Journal of Scientific Computing, Vol. 17, Nos. 1–4, December 2002 (© 2002)

A Fully Automatic *hp*-Adaptivity

L. Demkowicz,[1] W. Rachowicz,[1] and Ph. Devloo[1]

Received September 21, 2001; accepted (in revised form) November 12, 2001

We present an algorithm, and a 2D implementation for a fully automatic *hp*-adaptive strategy for elliptic problems. Given a mesh, the next, optimally refined mesh, is determined by maximizing the rate of decrease of the *hp*-interpolation error for a reference solution. Numerical results confirm optimal, exponential convergence rates predicted by the theory of *hp* methods.

KEY WORDS: *hp* finite elements; *hp*-adaptivity.

1. INTRODUCTION

The goal of this research is to work out a *fully automatic hp*-adaptive strategy that would deliver meshes with a minimum number of degrees of freedom (d.o.f.) in the full range of error level, especially in the preasymptotic range.

This is a highly nonlinear and non-convex problem. Ideally, we would love to have a mesh optimality condition expressed in terms of local mesh size h and order of approximation p. After all, algorithms based on optimality conditions are among the most efficient methods in optimization. For *h*-adaptive methods, such optimality criteria are based on the notion of *mesh density functions*, and lead to the error equidistribution principle [5]. Examples of such efforts can be found recently, e.g., in [14, 17].

Most of commercial implementations[2] of various versions of *hp* finite element methods, see Introduction in [20], rely on *a-priori* information about corner and edge singularities [2, 20], or boundary layers, see [20, 13], and begin the solution process with a generation of judiciously constructed initial meshes like the geometrically graded meshes of Babuška, or Shiskin type meshes for handling the boundary layers. Once the mesh is known, uniform p refinements are made. In more sophisticated implementations, adaptive p-refinements are used. If the nature of the singularity is known a priori, these

[1] Texas Institute for Computational and Applied Mathematics, The University of Texas at Austin. E-mail: leszek@ticam.utexas.edu

[2] Except for PHLEX, see [12].

0885-7474/02/1200-0117/0 © 2002 Plenum Publishing Corporation

techniques are very effective and difficult to beat. In fact, optimal meshes may deliver exponential rates of convergence (in terms of number of d.o.f.) *without* the use of higher order discretizations, see e.g., [11].

The situation becomes less clear, if the regularity results are not at hand. An unoptimal initial mesh, followed by *p*-refinements, may lead to meshes that deliver results worse than than standard *h*-adaptive methods. Years ago we heard from Prof. Oleg Zienkiewicz that, for error levels 1–5% (measured in energy norm relative to the norm of the solution), the *h*-adaptive meshes of quadratic elements are the best, and there is little need for any extra sophistication. Indeed, for many practical problems, the quadratic elements offer probably the best balance between the complexity of coding (one d.o.f. per node, no need for orientation of nodes) and the results they can deliver. We will return to this issue in the subsequent sections of this work. Attempts to correct an inefficient initial mesh through *h* refinements, have led to methods that perform *h* and *p*-refinements interchangeably, see e.g., the *Texas Three Step* strategy [15, 16, 6]. The resulting meshes are easy to obtain but, in general, do not lead to optimal results.

A genuine *hp* strategy was proposed in [1], where the choice between the *h* and *p*-refinement is based on monitoring local *h*-convergence rates. The methodology has recently been applied to solve hyperbolic problems in [10], in context of *discontinuous hp* discretizations.

In the presented work, we follow closely the original idea presented in [18], where the optimal mesh is obtained by minimizing a local projection error for a reference solution. The optimality of 1D results from [18], has recently been independently verified in [3] and [19]. The big question is, how to proceed for 2D and 3D meshes? In [18], following the original method of Babuška for *p* method error estimation, optimal meshes were constructed by minimizing local energy projection errors. The major problem with the technique is that, optimal refinements of neighboring elements are very often in conflict with each other. If the minimum rule[3] is used, and if from two neighboring elements one wants to be *h*- and the other *p*-refined, the common edge is not refined at all! This implies that the mesh optimization problem may be more complicated than just the choice between the *h*- or *p*-refinements.

The method presented in this paper derives from the recent progress in the understanding of the idea of *hp*-interpolation [8]. We will begin our presentation with the 1D case first, and then spend most of our time (space) presenting the 2D version of the method and illustrate it with numerical examples. At the end we shall attempt some conclusions and discuss possible generalizations.

[3] Order for an edge is set the minimum of the orders for the adjacent elements. Among other things, this is essential for the commutativity of the de Rham diagram [7].

2. THE *hp* MESH OPTIMIZATION IN 1D

We begin the presentation of our algorithm with the simplest class of one-dimensional elliptic problems. Given an interval $(0, l)$, the problem of interest is stated in the usual variational form as follows:

$$\begin{cases} u \in \hat{u}_D + V \\ b(u, v) = l(v) \quad \forall v \in V \end{cases} \tag{2.1}$$

Here $V \subset H^1(0, l)$ is the *space of test functions*, $\hat{u}_D \in H^1(0, l)$ denotes a *lift* of Dirichlet boundary condition data u_D, and $b(u, v)$ and $l(v)$ denote bilinear (sesquilinear), and linear (antilinear) forms corresponding to a particular boundary-value problem. For instance, for a boundary-value problem,

$$\begin{cases} -(a(x)\, u')' + b(x)\, u' + c(x)\, u = f(x) & x \in (0, l) \\ u = u_D & x = 0 \\ au' + \beta u = g & x = l \end{cases} \tag{2.2}$$

the space

$$V = \{v \in H^1(0, l) : v(0) = 0\} \tag{2.3}$$

and the forms are defined as follows,

$$\begin{aligned} b(u, v) &= \int_0^l \{au'v' + bu'v + cuv\}\, dx + \beta u(l)\, v(l) \\ l(v) &= \int_0^1 fv\, dx + gv(l) \end{aligned} \tag{2.4}$$

Given an *hp* finite element mesh, and the corresponding finite element space $V_{hp} \subset V$, the approximate solution is determined using the standard Galerkin procedure,

$$\begin{cases} u_{hp} \in \hat{u}_D + V_{hp} \\ b(u_{hp}, v_{hp}) = l(v_{hp}) \quad \forall v_{hp} \in V_{hp} \end{cases} \tag{2.5}$$

We shall assume that data $a(x)$, $b(x)$, $c(x)$, $f(x)$, β satisfy the usual regularity assumptions to guarantee the optimality of the Galerkin approximation, i.e., that there exists a constant $C > 0$ such that the actual approximation error is bounded by the best approximation error,

$$\|u - u_{hp}\| \leqslant C \inf_{v_{hp} \in V} \|u - (\hat{u}_D + v_{hp})\| \tag{2.6}$$

We proceed now with the definition of the *hp* interpolation. Given an interval $(a, b) \in (0, l)$, and $w \in H^1(a, b)$, we define its *(generalized) hp interpolant* $w_{hp} = \Pi w$ by setting up a constrained minimization problem,

$$\begin{cases} w_{hp} \in X_{hp}(a, b) \\ w_{hp} = w \qquad \text{at} \quad x = a, b \\ \int_a^b (w'_{hp} - w')^2 \, dx \to \min \end{cases} \tag{2.7}$$

Here $X_{hp}(a, b)$ denotes the finite element space corresponding to a finite element mesh covering interval (a, b). If the intervals are selected to coincide with finite elements, the notion reduces to the standard *hp*-interpolation [7, 8].

The constrained minimization problem is equivalent to the FE approximation of a local Dirichlet problem for a 1D Laplace problem defined over interval (a, b).

Minimization of the Interpolation Error. Suppose now that we are given some function $w \in H^1(0, l)$, and an existing *hp* FE mesh. We can determine an optimal *hp* refinement of the mesh by minimizing the corresponding *hp*-interpolation error or, more precisely, by maximizing the rate with which the interpolation error will decrease [18].

As the interpolation is done *locally*, for each of the elements of the initial mesh, we can proceed in two steps.

Step 1 (Local): Determine an Optimal Refinement for Each Element. We choose between the *p*-refinement and a sequence of *competitive h*-refinements that result in the same increase of the number of d.o.f. More specifically, we assume that the order p may be only increased by one. This implies that orders p_1, p_2 for the *h*-refined element sons must satisfy condition:

$$p_1 + p_2 - 1 = p \tag{2.8}$$

Projection problem (2.7) is then solved using the original element with order of approximation p raised to $p+1$, and p different FE spaces corresponding to the h refined element with the element sons of order p_1 and p_2. Thus, for a linear element, the choice is between a quadratic element and two linear elements; for a quadratic element, the choice is between a cubic element and two combinations: linear/quadratic or quadratic/linear element, and so on.

Step 2 (Global): Determine Which Elements to Refine. Once we have determined the optimal refinement for each of the elements, we can compute the corresponding decrease of the global interpolation error (squared),

$$\sum_K \left(\| (w - w_{hp}^{\text{old}})' \|_K^2 - \| (w - w_{hp}^{\text{new}})' \|_K^2 \right) \tag{2.9}$$

Here the summation extends over all elements K in the original mesh to be refined, w_{hp}^{old} and w_{hp}^{new} denote the *old* and the *new, optimal* interpolants, and the norm is the $L^2(K)$ norm. Denoting the element contributions in the formula above as η_K, we follow the optimality criterion derived in [18], and refine only those elements for which

$$\eta_K \geqslant \tfrac{1}{3} \max_K \eta_K \tag{2.10}$$

As each element is given only one d.o.f., η_K have the interpretation of the *rates* with which the element errors decrease. We *invest* only in those elements that bring the best "interest rates." As we can invest only one d.o.f. per element, the question is only whether to refine or not.

In other words, the goal of the mesh optimization is to minimize the interpolation error, and we go after a refinement that produces the greatest decrease of the cost functional. This can be interpreted as a discrete version of the steepest descent method, and element contributions in (2.9) are just discrete equivalents of the components of the cost functional gradient,

Factor $1/3$ above is somehow arbitrary and it reflects an integer version of the method with components of the gradient having been scaled to change between 0 and 1.5. Rounding to the integer values 0, 1, we refine all elements for which the gradient component is between 0.5 and 1.5. Hence the $1/3$ factor…

How to Choose the Reference Function? Obviously, the reference function has to be constructed using the FE solution and possibly the data to the problem. In [18], for the reference solution, a highly accurate postprocessed solution, obtained using the Babuška's extraction formulas was used. In the present implementation, we employ for the reference solution, FE solution $u_{h/2,\,p+1}$ corresponding to a *globally refined* mesh, *both in h and p*. The idea reflects the fact that the reference solution must resolve the local scales sufficiently well to allow for making the choice between h and p. In simpler words, we have to try first both h and p, before we can dismiss one of the two refinements. Anticipating a soaring critique of our choice (another academic method…), we defend ourselves up front with two arguments.

- Solution $u_{h/2,\,p+1}$ can (and must) be obtained using a multigrid, or at least a two-grid solver. The uniformity of the global refinement allows for an easy and efficient implementation of the two-grid solver.
- We do not intend to discard the *fine mesh solution*. On the contrary, this will be our final solution that we intend to deliver! This implies that with our optimal *coarse* mesh, we will shoot for errors essentially larger (5–10%) than the celebrated 1% error level.

The *hp* Algorithm. The final algorithm described below includes a simple error estimator where the error corresponding to the coarse mesh is

estimated simply by *computing* the norm of the difference between the fine and the coarse mesh solutions.

begin with an initial coarse mesh
do until exhausted
  solve the problem on the current mesh
  save data about the current mesh and dumpout the corresponding
    content of the data structure arrays
  refine globally the mesh, first in h, then in p
  solve the problem on the refined mesh, determining new solution $u_{h/2,\,p+1}$
  determine the (energy) norm of the difference between the two solutions
  if the error is acceptable then STOP
  use $u_{h/2,\,p+1}$ in place of the reference solution to determine optimal
    refinements of the current mesh
  dumpin the original mesh data and perform the optimal refinements
enddo

Example 1: A Solution with an Internal Layer. We illustrate the algorithm with the standard model problem,

$$\begin{cases} u(0) = \gamma_0, \quad u(1) = \gamma_1 \\ -u'' = f \end{cases}$$

where load $f(x)$ and Dirichlet data γ_0, γ_1 correspond to the exact solution with an internal layer,

$$u_{\text{exact}}(x) = \text{atan}(60(x - \pi/3)) \tag{2.11}$$

shifted to the left so that the symmetry is lost. The solution, together with the corresponding FE solution on an initial mesh of two linear elements is shown in Fig. 1.

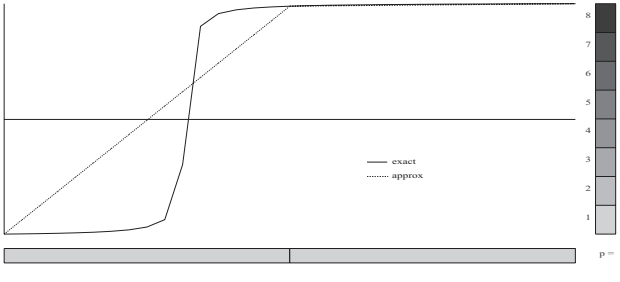

Fig. 1. Example 1: Exact solution and the corresponding FE approximation on the initial mesh of two linear elements.

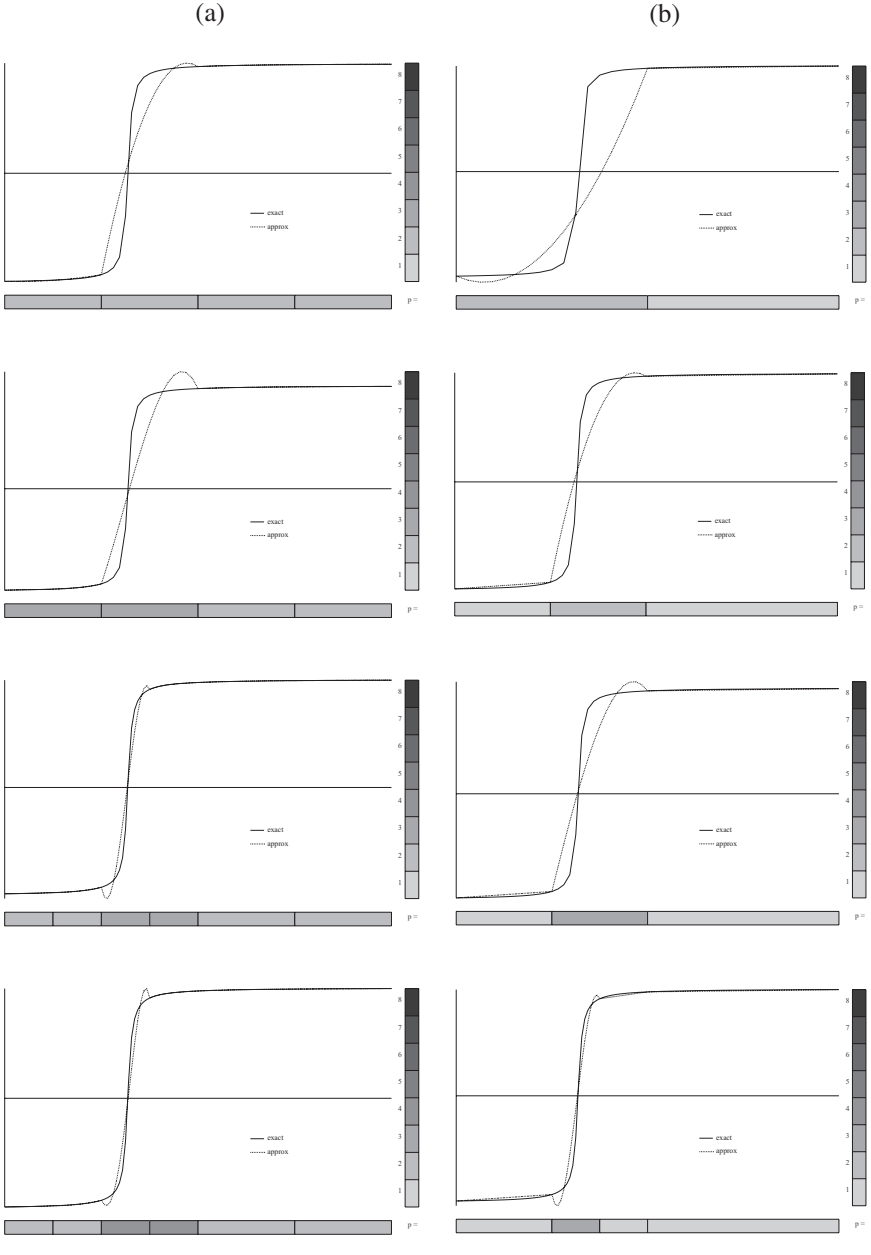

Fig. 2. Example 1: Stages 1–4 of the optimal *hp* mesh selection. (a) Reference solutions; (b) optimal *hp* interpolations.

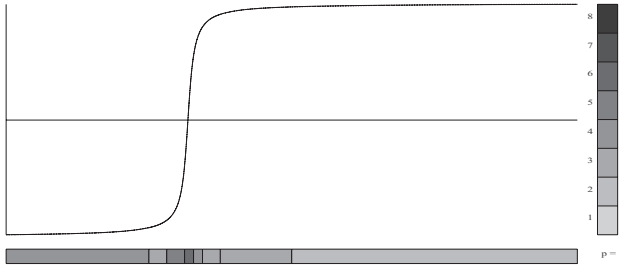

Fig. 3. Example 1: Final, optimal *hp* mesh.

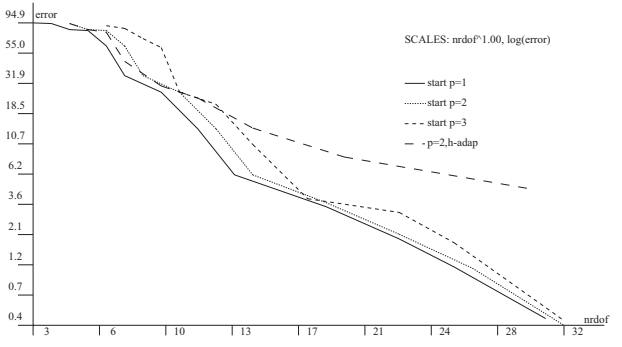

Fig. 4. Example 1: Convergence rates on the linear–log scale.

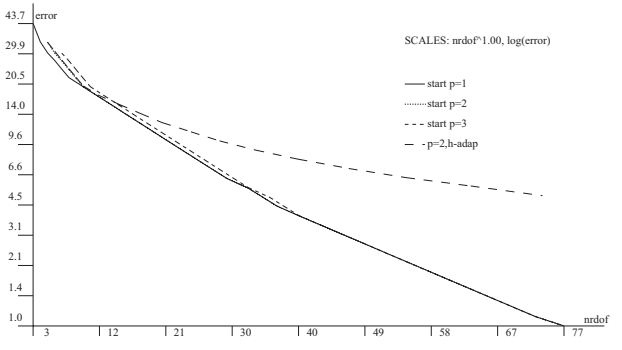

Fig. 5. Example 2: Rates of convergence.

We impose the error tolerance $tol = 1$ per-cent of the energy norm of the solution. Figures 2 show first 4 consecutive meshes, with the $h/2$, $p+1$ solution on the left, and optimal meshes on the right. The final mesh with the corresponding solution overlapping exactly the exact solution, is shown in Fig. 3.

The corresponding convergence history on the linear–log scale for three different initial meshes of two linear, quadratic and cubic elements, is presented in Fig. 4. The final meshes for all three cases are very similar and in all three cases we observe better than exponential convergence as indicated by the superlinear behavior on the linear–log scale.

For comparison, we present in Fig. 4 also the convergence history for just *h*-adaptivity (we simply force the algorithm to choose always the *h*-refinement) and quadratic elements. For the 1D example, the *hp*-adaptivity beats the *h*-adaptivity by one order of magnitude…

Example 2: A Singular Solution. $u_{\text{exact}}(x) = x^{0.6}$. Error tolerance tol = 1% of the energy norm of the solution.

Again, we start with an initial mesh of two linear, quadratic or cubic elements. In all three cases we end up with the same, geometrically refined mesh. The corresponding, exponential convergence rates are represented on the linear–log scale in Fig. 5, and compared again with *h*-adaptivity for quadratic elements. The *hp* adaptivity again significantly outperforms the *h*-adaptivity.

3. THE 2D ALGORITHM

Again, we focus on the standard two-dimensional elliptic problems. Given a domain Ω, we define the bilinear and linear forms as follows,

$$b(u,v) = \int_\Omega \left\{ \sum_{i,j=1}^{2} a_{ij}(\mathbf{x}) \frac{\partial u}{\partial x_i} \frac{\partial v}{\partial x_j} + \sum_{i=1}^{2} b_i(\mathbf{x}) \frac{\partial u}{\partial x_i} v + c(\mathbf{x})\, uv \right\} d\mathbf{x}$$

$$+ \int_{\Gamma_3} \beta(\mathbf{x})\, uv\, ds \tag{3.12}$$

$$l(v) = \int_\Omega f(\mathbf{x})\, v\, d\mathbf{x} + \int_{\Gamma_2 \cup \Gamma_3} g(\mathbf{x})\, v\, ds$$

with the corresponding boundary-value problem,

$$\begin{cases} -\sum_{j=1}^{2} \frac{\partial}{\partial x_j} \left(\sum_{i=1}^{2} a_{ij}(\mathbf{x}) \frac{\partial u}{\partial x_i} \right) + \sum_{i=1}^{2} b_i(\mathbf{x}) \frac{\partial u}{\partial x_i} + c(\mathbf{x})\, u = f(\mathbf{x}) & \text{in } \Omega \\ u = u_D & \text{on } \Gamma_1 \\ \sum_{i=1}^{2} a_{ij}(\mathbf{x}) \frac{\partial u}{\partial x_i} = g(\mathbf{x}) & \text{on } \Gamma_2 \\ \sum_{i=1}^{2} a_{ij}(\mathbf{x}) \frac{\partial u}{\partial x_i} + \beta(\mathbf{x})\, u = g(\mathbf{x}) & \text{on } \Gamma_3 \end{cases} \tag{3.13}$$

where $\Gamma_1, \Gamma_2, \Gamma_3$ denote the Dirichlet, Neumann, and Cauchy parts of the boundary.

We shall assume that domain Ω can be covered exactly with (possibly curvilinear, isoparametric) finite elements or, equivalently speaking, we shall neglect the error due to the approximation of the geometry.

We define now the two-dimensional version of the (generalized) *hp*-interpolation. Given a function w defined on an *hp* finite element mesh, we proceed in three steps.

Vertex (Linear) Interpolant. We construct the standard linear Lagrange interpolant w^1_{hp} of function w using its values at vertices $\mathbf{a}$,

$$w^1_{hp}(\mathbf{a}) = w(\mathbf{a}) \tag{3.14}$$

Edge Interpolant. For each edge e in the mesh, we project the difference $w - w^1_{hp}$ onto the *space of edge shape functions* $V_{hp}(e)$, vanishing at the edge endpoints,

$$\begin{aligned} &w^e_{hp} \in V_{hp}(e) \\ &\|w - w^1_{hp} - w^e_{hp}\|_{H^{1/2}_{00}(e)} \to \min \end{aligned} \tag{3.15}$$

This is equivalent to solving a 1D system of equations,

$$\begin{cases} w^e_{hp} \in V_{hp}(e) \\ (w^e_{hp}, v_{hp})_{H^{1/2}_{00}(e)} = ((w - w^1_{hp}), v_{hp})_{H^{1/2}_{00}(e)} \quad \forall v_{hp} \in V_{hp}(e) \end{cases} \tag{3.16}$$

The global *edge interpolant* w^2_{hp} is then constructed as the sum of finite element lifts of the edge interpolants w^e_{hp},

$$w^2_{hp} = \sum_e \hat{w}^e_{hp} \tag{3.17}$$

Interior Interpolant. For each element interior K, we project the difference $w - w^1_{hp} - w^2_{hp}$ onto the *space of element bubble functions* $V_{hp}(K)$,

$$\begin{aligned} &w^3_{hp} \in V_{hp}(K) \\ &\|w - w^1_{hp} - w^2_{hp} - w^3_{hp}\|_{H^1_0(K)} \to \min \end{aligned} \tag{3.18}$$

This is again equivalent to solving a system of equations,

$$\begin{cases} w^3_{hp} \in V_{hp}(K) \\ \displaystyle\int_K \nabla w^3_{hp} \, \nabla v_{hp} \, d\mathbf{x} = \int_K \nabla(w - w^1_{hp} - w^2_{hp}) \, \nabla v_{hp} \, d\mathbf{x} \quad \forall v_{hp} \in V_{hp}(K) \end{cases} \tag{3.19}$$

The final *hp* interpolant is now defined as the sum of the vertex, edge and element interiors interpolants,

$$w_{hp} = w_{hp}^1 + w_{hp}^2 + w_{hp}^3 \tag{3.20}$$

If K is an element sharing (whole) edge e, the $H_{00}^{1/2}(e)$ inner product and the corresponding norm can be understood in the following sense,

$$(u, v)_{H_{00}^{1/2}(e)} = \int_K \nabla\tilde{u}\, \nabla\tilde{v}\, d\mathbf{x}, \qquad \|u\|^2_{H_{00}^{1/2}(e)} = (u, u)_{H_{00}^{1/2}(e)} \tag{3.21}$$

where $\tilde{u}$ denotes the harmonic extension of function u defined on edge e, extended by zero to the rest of the boundary of element K. It has been shown in [8] that the *hp* interpolation operator $\Pi: w \to w_{hp}$ is well defined and continuous on space $H^{1+\epsilon}(\Omega)$,

$$\|\Pi w\|_{H^1} \leqslant C\, \|w\|_{H^{1+\epsilon}} \tag{3.22}$$

with constant C *independent* of both element size h, and order of approximation p. As the interpolation preserves functions from finite element space X_{hp}, this leads to the optimality (up to the ϵ) of the interpolation error estimate,

$$\|w - \Pi w\|_{H^1} \leqslant C \inf_{w_{hp} \in X_{hp}} \|w - w_{hp}\|_{H^{1+\epsilon}} \tag{3.23}$$

The result motivates our mesh refinement strategy based again on the idea of minimizing the interpolation error for a reference solution.

The $H_{00}^{1/2}$ Norm. It has been shown in [8], that any function $w \in H^{\frac{1}{2}+\epsilon}$ that vanishes at the edge endpoints, belongs to space $H_{00}^{1/2}(e)$. This justifies the edge interpolation procedure for edges whose both endpoint vertices are *unconstrained*. For an edge with a constrained vertex ("hanging node"), however, the corresponding vertex node value comes from the interpolation along the *constraining, "big" edge*, and may not match the value of the function being interpolated. Consequently, the edge projection problem is ill-posed. In the practical implementation we commit much bigger crime, replacing the $H_{00}^{1/2}$ norm with a weighted H_0^1 norm,

$$\|u\|_e^2 = \int_e \left(\frac{\partial u}{\partial s}\right)^2 \left(\frac{\partial s}{\partial \xi}\right)^{-1} ds = \int_0^1 \left(\frac{\partial u}{\partial \xi}\right)^2 d\xi \tag{3.24}$$

Here $\mathbf{x} = \mathbf{x}(\xi)$, $\xi \in (0, 1)$ is the FE parametrization for the edge, and

$$\frac{\partial s}{\partial \xi} = \sqrt{\sum_{i=1}^{2} \left(\frac{\partial x_i}{\partial \xi}\right)^2} \tag{3.25}$$

Note that this weighted H_0^1 norm scales with the length of the edge the same way as the $H_{00}^{1/2}$ norm. As for the reference solution we employ again the finite element solution $u_{h/2,\,p+1}$ corresponding to the globally refined mesh, the edge projection (at least for affine elements) takes place only for piecewise polynomials, and the choice between the two norms results simply in slightly different metrics. We have actually implemented both norms and studied the corresponding edge projection problems. We do not present a detailed evidence here, but for order $p < 10$, the difference between the two projections is insignificant, and the obtained finite element meshes are identical. The difference between the two norms, however, grows with p and, for higher order, spectral-type implementations, the choice of the norm will become essential. We also emphasize that the presence of the weight is essential. Without it, the edge part of the algorithm presented next, fails and delivers nonoptimal meshes.

Minimization of the Interpolation Error in 2D. We are ready now to discuss steps of the 2D mesh optimization algorithm. As the 2D implementation is much more technical than in 1D, we shall present one step at a time, and illustrate them with an example of an actual *hp* mesh. The illustrations will correspond to a 2D version of the problem with an internal layer presented in the previous section. We select for domain Ω a unit square, and set up the exact solution as

$$u_{\text{exact}}(\mathbf{x}) = \operatorname{atan}(60r) \tag{3.26}$$

where r is a polar coordinate with origin at $(1.25; -0.25)$. We shall solve simply the Laplace equation with Dirichlet boundary conditions imposed at edges $x = 0$, $y = 1$, and Neumann boundary conditions on the rest of the boundary.

Step 1: Preliminaries. Suppose we start with a mesh shown in Fig. 6. We begin by saving necessary data on the mesh. This includes the list of elements in the mesh, and the corresponding orders of approximation, and the list of mid-edge nodes. We solve then the problem on the mesh and save the solution in the data structure arrays. If this is a model problem for which the exact solution is known, we compute the global error and output it, together with the number of d.o.f., for a further graphical postprocessing. The mesh shown in Fig. 6 has only 160 d.o.f. and delivers a solution with 37.4% relative error, measured in H_0^1-norm, with respect to the H_0^1-norm of the exact solution.

Step 2: Global hp-Refinement, Error Estimation. Determining Element Isotropy Flags. We perform a global, *isotropic hp*-refinement, breaking each element into four sons, and then raising the order of approximation for each node by one. New d.o.f. are initiated in such a way that the refined mesh still supports the FE solution on the initial, coarse mesh. We save the existing d.o.f. corresponding to the coarse mesh solution, and solve the problem on the fine mesh. We estimate the error for the coarse

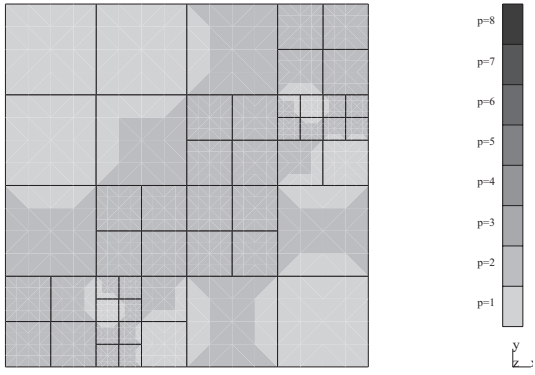

Fig. 6. The 2D *hp* algorithm: an initial, coarse *hp* mesh.

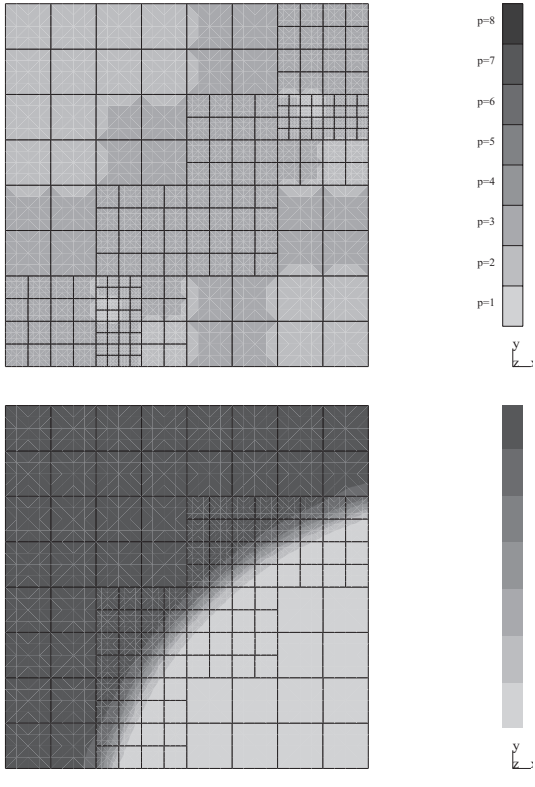

Fig. 7. The 2D *hp* algorithm: the fine mesh with the corresponding FE solution.

mesh by computing the norm of the difference between the coarse and fine mesh solutions. The fine mesh, together with the corresponding FE solution, is shown in Fig. 7. The efficiency index for the error estimate, defined as the difference of the actual error and the estimated error, relative to the exact error, for the presented case equals 0.974. Finally, we save the fine mesh solution in an element fashion. During the error computation, we determine the *element isotropy (h-refinement) flags*. This is done in the following way. Let $e(\mathbf{x}) = \hat{e}(\xi)$ denote the error function, i.e., the difference between the coarse and fine mesh solutions. The H_0^1 element error, expressed in master element coordinates ξ looks as follows,

$$\int_0^1 \int_0^1 \left(g_{11} \left(\frac{\partial e}{\partial \xi_1} \right)^2 + g_{22} \left(\frac{\partial e}{\partial \xi_2} \right)^2 + 2 g_{12} \frac{\partial e}{\partial \xi_1} \frac{\partial e}{\partial \xi_2} \right) J \, d\xi_1 \, d\xi_2 \qquad (3.27)$$

where

$$g_{ij} = \sum_{k=1}^{2} \frac{\partial x_k}{\partial \xi_i} \frac{\partial x_k}{\partial \xi_j}, \qquad J = \det \left(\frac{\partial x_i}{\partial \xi_k} \right)$$

The anisotropy flag is set to 1, which indicates that the element should be refined across the ξ_1 axis only, if two conditions are met:

- the contribution corresponding to the off-diagonal term g_{12} in the element metric is less (in absolute value) than 1% of the sum of the main diagonal terms;
- the contribution corresponding to g_{22} term is less than 1% of the one corresponding to g_{11}.

Analogously, we identify the other anisotropic case with flag 2, otherwise the element is declared as a candidate for an isotropic refinement only. Clearly, the criterion favors the isotropic refinements and anticipates the need for anisotropic refinements only in such extreme cases as one-dimensional test cases, or solutions with one dimensional boundary layers aligned with the mesh.

Step 3: Determining Optimal Edge Refinements. We loop through the edges in the initial mesh, and use the fine mesh solution to determine the optimal refinement for each edge in the mesh. We use the weighted H_0^1 norm to choose between the *p*-refinement and a competitive *h*-refinement, using the 1D algorithm. Two possibilities occur:

- The *p*-refinement wins. We save the d.o.f. for the edge interpolant, i.e., projection of the difference of the fine mesh solution and its linear interpolant, onto the space of edge shape functions corresponding to the *p*-refinement. We save the corresponding decrease of the edge interpolation error. We do not discard the most competitive *h*-refined edge projection. Instead, we use the corresponding two

elements mesh as a starting point to determine new (non-competitive, greater) orders of approximation for the new mid-edge nodes in such a way, that the error of the corresponding projection matches that for the *p*-refined element. We save the corresponding d.o.f. for the *h*-refined element. We label the edge with flag = 1.

- The *h*-refinement wins. We save the d.o.f. for the edge interpolant corresponding to the *h*-refined edge, the corresponding optimal, orders of approximation for the new mid-edge nodes, and the decrease of the edge interpolation error. We label the edge with flag = −1.

In both cases we also save d.o.f. corresponding to the projection on the coarse mesh shape functions. This is for initiating the d.o.f. for edges that will not be refined.

Finding the optimal distribution of orders of approximation p for the refined edge falls into a general class of finding optimal *p*-refinements for a given mesh. We follow here the standard simple strategy by determining element contributions to the global error, and increasing the order of approximation in all elements (in this case, maximum two only…) whose contributions are greater than a prespecified percentage[4] of the maximum contribution. Then we try out the new mesh, compute the corresponding projection error, and continue until we reach a limiting, prespecified minimal error decrease. Obviously, the orders for the new mid-edge nodes do not exceed that of the *p*-refined edge, and due to the nested spaces, we are guaranteed to match the error for the *p*-refined edge.

And a final remark on the edges with constrained endpoint vertex nodes. Yes, we "cheat" here, as the corresponding vertex values result from the constrained approximation, and do not match those of the projected function (reference solution). For straight edges (constant weight) linear functions are orthogonal to edge shape functions in H_0^1 product, and the "cheating" does not affect the corresponding projections, but it *does change* the corresponding error decrease rate.

Step 4: Global Edge Minimization Problem. Following the 1D strategy, we determine the maximum error decrease rate, and identify all edges with error decrease equal at least to one third of the maximum one. These are the edges that we intend to refine. We label all remaining edges with flag = 0.

Step 5: Optimal h Refinement of Elements. We use the information about the optimal refinement for edges to construct an optimally *h*-refined mesh. An element is broken into four (isotropic refinement) or two (anisotropic

[4] 70% for all presented numerical results.

Table I. The 2D *hp* Algorithm: Ultimate Edge Refinement Flags

label	desired refinement	actual refinement
−1	*h*-refinement	*h*-refinement
1	*p*-refinement	*p*-refinement
−11	*p*-refinement	*h*-refinement
0	no refinement	no refinement
−10	no refinement	*h* refinement

refinement) element sons, if at least one of its edges has flag $=-1$, i.e., wants to be *h*-refined. In this way we give a priority to *h* refinements over *p*-refinements. The choice between the isotropic and anisotropic refinement is based on the isotropy flags determined earlier. At this point, all edges labeled with -1 got broken but, due to the mesh regularity assumptions,[5] some edges with labels 1 or 0 got broken as well. More precisely, we end up with five kinds of edges, labeled as shown in Table I.

Step 6: Initiating d.o.f. for Edges with Label=−10. At this point, we initiate the d.o.f. for all refined edges with the values saved earlier. The only case for which we have not got values at hand is the case of edges that were not supposed to be refined in any way but that got *h*-refined anyway. We revisit them one more time and determine the minimal orders of approximation for the new mid-edge nodes by requesting that the corresponding projection error matches the error corresponding to the coarse mesh within a prescribed tolerance *tol*. The tolerance is determined by taking 1% of the anticipated drop in the global error and dividing it by the number of the label 10 edges,

$$\text{tol} = 0.01 \frac{\sum_{e,\,\text{label}=-1,\,1,\,-11} \text{edge error decrease}}{\text{number of edges with label 10}} \tag{3.28}$$

The 1% is again completely arbitrary and could be replaced with more or less stringent parameter. The criterion reflects our conviction that, for edges contributing with small error, it does not make sense to match exactly the existing error and we can allow for a slight increase of the local error as long as the global error is under control. Figure 8 presents the mesh obtained after the optimal refinement of edges. Note that the orders of approximation for element middle nodes at this point are incidental (inherited from the coarse mesh).

Step 7: Determining Optimal Orders of Approximation for the Refined Element Interiors. At this point, the topology of the mesh has been determined, and we need only to determine new, optimal orders for the element interiors. This is done locally, one (coarse mesh) element at the

[5] We use the standard one-irregular meshes strategy.

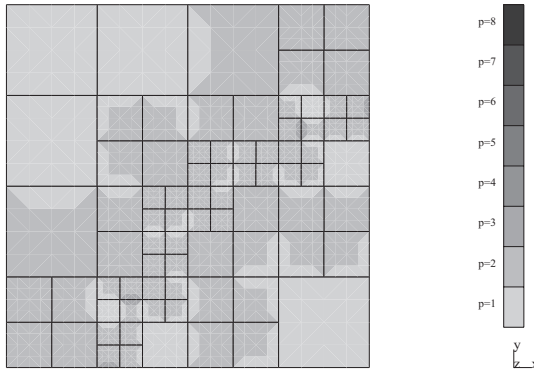

Fig. 8. The 2D *hp* algorithm: mesh after edge refinements.

time. In essence, we are solving a Laplace equation (H_0^1-projection problem) with Dirichlet boundary conditions,

$$\int_K (\nabla(u_{h/2,\,p+1} - \hat{u} - u_{\text{opt}}))^2 \, d\mathbf{x} \to \min \tag{3.29}$$

where $\hat{u}$ is a lift of already known interpolant along the optimally refined edges, and u_{opt} denotes the unknown, optimal p FE solution to the problem. We solve this p-adaptivity problem the same way as the 1D edge problem discussed above. A few details:

1. There are three distinct cases illustrated in Fig. 9 below, a p(only)-refined element, an element refined anisotropically, in either horizontal or vertical direction, or an element refined isotropically, into four element sons. The corresponding, unknown orders of approximation are shown in the figure. We assume that the orders for edges are determined using the minimum rule.
2. We begin the optimization process with values that are implied by already known orders on the element boundary and the min rule. If the element is declared to be isotropic, we begin with an isotropic (if possible), initial order of approximation.
3. Having solved the problem for given p's, we compute element contributions to the error, and increase the order of approximation in all (one, two, or four) subelements that contribute within 70% of the max error. We allow for the order to grow anisotropically, according to the directional contributions to the subelement error.
4. We monitor the error decrease rate,

$$\Delta = \frac{\|u_{h/2,\,p+1} - \hat{u} - u_{\text{old}}\|^2 - \|u_{h/2,\,p+1} - \hat{u} - u_{\text{new}}\|^2}{\text{ndof}_{\text{new}} - \text{ndof}_{\text{old}}} \tag{3.30}$$

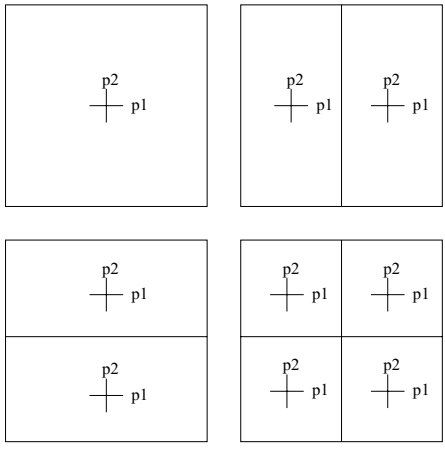

Fig. 9. The 2D *hp* algorithm: Step 7—determining optimal orders of approximation for refined elements.

where $u_{\text{old}}, u_{\text{new}}$ stand for the previous and current solution to projection problem (29), and ndof_{old}, ndof_{new} denote the corresponding number of d.o.f. *interior* to the refined element.

5. The order of approximation cannot exceed that of the fine mesh, supporting solution $u_{h/2,\, p+1}$.
6. The local optimization problem is stopped when two conditions are satisfied:
   - The projection error is smaller that the projection error corresponding to the unrefined element and the current order of approximation.
   - The error decrease rate is less or equal than $1/3$ of a prespecified rate Δ_0.

   In order to be consistent with the edge part of the algorithm, Δ_0 should be equal to the maximum rate for all refined elements in the mesh. This would be excessively expensive, and we select for Δ_0 the rate corresponding to the element for which edge error decrease rates were maximal. This is possible, since the edge minimization problems were solved for *all* edges. Once we know the element, we run the algorithm for determining the optimal order p, and record the corresponding maximal error decrease rate Δ_0.

Step 8: Enforcing the min Rule. It may happen that, once the element orders have been determined, the edge orders do not conform anymore to the min rule. We increase then the edge order, setting it to the minimum of the orders for the adjacent elements. In the end we obtain the next optimally refined *hp* mesh illustrated in Fig. 10.

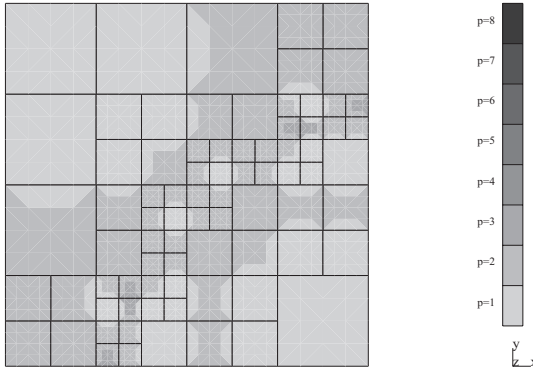

Fig. 10. The 2D *hp* algorithm: next optimal mesh.

Final *hp* algorithm is the same as in 1D. Starting with an initial mesh, we continue the iterative process until we meet an error criterion. This usually means that the coarse mesh should deliver an error within the range of 5 to 10%, as we have at our disposal the fine mesh solution (we stop after the coarse error estimation, i.e., after Step 2).

Example 3: Problem with the Internal Layer. We start with the example that we have used to illustrate the *hp* algorithm with. Figure 11 presents the convergence history for three starting initial, uniform meshes of only four elements, with $p = 1, 2, 3$. The rates are presented on the linear–log scale.

The results are compared with a simple *h*-adaptive method for a mesh of quadratic elements, and a strategy in which we refine all elements that contribute with errors within 70% of the max contribution. More precisely,

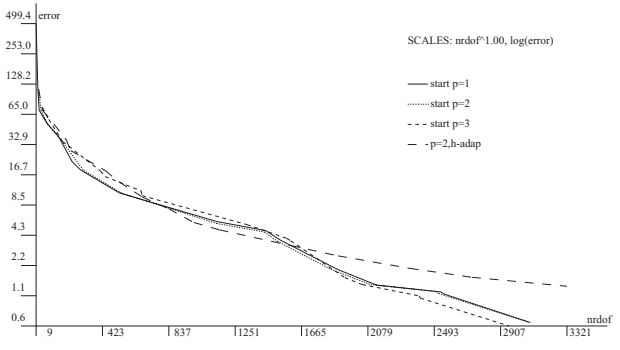

Fig. 11. Example 3: Convergence history for the *hp* refinements compared with the convergence history for an *h*-adaptive strategy with quadratic elements.

we estimate the error by computing the norm of the difference between the coarse mesh and fine mesh solutions. Here the fine mesh is obtained by performing only a global *h*-refinement. Representing the total error as the sum of element contributions,

$$\text{error}^2 = \sum_K \eta_K^2 \tag{3.31}$$

we refine all elements for which

$$\eta_K^2 \geqslant 0.7 \max_K \eta_K^2$$

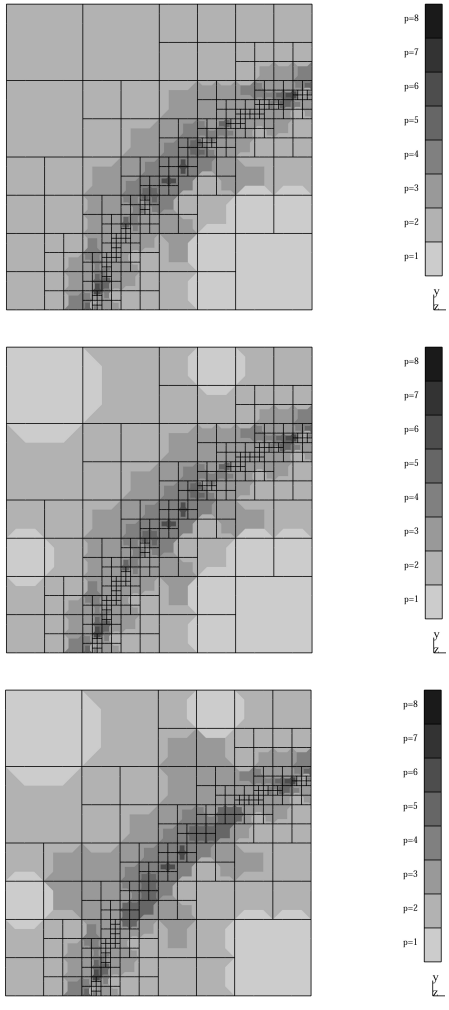

Fig. 12. Example 3: Final *hp* meshes.

Two comments:

- The results for all three *hp* meshes are very close to each other and indicate a little sensitivity with respect to the choice of the initial mesh. The slight difference in the number of d.o.f. may be attributed to the fact that the presented *hp* algorithm lacks *p* unrefinements. The ultimate meshes are also very similar to each other as it can be seen on Fig. 12.
- The *h*-adaptivity is very competitive in the preasymptotic range. In fact, it even slightly wins for a range of number of d.o.f.. Eventually though, the *hp* method starts converging exponentially, whereas the *h* method settles down with an algebraic rate of convergence (equal 2). The algebraic rate can be easier observed on Fig. 13 that represents the same convergence history on log–log scale.

Example 4: A Smoother Problem. At this point, we may start wondering whether the whole *hp* sophistication pays off?! After all, we are here after a good coarse mesh. Well, the results depend very much on the problem. For example, if we consider the same problem but with a smoother solution,

$$u_{\text{exact}}(\mathbf{x}) = \text{atan}(20r) \tag{3.32}$$

the convergence history presented in Fig. 14 is more favorable for the *hp* algorithm, also in the preasymptotic range.

Both algorithms have started with the same initial mesh of quadratic elements. The final meshes, delivering the 1% error are shown in Fig. 15.

Example 5: The L-Shape Domain Problem. In this standard test case for the *hp* methods, our algorithm performs very well. The results are again

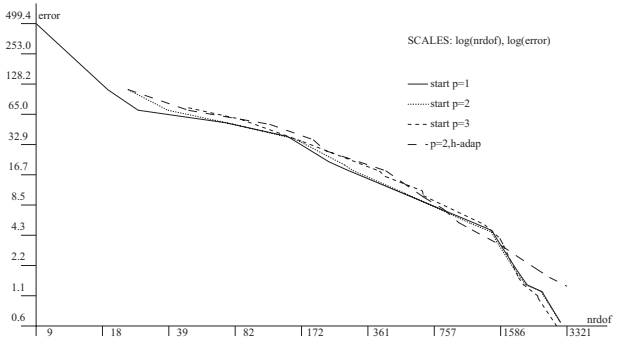

Fig. 13. Example 3: Convergence history for the *hp* refinements and *h* refinements on log–log scale.

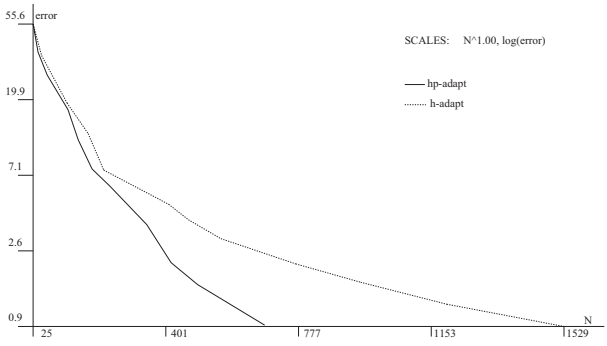

Fig. 14. Example 4: Convergence history for the *hp* refinements and *h* refinements on linear–log scale.

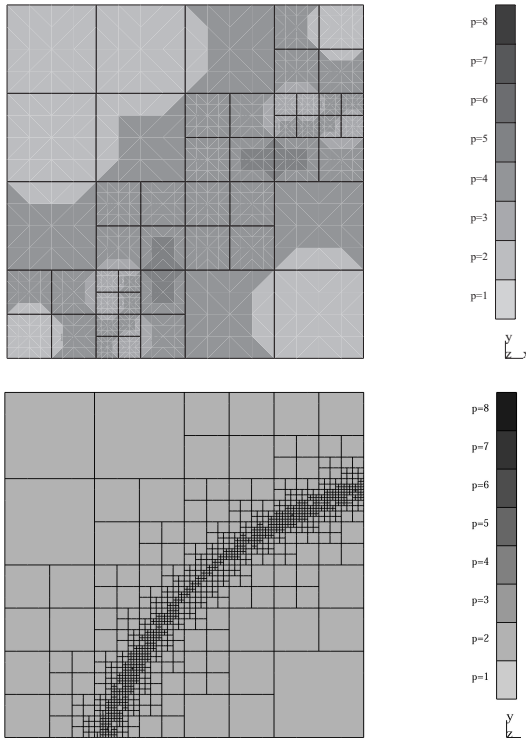

Fig. 15. Example 4: Optimal *hp* and *h* meshes.

independent of the initial mesh, as shown in Fig. 16. The *hp* method outperforms the *h* refinements but, again, only in the asymptotic range. In the preasymptotic range, the errors are comparable.

A typical, optimal *hp* mesh, presented in Fig. 17, coincides practically with the geometrically graded meshes of Babuška, and the linear graphs presented in Fig. 16, on scale $N^{1/3}$, log error, indicate the optimal, exponential convergence rates predicted by the theory, see [20]. The results of the *hp* mesh optimization are independent of the starting order of approximation. The algorithm starts choosing the *h*-refinement only when $p = 3$. If we begin with linear or quadratic elements, in the first couple of steps *p*-refinements are selected, with the order increasing to $p = 3$. Consequently, the three convergence history curves for the *hp*-adaptivity in Fig. 16 overlap each other.

4. FINAL REMARKS

We have presented a new, fully automatic, genuine *hp*-refinement strategy for a class of boundary-value problems that can be solved with H^1-conforming, i.e., continuous *hp* FE discretizations. The method is based on an interaction between two meshes, a coarse *hp* mesh, and a fine *hp* mesh obtained from the coarse mesh with a uniform *hp*-refinement. For quads this means that each element is refined into four element sons, and the order of approximation for each node in the mesh is increased by one. Having solved the problem on the fine mesh, we construct a new, optimal coarse mesh by minimizing the coarse grid *hp*-interpolant of the fine mesh solution.

The mesh has been tested on a number of 1D and 2D model problems, delivering optimal (exponential) convergence rates, consistent with the theory of optimal *hp* discretizations. Most importantly, the method delivers

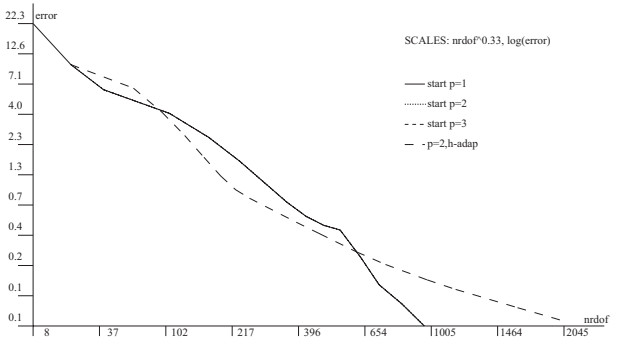

Fig. 16. Example 5: Convergence history for the *hp* refinements and *h* refinements on $N^{1/3}$-log scale.

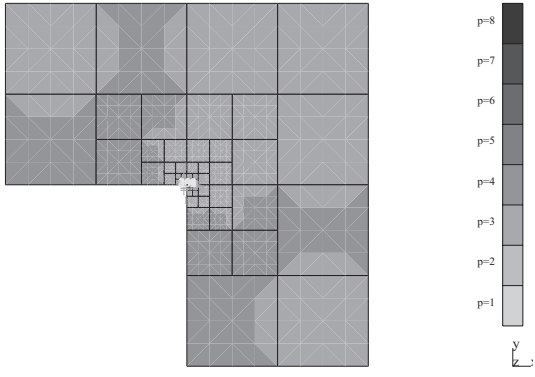

Fig. 17. Example 5: Optimal *hp* mesh for the L-shape domain problem.

meshes that are optimal in the full range of error level, including very coarse meshes.

The methodology is *problem independent* and can be coded as a "black box," and applied to solve different classes of problems as long as the minimization of the error in the H^1-norm is relevant.

The methodology extends to the 3D case, although theoretical foundations for 3D *hp* interpolation estimates are still missing. We are in process of coding a 3D version of the presented method and hope to present it soon. In the spirit of the de Rham diagram, see [7, 8] and the literature therein, the work can (and will be) extended to $H(\text{curl})$ or $H(\text{div})$-conforming *hp* discretizations. This is especially important for applications to Maxwell's equations, where the complex nature of singularities makes the explicit design of optimal meshes virtually impossible.

The method *is sensitive* to quadrature error. This has forced us to use expensive adaptive integration schemes, and prevented reporting the CPU time for the presented model cases. We do not think of the issue being critical as in the actual industrial problems loads are, most of the time, piecewise constant.

The methodology requires an efficient implementation, at minimum, of a two-grid solver. In 3D, the number of d.o.f. will jump from 200 thousand in the coarse mesh to up to 3,4 million in the fine mesh. Without the solver, the method has no chance for a survival in the "practical world."

The method employs a few parameters which may require a careful tuning. For instance, we might give a slight preference to the *h* refinement when deciding on the optimal refinement for edges. One of the reasons why the *h*-adaptive meshes of quadratic elements sometimes deliver slightly better results, may be explained by remembering that the fine mesh solution still does not support the full "spectrum" of the exact solution. By tuning our parameters, we may be able to get a little bit closer to the *h*-refined meshes in the preasymptotic range. We have not attempted any

such tuning in the presented work. According to our present experience, changing the parameters may result in slightly different refinement histories, but it does not produce dramatic changes in the final meshes.

Coding of the method is a real challenge and it represents a very difficult test for the data structure supporting *hp* refinements. We have not talked about this fundamental issue here at all, as we hope to present soon a substantially new concept in coding the *hp* methods [9]. Contrary to recent 3D implementations, see e.g., [21, 4], we still insist on using one-irregular meshes with hanging nodes…

In conclusion, we believe that the presented method may help us to develop a new, competitive tool for solving most difficult, multi-scale problems in complex 3D geometries.

ACKNOWLEDGMENTS

The work of the first and the second author has been supported by Air Force under Contract F49620-98-1-0255, and the work of the second author also by grant PB 1465/T11/2001/20 from Polish Science Foundation. The computations reported in this work were done through the National Science Foundation's National Partnership for Advanced Computational Infrastructure. The authors are greatly indebted to Prof. Ivo Babuška for numerous discussions on the subject.

REFERENCES

1. Ainsworth, M., and Senior, B. (1997). Aspects of an adaptive *hp*-finite element method: Adaptive strategy, conforming approximation, and efficient solvers. *Comput. Methods Appl. Mech. Engrg.* **150**, 65–87.
2. Babuška, I., and Guo, B. Q. (1996). Approximation properties of the *hp* version of the finite element method. In Babuška, I., and Oden, J. T. (eds.), *Computer Methods in Applied Mechanics and Engineering, Special Issue on p and hp-Methods*, Vol. 133, pp. 319–346.
3. Babuška, I., Strouboulis, T., and Copps, K. (1997). *hp* Optimization of finite element approximations: Analysis of the optimal mesh sequences in one dimension. *Comput. Methods Appl. Mech. Engrg.* **150**, 89–108.
4. Cugnon, F. (2000). *Automatisation des Caculs Éléments Finis dans le Cadre de la Méthode p*, Ph.D. Dissertation, Universite de Liege.
5. Demkowicz, L., Oden, J. T., and Devloo, Ph. (1985). On *h*-type mesh refinement strategy based on a minimization of interpolation error. *Comput. Methods Appl. Mech. Engrg.* **53**, 67–89.
6. Demkowicz, L., and Oden, J. T. (1996). Application of *hp*-adaptive BE/FE methods to elastic scattering. *Comput. Methods Appl. Mech. Engrg.* **133**(3/4), 287–318.
7. Demkowicz, L., Monk, P., Vardapetyan, L., and Rachowicz, W. (2000). De Rham diagram for *hp* finite element spaces. *Mathematics and Computers with Applications* **39**(7/8), 29–38.
8. Demkowicz, L., and Babuška, I. (2001). *Optimal p Interpolation Error Estimates for Edge Finite Elements of Variable Order in 2D*, TICAM Report 01-11, submitted to *SIAM J. Numer. Anal.*
9. Demkowicz, L., and Pardo, D. *The Ultimate Data Structure for Three Dimensional, Anisotropic hp Refinements*, TICAM Report, in preparation.

10. Houston, P., and Suli, E. *hp*-Adaptive discontinuous Galerkin finite element methods for first order hyperbolic problems. *SIAM J. Numer. Anal.*, to appear.
11. Ingerman, D., Druskin, V., and Knizhnerman, L. (2000). Optimal finite difference grids and rational approximations of the square root. I. Elliptic problems. *Comm. Pure Appl. Math.* **8**, 1039–1066.
12. Liszka, T., Tworzydlo, W., Bass, J., Sharma, S., Westermann, T., and Yavari, B. (1997). ProPHLEX—An *hp* adaptive finite element kernel for solving coupled systems of partial differential equations in computational mechanics. *Comput. Methods Appl. Mech. Engrg.* **150**, 251–271.
13. Melenk, J. M. (1997). On the robust exponential convergence of *hp* finite element methods for problems with boundary layers. *IMA J. Numer. Anal.* **17**, 577–601.
14. Novotny, A. A., Pereira, J. T., Fancello, E. A., and de Barcellos, C. S. A fast *hp* adaptive finite element mesh design for 2D elliptic boundary value problems. *Comput. Methods Appl. Mech. Engrg.*, in print.
15. Oden, J. T., Wu, W., and Ainsworth, M. (1995). Three step *hp* adaptive strategy for the incompressible Navier–Stokes equations. In Babuška, I., and Flahertyin, J. E. (eds.), *Modeling, Mesh Generation and Adaptive Numerical Methods for Partial Differential Equations*, IMA Minnesota.
16. Oden, J. T., Patra, A., and Feng, Y. (1992). An *hp* adaptive strategy. In Noor, A. K. (ed.), *Adaptive Multilevel and Hierarchical Computational Strategies*, ASME Publication, Vol. 157, pp. 23–46.
17. Patra, A., and Gupta, A. (2001). A systematic strategy for simultaneous adaptive *hp* finite element mesh modification using nonlinear programming. *Comput. Methods Appl. Mech. Engrg.* **190**, 3797–3818.
18. Rachowicz, W., Oden, J. T., and Demkowicz, L. (1989). Toward a universal *h-p* adaptive finite element strategy, Part 3. Design of *h-p* meshes. *Comput. Methods Appl. Mech. Engrg.* **77**, 181–212.
19. Schmidt, A., and Siebert, K. G. (2000). A posteriori estimators for the *hp* version of the finite element method in 1D. *Appl. Numer. Math.* **35**, 143–66.
20. Schwab, Ch. (1998). *p and hp-Finite Element Methods*, Clarendon Press, Oxford.
21. Zumbusch, G. (1996). *Simultaneous hp Adaption in Multilevel finite Elements*, Ph.D. Dissertation, Freie Universtität Berlin, 1995, published by Shaker Verlag, Aachen.