^{e_k}(v_k)$ is the value of the constraining edge shape function at v_k. Note that v_r and v_s can themselves be either constrained or unconstrained. If unconstrained, their extended connectivity arrays are assumed to have the form $\boldsymbol{m}^{v_r} = \{[1, m^{v_r}]\}$, $\boldsymbol{m}^{v_s} = \{[1, m^{v_s}]\}$.

The reader does not have to implement these algorithms—they are contained in a modular C++ library HERMES which is designed to facilitate the implementation of adaptive *hp*-FEM for the solution of nonlinear coupled problems. For more details see the web page http://hpfem.math.utep.edu/hermes.

4. Automatic *hp*-adaptivity with arbitrary-level hanging nodes

The major difference between automatic adaptivity in standard FEM and in the *hp*-FEM is a large number of refinement options for higher-order elements. In standard *h*-adaptivity it is sufficient to use, e.g., the red–green refinement technique which was described in Section 2. In *hp*-adaptivity, one can either increase the polynomial degree of an element without subdividing it in space or one can split an element in space and distribute the polynomial degrees in the subelements in multiple ways.

This implies that traditional error estimates (one number per element) do not provide enough information to guide *hp*-adaptivity. One needs a better knowledge of the *shape* of the error function $e_{h,p} = u - u_{h,p}$. In principle, it might be possible to obtain this information from estimates of higher derivatives or using postprocessing techniques such as [8], but these approaches are not very practical. Usually it is easier to use a *reference solution*, i.e., an approximation u_{ref} which is at least one order more accurate than $u_{h,p}$ and which can be obtained from $u_{h,p}$ with some moderate computing effort. In practise, usually we enrich the FE space by increasing the polynomial degree of elements, in contrast to other authors [13,18] who also subdivide the elements. The *hp*-adaptivity is then guided by an a posteriori error estimate of the form

$$e_{h,p} = u_{\text{ref}} - u_{h,p} \tag{4}$$

In our implementation we follow the ideas [12,23,24] where u_{ref} is computed efficiently in an enriched finite element space. However, instead of an iterative solver combined with a two-level multigrid we apply a direct sparse solver (UMFPACK) and exploit the LU decomposition of the former reference solution via the Schur complement. While the two-grid approach is taylored for positive-definite problems, the latter is virtually equation-independent.

The outline of our *hp*-adaptivity algorithm is as follows:

(1) Assume an initial coarse mesh $\tau_{h,p}$ consisting of piecewise-quadratic elements. User input includes prescribed tolerance TOL > 0 for the energy norm of the approximate error function (4) as well as a number DDOF of degrees of freedom to be added in each *hp*-adaptivity step.
(2) Compute coarse mesh approximation $u_{h,p} \in V_{h,p}$ on $\tau_{h,p}$.
(3) Find reference solution $u_{\text{ref}} \in V_{\text{ref}}$, $V_{h,p} \subset V_{\text{ref}}$, where V_{ref} is an enriched finite element space.

(4) Construct the approximate error function (4), calculate its energy norm ERR_i on every element K_i in the mesh, $i = 1, 2, \ldots, M$. Calculate the approximation of the global error,

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

(5) If $\mathrm{ERR} \le \mathrm{TOL}$, stop computation and proceed to postprocessing.
(6) Sort all elements into a list L according to their value ERR_i in decreasing order.
(7) Until the number of added degrees of freedom is greater or equal to DDOF do:
  (a) Take next element K from the list L.
  (b) Perform *hp*-refinement of K (to be described in more detail below).
(8) Adjust polynomial degrees on unconstrained edges using the so-called *minimum rule* (every unconstrained edge is assigned the minimum of the polynomial degrees on the pair of adjacent elements).
(9) Continue with step 2.

Notice that the refinement of the element K in Step 7b is completely *local*, i.e., one does not have to take into consideration the situation on elements adjacent to K. This is possible thanks to the arbitrary-level hanging nodes and thus the presented algorithm is much simpler compared to other existing *hp*-adaptive strategies such as, e.g., Refs. [13,17,18].

4.1.1. Selection of optimal hp-refinement

For every element $K \in \tau_{h,p}$ of polynomial degree p we consider the following $\hat{N} = 83$ *hp*-refinement options:

(1) Increase the polynomial degree of K to $p+1$ without spatial subdivision.
(2) Increase the polynomial degree of K to $p+2$ without spatial subdivision.
(3) Split K into four similar triangles K_1, K_2, K_3, K_4. Define p_0 to be the integer part of $(p+1)/2$. For each K_i, $1 \le i \le 4$ consider polynomial degrees $p_0 \le p_i \le p_0 + 2$. Edges lying on the boundary of K inherit the polynomial degree p_j of the adjacent interior element K_j. Polynomial degrees on interior edges are determined using the minimum rule.

For each of these $\hat{N}$ options we perform a standard H^1-projection of the reference solution u_{ref} onto the corresponding piecewise-polynomial space on the element K. The candidate with minimum projection error is selected. Notice that each of these $\hat{N}$ projection problems requires the solution of a small linear system. These systems are solved efficiently by exploiting the embedded structure of the projection matrices in combination with the hierarchic property of the Cholesky factorization. The algorithmic complexity of this step is linear (more precisely, it is Cn where n is the number of elements to be refined and C is a fixed constant depending on the number of element refinement options). In practice, the CPU time needed for the selection of optimal element refinements is roughly comparable to the solution time of the matrix problem on the coarse mesh.

Let us comment briefly on the extension to 3D. In our 3D code (which was not discussed here) we use hexahedral elements. Every hexahedral element can be split into 2, 4, or 8 smaller hexahedra in 7 different ways. If we allow for 3 polynomial options on every subelement, the total number of *hp*-refinement options is $\hat{N} \approx 6600$. When we only consider two polynomial options for every subelement, this large number drops to $\hat{N} \approx 300$. At the same time, however, the convergence curve becomes more flat and one needs more refinement steps in order to achieve a given accuracy. We do not know yet which approach is globally faster. The problem of minimizing the CPU time is a challenging nonlinear optimization problem which we plan to address in the future.

5. Numerical examples

In this section we present three numerical examples: First, in Sections 5.1 and 5.2 we compare the performance of different automatic adaptive strategies on well-known benchmark problems which possess exact solutions. Then, in Section 5.3 we apply the new *hp*-adaptive strategy based on arbitrary-level hanging nodes to tackle a challenging problem of linear elasticity—the Girkmann problem [27].

5.1. Example 1: L-shape domain problem

In this section we compare the performance of the new *hp*-adaptive strategy from Section 4 with another automatic *hp*-adaptive strategy [13]. For this purpose, we use the standard L-shape domain problem whose exact solution has a singular gradient (see, e.g., Ref. [27]). The computational domain is the well-known square missing a quarter, $\Omega = (-1, 1)^2 \backslash (-1, 0)^2 \subset \mathbb{R}^2$. We solve the Laplace equation $-\Delta u = 0$ in Ω with the Dirichlet boundary condition

$$u(\boldsymbol{x}) = R(\boldsymbol{x})^{2/3} \sin\left(\frac{2\theta(\boldsymbol{x})}{3} + \frac{\pi}{3}\right), \tag{5}$$

$\boldsymbol{x} \in \partial\Omega$. Here, $R(\boldsymbol{x})$ and $\theta(\boldsymbol{x})$ are the standard spherical coordinates in $\mathbb{R}^2$. The exact solution u given by (5) in Ω. We do not include the pictures of u or Ω here since they are well-known (see, e.g., Ref. [24]).

The approximation error is measured relatively as

$$E_{\text{rel}} = \frac{\|u - u_{h,p}\|_{H^1}}{\|u\|_{H^1}} \times 100\%, \tag{6}$$

where $u_{h,p}$ is the approximate solution and $\| \cdot \|_{H^1}$ is the standard H^1 norm.

For illustration, several steps of the adaptive algorithm are shown in Figs. 7–9 (the corresponding approximate solutions are not shown since they are visually identical to the exact solution.

We can see that in areas where the solution $u_{h,p}$ is smooth, the algorithm tends to use large elements with higher polynomial degrees. On the other hand, near the singularity at the re-entrant corner small elements with lower polynomial degrees are selected. This is in good correspondence with the theory of *hp*-FEM. A convergence comparison is shown in Fig. 10.

Here, the solid line represents the convergence history of the first 15 steps of the adaptive algorithm (note the logarithmic scale on the vertical axis). For comparison, we have extracted the results for the same problem from [13]. Even though our algorithm is considerably simpler than the one in [13], it gives comparable or better results. The "×" mark in Fig. 10 is a result obtained from a human-created *hp*-mesh from [22].

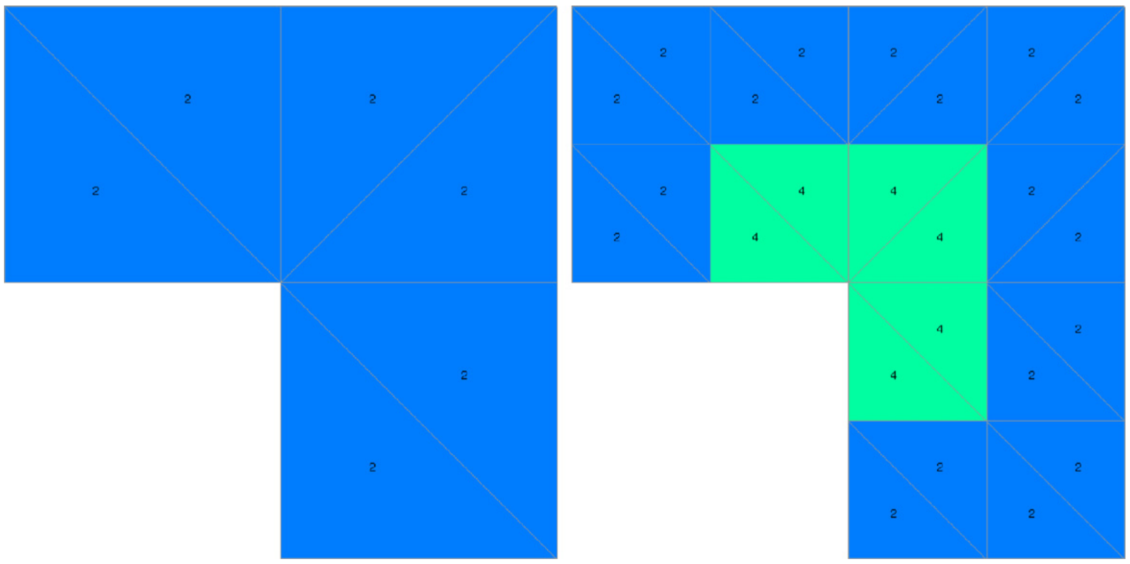

Fig. 7. The initial (piecewise-quadratic) six-element mesh and the mesh after the third step.

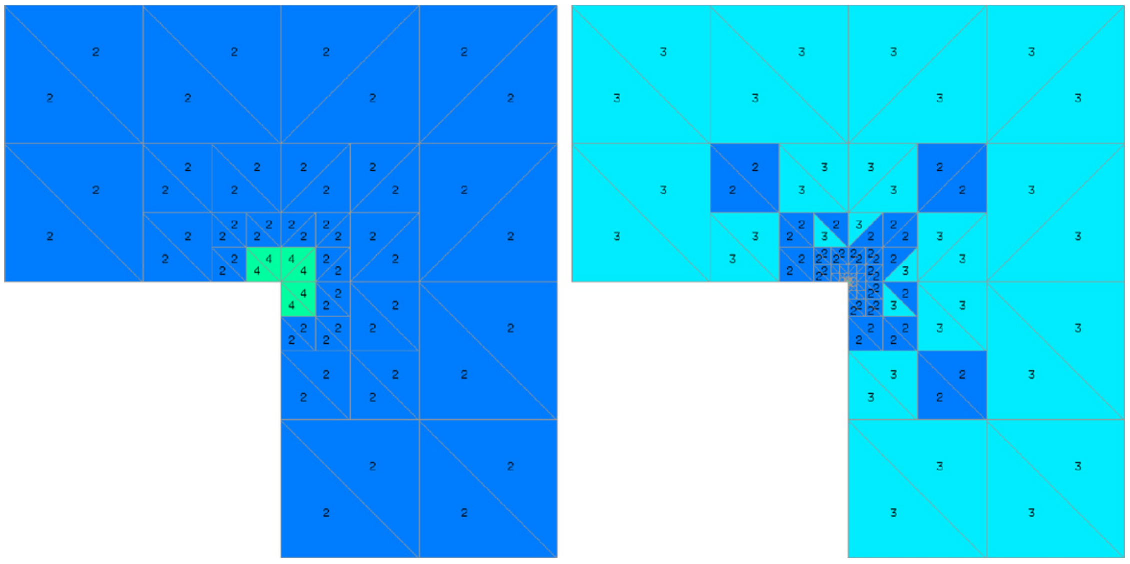
Fig. 8. Meshes after the fifth and eighth steps.

5.2. Example 2: problem with internal layer

The next problem is taken from [13]. Solved is the Poisson equation $-\Delta u = f$ in a unit square domain $\Omega = (0,1)^2$, where both the right-hand side f and the Dirichlet boundary condition on $\partial\Omega$ are constructed to match the exact solution $u(x,y) = \text{atan}(60(R(x,y) - \pi/3))$. Here, $R(x,y)$ is the radial component of a polar coordinate with origin in the point (5/4, −1/4), i.e., $u(x, y) = \text{atan}(60(\sqrt{(x - 5/4)^2 + (y + 1/4)^2} - \pi/3))$.

Fig. 11 shows successive approximate solutions $u_{h,p}$ and the corresponding hp-meshes. In particular, notice the formation of multiple-level hanging nodes.

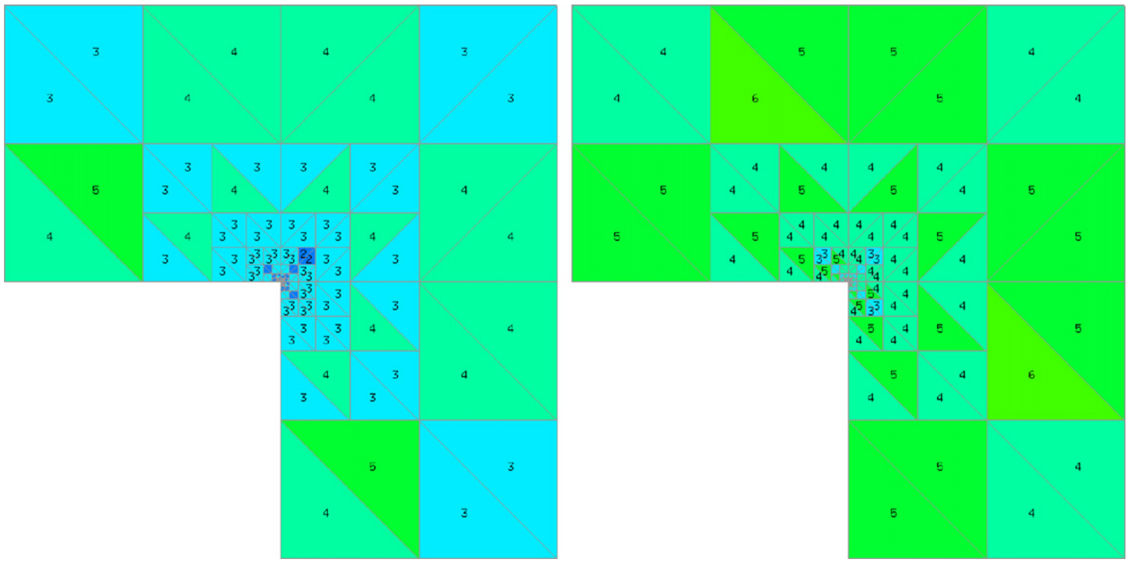
Fig. 9. Meshes after the 12th and 15th steps.

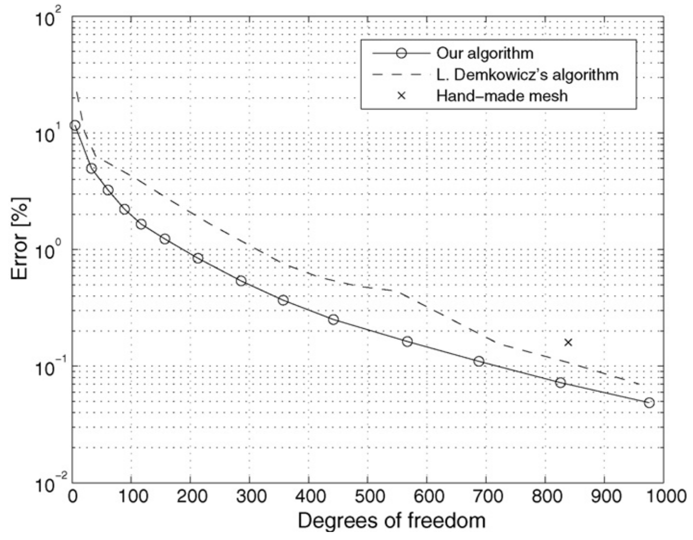

Fig. 10. Convergence history for the L-shape domain problem.

The reader can see in Fig. 12 that in this case the performance of our algorithm is similar to the performance of algorithm [13]. Fig. 13 shows that the convergence rates of the *hp*-FEM are significantly faster that those of standard adaptive FEM with piecewise-quadratic elements.

5.3. *Example 3: The Girkmann problem*

The next problem was published and solved using classical civil engineering methods by Girkmann in 1959. The computation is concerned with the elastic behavior of a spherical dome supported by a stiffening ring (see ref. [27], p. 327).

The geometry shown in Fig. 14 is axisymmetric, with the values $\alpha = 40^\circ$, $H = 6$ cm, $R = 15$ m, $|AB| = 60$ cm, $|BC| = 50$ cm. The point E lies on the line connecting D with the center of the sphere. The structure (both the dome and the ring) is made of concrete with the Poisson ratio $\nu = 0$ and the Young modulus $E = 30$ GPa. The density of concrete is assumed to be 3333 kg/m^3.

The structure is modelled using the Lamé equations of elasticity equipped with the boundary conditions $u_r = 0$, $\partial u_z/\partial r = 0$ on the axis of symmetry and $u_z = u_r = 0$ on the bottom of the stiffening ring. Sought is the total resultant force exerted by the dome on the ring along the line DE. The challenge in this problem is to resolve accurately two strong singularities at the points D and E. Additional (weaker) singularities at the points A, B are induced by the boundary conditions. Sought is the displacement $\boldsymbol{u}(r, z) = [u_r(r, z), u_z(r, z)]^{\mathrm{T}}$.

Fig. 15 shows the singularities in the Von Mises stress

$$\sigma = \frac{1}{\sqrt{2}}\sqrt{(\tau_{rr} - \tau_{zz})^2 + (\tau_{zz} - \tau_{\theta\theta})^2 + (\tau_{\theta\theta} - \tau_{rr})^2 + 6\tau_{rz}^2}$$

at the points D and E:

The meshes used for both displacement components u_r and u_z are geometrically identical, but we allow the adaptive algorithm to use elements of independent polynomial degrees for each of them. Figs. 16 and 17 show the distribution of the polynomial degrees in several stages of the adaptive process for u_r and u_z, respectively.

Fig. 18 shows the convergence history of the *hp*-adaptive process. Fig. 19 shows the convergence of the integral over the surface circumscribed by the line DE in space, $\int \tau_{ij} n_i \, \mathrm{d}S = F_{h,p}$. Given the density of concrete 3333 kg/m^3, the weight of the dome is 192.3 tonnes. This means that the total force exerted on the ring is

$$F = mg = 192,300 \times 9.81\,\mathrm{N} = 1.886 \times 10^6\,\mathrm{N}.$$

We used this information to verify that the error in the total force shown in Fig. 19 was about 0.8%.

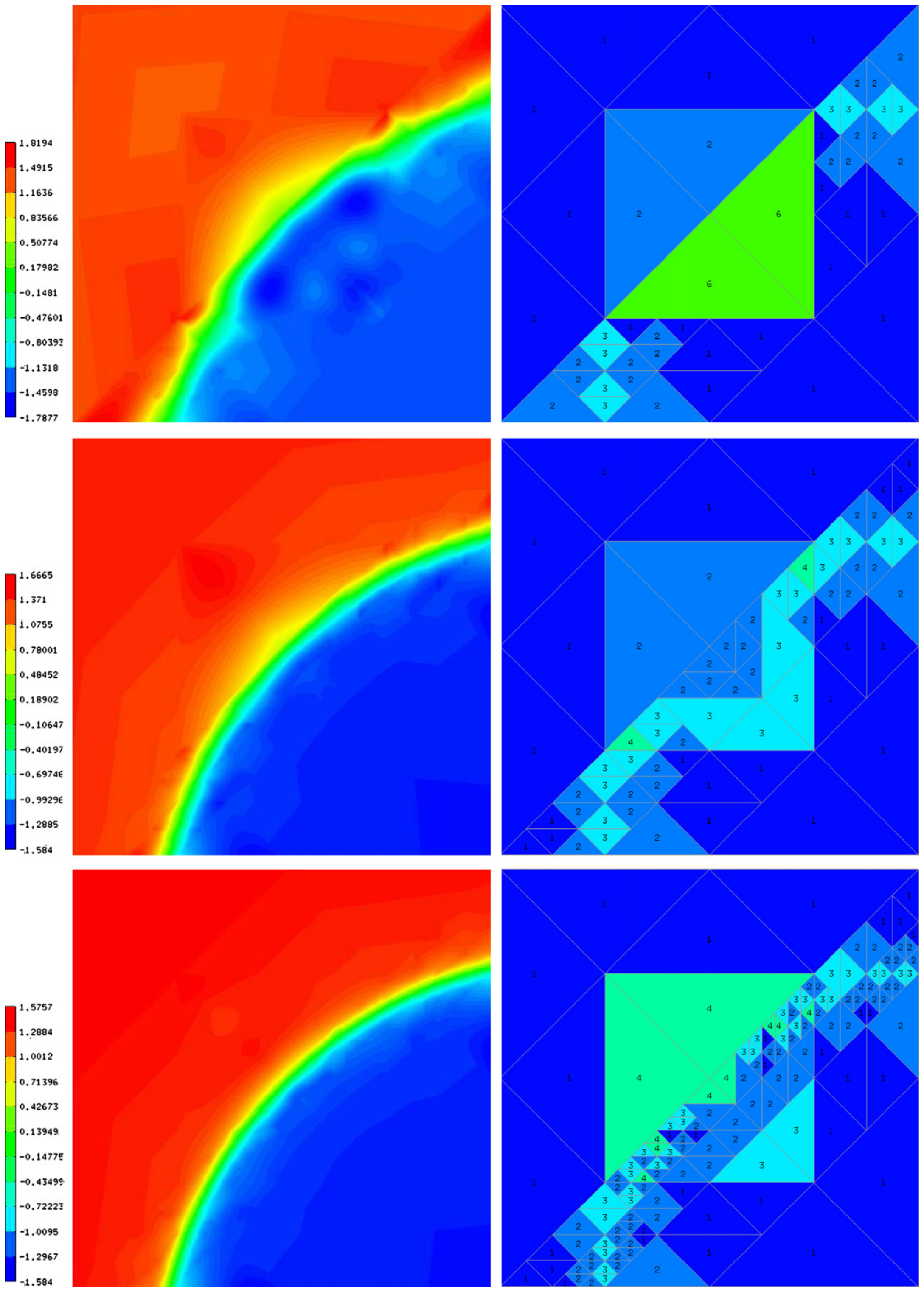

Fig. 11. Steps 5, 7, and 9 of the adaptive process.

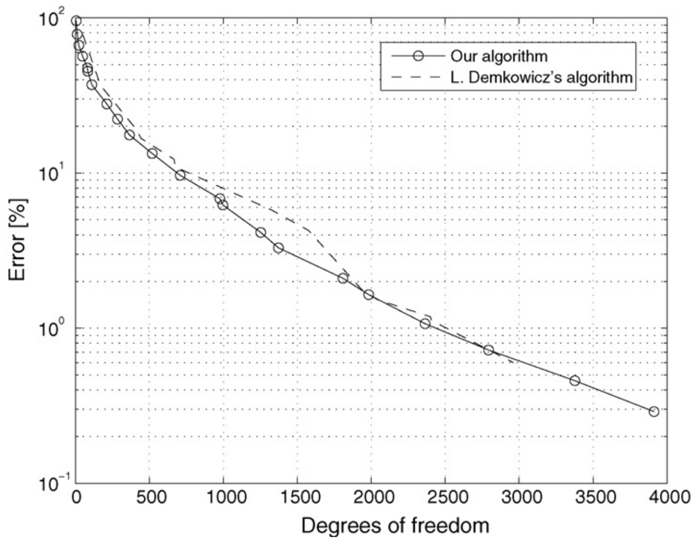

Fig. 12. Performance comparison for the internal layer problem.

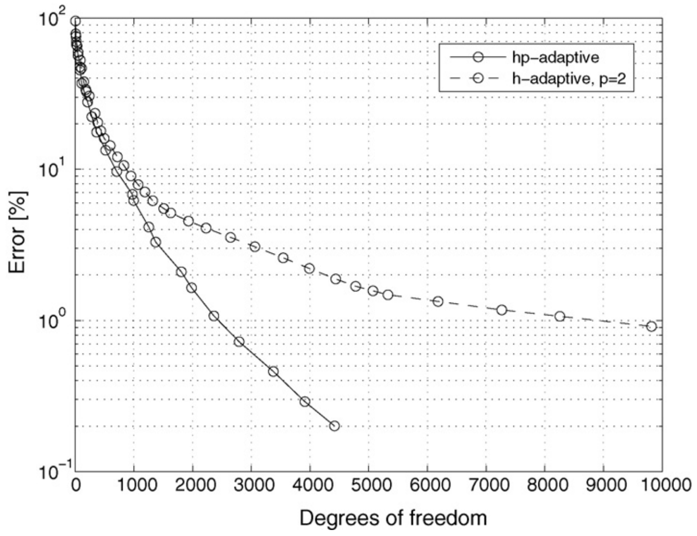

Fig. 13. Comparison of *hp*-adaptivity and *h*-adaptivity.

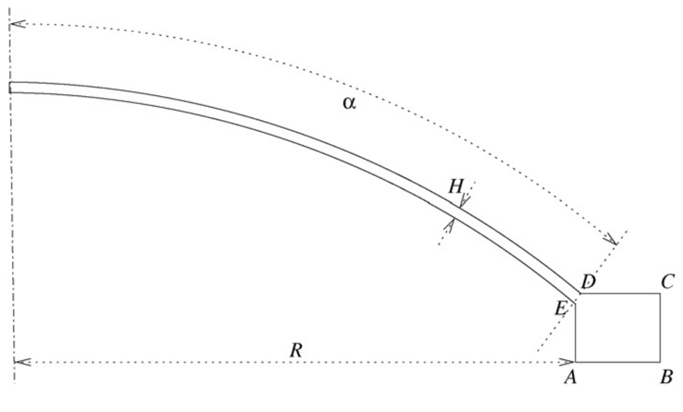

Fig. 14. Axisymmetric geometry of the Girkmann problem (not to scale).

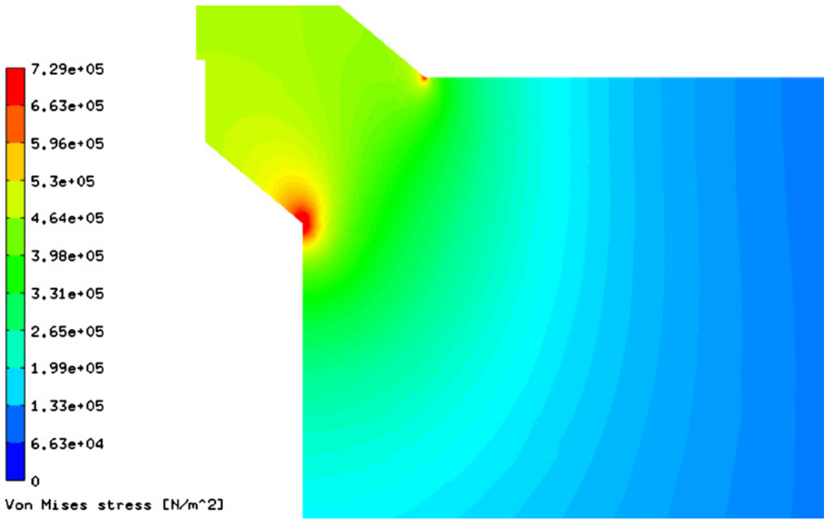

Fig. 15. Detail of Von Mises stress in the stiffening ring with singularities at the re-entrant corners.

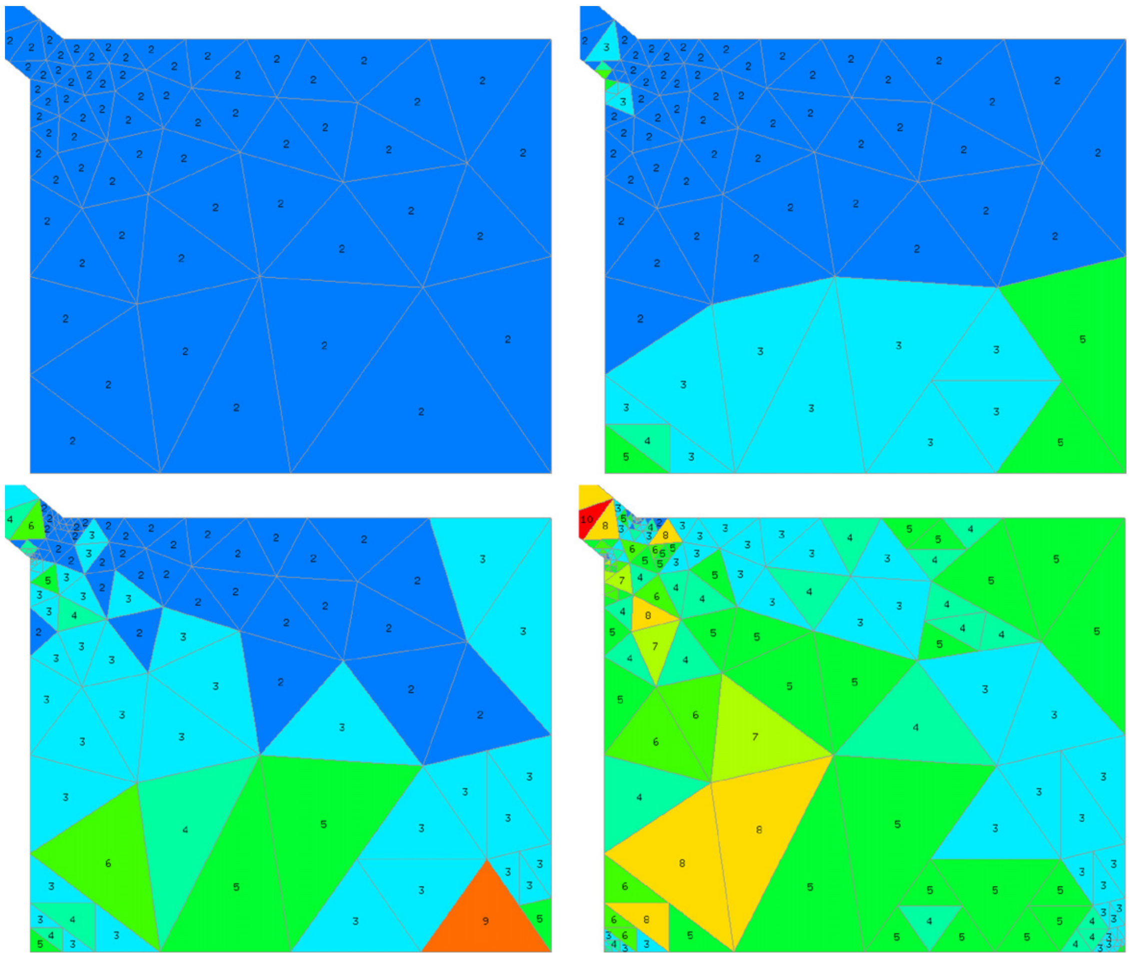

Fig. 16. Meshes for the *r* component of displacement, steps 1, 4, 6, 14.

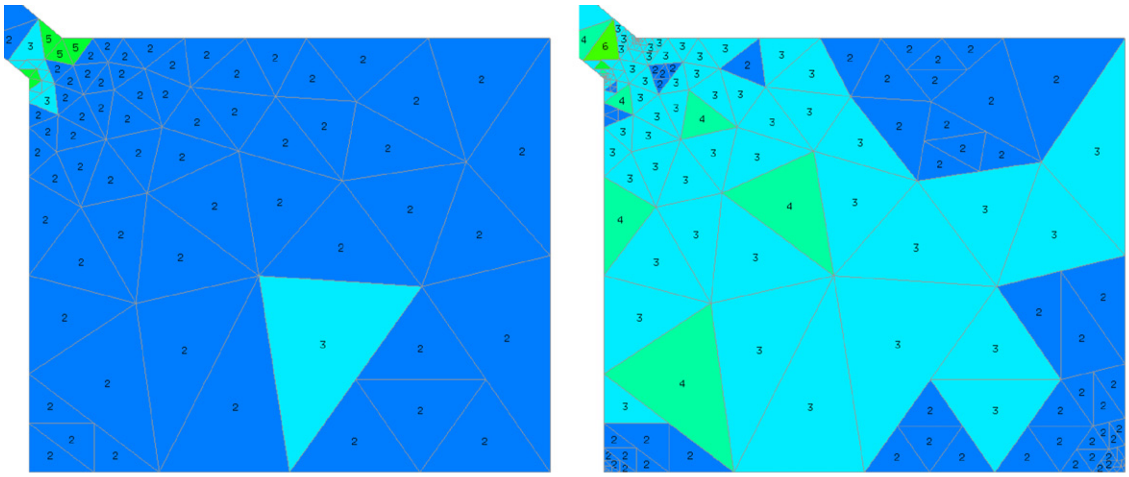

Fig. 17. Meshes for the z component of displacement, steps 4, 14.

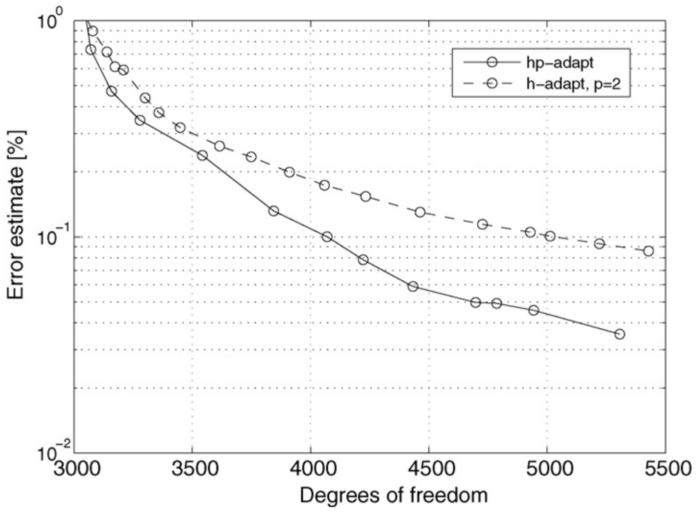

Fig. 18. Convergence history for the Girkmann problem.

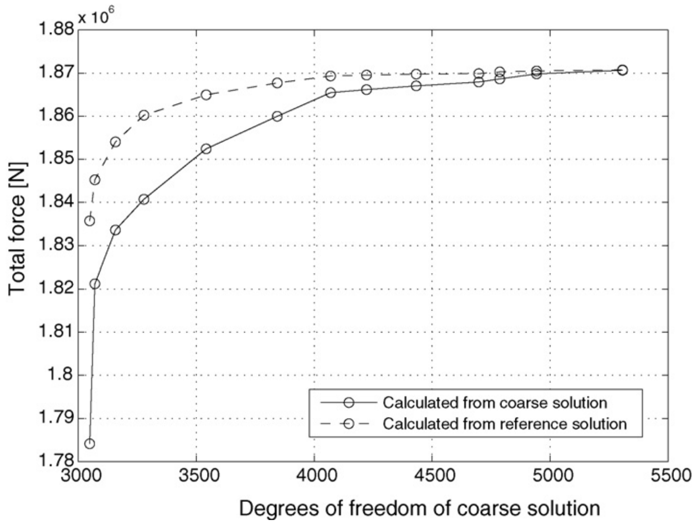

Fig. 19. Total force exerted by the dome on the stiffening ring (curves corresponding to the coarse mesh solution and reference solution).

6. Conclusion and outlook

The authors believe that adaptive higher-order finite element methods (*hp*-FEM) contain large computational potential which has not been fully exploited yet. Their exponential convergence remains somehow on the theoretical level where it was predicted, with very few people having practical experience with it. Largely, this is due to the difficulty of implementation of higher-order finite element discretizations and automatic *hp*-adaptivity algorithms.

This situation motivated the authors to simplify the algorithmic approach to automatic *hp*-adaptivity by splitting the algorithm into two parts—treatment of arbitrary-level constrained approximation (arbitrary-level hanging nodes) and 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 and which can slow down the convergence of the method significantly.

Among current research activities of the authors is the development of a modular software library which will facilitate the operation with higher-order finite elements and mesh adaptation procedures, and allow relatively simple application of the *hp*-FEM to various problems coming from engineering and scientific applications. More information about this effort can be found on the webpage http://hpfem.math.utep.edu.

Acknowledgments

The authors acknowledge the financial support of the Czech Science Foundation (Projects No. 102/05/0629 and 102/07/0496), and of the Grant Agency of the Academy of Sciences of the Czech Republic (Project No. IAA100760702).

References

[1] B. Aksoylu, S. Bond, M. Holst, An Odyssey into local refinement and multilevel preconditioning III: implementation and numerical experiments, SIAM J. Sci. Comput. 25 (2003) 478–498.

[2] I. Babuška, M. Griebel, J. Pitkaranta, The problem of selecting the shape functions for *p*-type elements, Int. J. Num. Methods Eng. 28 (1989) 1891–1908.

[3] 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. 49 (1986) 577–612.

[4] 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. 49 (1986) 613–657.

[5] 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. 49 (1986) 659–683.

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

[7] I. Babuška, B. Szabo, I.N. Katz, The *p*-version of the finite element method, SIAM J. Numer. Anal. 18 (1981) 515–545.

[8] I. Babuska, K. Izadpanah, B. Szabo, The postprocessing technique in the finite element method. The theory and experience, in: H. Kardestuncer (Ed.), Unification of Finite Element Methods, Elsevier Science Publishers B.V., North-Holland, 1984, pp. 97–121.

[9] A.C. Bauer, A.K. Patra, Performance of parallel preconditioners for adaptive *hp*-FEM discretizations of incompressible flows, Commun. Numer. Methods Eng. 18 (2002) 305–313.

[10] A.C. Bauer, A.K. Patra, Robust and efficient domain decomposition preconditioners for adaptive hp finite element approximation for linear elasticity with and without discontinuous coefficients, Int. J. Numer. Methods Eng. 59 (2004) 337–364.

[11] E. Bertolazzi, Discrete conservation and discrete maximum principle for elliptic PDEs, Math. Models Methods Appl. Sci. 8 (1998) 685–711.

[12] L. Demkowicz, J.T. Oden, W. Rachowicz, O. Hardy, Toward a universal *hp*-adaptive finite element strategy. Part 1: constrained approximation and data structure, Comput. Methods Appl. Math. Eng. 77 (1989) 79–112.

[13] L. Demkowicz, W. Rachowicz, P. Devloo, A fully automatic *hp*-adaptivity. TICAM Report No. 01–28, University of Texas at Austin, 2001.

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

[15] A. Laszloffy, J. Long, A.K. Patra, Simple data management, scheduling and solution strategies for managing the irregularities in parallel adaptive hp finite element simulations, Parallel Comput. 26 (2000) 1765–1788.

[16] J.M. Melenk, *hp*-Finite Element Methods for Singular Perturbations, Springer-Verlag, Berlin, 2002 (Lecture Notes in Mathematics 1796).

[17] D. Pardo, L. Demkowicz, Integration of hp-adaptivity and a two grid solver for elliptic problems, Comput. Methods Appl. Mech. Eng. 195 (7/8) (2006).

[18] M. Paszynski, J. Kurtz, L. Demkowicz, Parallel, fully automatic hp-adaptive 2D finite element package, TICAM Report 04-07, The University of Texas at Austin, 2004.

[19] A.G. Peano, Hierarchies of conforming finite elements for plane elasticity and plate bending, Comput. Math. Appl. 2 (1976) 211–224.

[20] W. Rachowicz, L. Demkowicz, An *hp*-adaptive finite element method for electromagnetics. Part II: A 3D implementation, Int. J. Numer. Methods Eng. 53 (2002) 147–180.

[21] C. Schwab, *p*- and *hp*-Finite Element Methods, Theory and Applications to Solid and Fluid Mechanics, Oxford University Press, New York, 1998.
[22] P. Šolín, Partial Differential Equations and the Finite Element Method, J. Wiley & Sons, 2005.
[23] P. Šolín, L. Demkowicz, Goal-oriented hp-adaptivity for elliptic problems, Comput. Methods Appl. Mech. Eng. 193 (2004) 449–468.
[24] P. Šolín, K. Segeth, I. Doležel, Higher-Order Finite Element Methods, Chapman & Hall/CRC Press, Boca Raton, 2003.
[25] P. Šolín, T. Vejchodský, On a weak discrete maximum principle for hp-FEM, J. Comput. Appl. Math. 209 (2007) 54–65.
[26] M. Zítka, P. Šolín, T. Vejchodský, F. Avila, Imposing orthogonality to hierarchic higher-order finite elements, Math. Comput. Simul. 76 (2007) 211–217.
[27] B. Szabo, I. Babuška, Finite Element Analysis, J. Wiley & Sons, New York, 1991, p. 368.
[28] T. Vejchodský, P. Šolín, Discrete maximum principle for higher-order finite elements in 1D, Math. Comput., November 2006, in press.
[29] T. Vejchodský, P. Šolín, M. Zítka, Modular hp-FEM system HERMES and its application to the Maxwell's equations, Math. Comput. Simul. 76 (2007) 223–228.