Reprinted from

Computer methods in applied mechanics and engineering


A parallel adaptive strategy for $hp$ finite element computations

J.T. Oden$^{1,*}$, Abani Patra$^2$

Texas Institute For Computational and Applied Mathematics, University of Texas at Austin, Austin, TX 78712, USA

Received 1 May 1994
A parallel adaptive strategy for $hp$ finite element computations

J.T. Oden$^1$, Abani Patra$^2$

Texas Institute For Computational and Applied Mathematics, University of Texas at Austin, Austin, TX 78712, USA

Received 1 May 1994

Abstract

In this work, a new three-step adaptive strategy for $hp$ finite element discretizations is presented. Techniques for parallelizing these calculations on distributed memory multiprocessor computers are also developed. Numerical examples are used to demonstrate convergence properties and parallel efficiencies. Orders of magnitude reduction in computational efficiency over conventional methods are observed.

1. Introduction

Theoretically, $hp$ finite element methods are capable of delivering exponential rates of convergence, and such rates have been observed frequently in numerical experiments (e.g. [7, 19]). This remarkable fact has broad implications in many areas of scientific and engineering computation: exponentially convergent schemes should be able to deliver results of a prescribed accuracy with orders of magnitude fewer degrees of freedom (dof) than conventional $h$ version FEMs, finite difference or finite volume methods, these being capable of converging only algebraically. If, in addition, such super-algebraic schemes could be parallelized, then additional efficiencies over classical approaches can be attained.

Two major difficulties in attaining such dramatic improvements in performance are:

1. to achieve exponential convergence rates, a sequence of solutions on good $hp$ meshes must be obtained, necessitating the expenditure of some computational overhead and requiring the construction of adequate meshes having non-uniform distribution of mesh size $h$ and element order $p$, and

2. a non-conventional parallelization strategy must be devised that allows for load balancing on non-uniform $hp$ meshes.

The first difficulty presents a possible paradox: if exponential convergence of error with respect to problem size is indeed possible, is the computational effort needed to attain it so large that the advantage of such schemes are completely lost? If the second difficulty can be overcome, can the computational overhead needed to produce exponentially convergent schemes be offset by parallel computation?

The present work addresses adaptive parallel strategies aimed at resolving these issues. An adaptive $hp$ scheme, preliminary versions of which were discussed in [13] is presented, which is based on theoretical a priori and a posteriori error estimates and which, experimentally, delivers exponential convergence with respect to both problem size and computational effort, as measured in CPU times. This performance is observed without multiprocessor computing. However, in additional, a parallel version of the scheme is described which provides additional speedups when implemented on...
multiprocessor computers. To the best of our knowledge, few adaptive \( hp \) schemes have been proposed in the literature for general \( hp \) meshes. For general \( hp \) meshes, Rachowicz et al. [19] developed an \( hp \) adaptive strategy that led to near optimal meshes for one- and two-dimensional problems, but this scheme could require many sequential steps and meshes before yielding a good mesh. A more advanced version of this algorithm is used in the commercial packages P3/CFD and PHLEX.

In this paper, we present an \( hp \)-adaptive strategy that leads to reasonable \( hp \) meshes with low computational costs. The algorithm is described in relation to a model elliptic boundary-value problem and the results of a number of numerical experiments are given to demonstrate its effectiveness and limitations.

2. Model problem and preliminaries

We focus on a model class of Poisson problems in which a function \( u = u(x, y) \) is sought satisfying

\[
\begin{align*}
-\Delta u &= f \quad \text{in } \Omega \\
u &= w \quad \text{on } \partial \Omega
\end{align*}
\]

where \( \Delta \) is the Laplacian, \( \Omega \) is an open polygonal domain in \( \mathbb{R}^2 \) with boundary \( \partial \Omega \). The data of the problem class is characterised as follows:

(i) The boundary \( \partial \Omega \) consists of a set of line segments meeting at a number \( M_r \) of vertices \( A \) with interior angle \( \alpha_A, 0 \leq \alpha_A < 2\pi \), \( A = 1, 2, \ldots, M_r \). We denote

\[
\alpha = \frac{\pi}{\max(\alpha_A)}
\]

where \( \alpha \) is the order of the singularity associated with the solution of (1)

(ii) Using standard Sobolev space notations, the regularity of the source function \( f \) is defined by

\[
f \in H^s(\Omega), \quad s \geq 0
\]

(iii) The regularity of the solution \( u \) of the model problem is characterized by

\[
\| u \|_{r,n} \leq C \| f \|_{s,n}
\]

where \( \| \cdot \|_{r,n} \) is the Sobolev norm on \( \Omega \) and \( r = \min(s + 2, 1 + \alpha - \varepsilon) \)

\[
u \in (w^*) + H^1_0(\Omega) = V
\]

where \( w^* \) is an \( H^1 \) function, the trace of which coincides with the boundary data \( w \) on \( \partial \Omega \) and \( \varepsilon \) is an arbitrary positive number.

We next consider a family of finite element approximations of (1) in which the domain \( \Omega \) is approximated by a sequence of meshes \( \{ \Omega_h \} \) over which piecewise polynomial approximations \( u^{hp} \) of \( u \) are constructed. We define

\[
\Omega = \bigcup_{K=1}^\infty \Omega_K \\
\text{diam}(\Omega_K) = h_K \\
u^{hp}|_{\Omega_K} \in \mathcal{P}_{p_h}(\Omega_K) \\
u^{hp} \in V^{hp} \subset H_0^1(\Omega) + \{ w^* \} \\
w^* \in H^1(\Omega), \quad \gamma w^* = u \quad \text{on } \partial \Omega
\]

Thus, we consider conforming \( hp \)-approximations of (1) defined by a family of finite element subspaces \( \{ V^{hp} \} \) of \( V \), such that the restrictions of functions on \( V^{hp} \) to an element \( \Omega_K \) belongs to the space of \( \mathcal{P}_{p_h}(\Omega_K) \) of polynomials of degree \( p_K \) over \( \Omega_K \). Such \( hp \)-finite element approximations, in which the element size \( h_K \) and the degree \( p_K \) of element shape functions vary over the mesh, are
discussed in [19]. We assume standard properties on the regularity of meshes (e.g. quasiformity) so that standard asymptotic approximation properties hold. In particular if \( e = u - u^{hp} \) is the approximation error,

\[
\|e\|^2 \leq C \sum_{K=1}^{N} \frac{h^{2p_K}_K \|u\|^2_{r,A_K}}{p^{2p'_K}_K}
\]

(7)

where \( C \) is a constant independent of element size \( h_K \), order \( p_K \), or \( u \); \( \|u\|_{r,A_K} \) is the \( H^r \) norm of the exact solution over element \( \Omega_K \) and

\[
\mu_K = \min(p_K, r - 1) \quad \nu_k = r - 1
\]

(8)

and we have used the notation \( \|\cdot\|_1 = \|\cdot\|_{1,A} \).

In view of (4), we shall assume that (7) can be written in the form

\[
\|e\|^2 \leq \sum_{K=1}^{N} \frac{h^{2p_K}_K \Lambda^2_K}{p^{2p'_K}_K}
\]

(9)

where

\[
\Lambda_K = C \|u\|_{r,A_K}
\]

(10)

According to Babuska and Suri [3], when a singularity of strength \( \alpha \) exists due to the presence of corners, defined earlier, the global rate of convergence of the \( p \)-version finite element approximation in \( H^1 \) is \( O(p^{-\min(s+1,2\alpha)}) \). We will therefore take as exponents in (8),

\[
\mu_K = \min(p_K, s + 1, \alpha), \quad \nu_K = \min(s + 1, 2\alpha)
\]

(11)

3. The \( hp \)-adaptive strategy

The goals in a successful \( hp \) adaptive strategy are twofold: (1) the strategy should lead to a discretization that corresponds to a solution with a specified accuracy and (2) this should be accomplished in a minimum number of steps with a minimum number of unknowns. Such a mesh is termed ‘optimal’. It has been shown [19] that for an optimal mesh using \( h \) refinements, the error is equidistributed over all the elements asymptotically as \( h \to 0 \). Using this tool and a priori estimates of the form (8), we now outline two simple algorithms that result in a good \( hp \)-mesh in three steps.

We consider a sequence of three meshes: \( \Omega_h \), \( \Omega_p \) and \( \Omega_h \). There are \( N_0 \) elements in \( \Omega_h \) and \( N_f \) elements in \( \Omega_h \). Our strategy is based on the idea of treating the inequality (9) as an equality, a risky assumption, but one that seems to give some basis for an effective scheme. Thus, we set for some mesh \( \Omega_h \) with \( N \) elements,

\[
\|e\|^2 = \sum_{K=1}^{N} \frac{h^{2\mu_K}_K \Lambda^2_K}{p^{2p'_K}_K}
\]

(12)

This is the starting point for the development of our algorithms.

3.1. Notation

We introduce some preliminary notation which is used in describing the algorithm:

- \( n \) = error index = \( \|e\|_{1,\Omega} / \|u\|_{1,\Omega} \), \( u \) = a solution of (1).
- \( \theta_K \) = local error indicator provided by an appropriate a-posteriori error estimate of \( \|e\|_{1,\Omega_K} \)
- \( \Theta = \left( \sum_{K=1}^{N} \theta^2_K \right)^{1/2} \) = global error estimate (= \( \|e\|_{1,\Omega} \))
- \( \Lambda = \left( \sum_{K=1}^{N} \Lambda^2_K \right)^{1/2} \), \( \Lambda_K \) being defined by (10)
3.2. Algorithm

Step 0. Initialization

Begin by defining a target error index

$$\eta_t \equiv \eta_{\text{TARGET}} = \frac{\|e\|_1}{\|u\|_1}$$  \hfill (13)

and an intermediate error level

$$\eta_i = \gamma \eta_t$$  \hfill (14)

where $\gamma$ is a parameter generally ranging from 5 to 10. For example, let $\eta_t = 0.01$ (1% error) and $\eta_i = 0.07$.

Step 1. Introduce an initial mesh $\Omega_{n_0}$ of $N_0$ elements with uniform $p = p_0$ and an almost uniform mesh size distribution $h_0(h_0 \geq h_{\alpha}, 1 \leq e \leq N_0)$ sufficiently fine that the solution will fall in the asymptotic range of the convergence curve for $h$-refinements. Then solve the problem on the initial mesh and use an a posteriori error estimator to estimate the global and local errors $\theta_0$ and $\theta_k$

$$\|e\|_1^2 = \theta_0^2 = \sum_{k=1}^{N_0} \theta_{0k}^2$$  \hfill (15)

where $N_0$ is the number of elements in the initial mesh and $\theta_{0k}$ is the element level error indicator. Now, from the orthogonality of the error to $V^{hp}$

$$\|u\|_1^2 = \|U_0\|_1^2 = \|u_{00}\|_1^2 + \theta_0^2$$  \hfill (16)

$\|U_0\|_1$ can be used to compute an approximation of the initial error index

$$\eta_0 = \frac{\theta_0}{\|U_0\|_1}$$  \hfill (17)

Since the initial mesh is essentially uniform, using (12) and setting $\mu_k$ to $\mu_0$ and $\nu_k$ to $\nu_0$, we can obtain

$$\theta_0^2 = \|e\|_1^2 = h_0^{2\mu_0} \sum_{k=1}^{N_0} A_k^2 = \frac{h_0^{2\mu_0} A^2}{p_0^{2\nu_0}}$$  \hfill (18)

or

$$A^2 = \frac{\theta_0^2 p_0^{2\nu_0}}{h_0^{2\mu_0}}$$  \hfill (19)

By definition

$$\theta_0^2 = \sum_{k=1}^{N_0} \theta_{0k}^2$$  \hfill (20)

$$A^2 = \sum_{k=1}^{N_0} A_k^2$$  \hfill (21)

Eqs. (19)-(21) suggest that locally we can approximate $A_k$ as

$$A_k = \frac{\theta_{0k} A}{\theta_0}$$  \hfill (22)

For the next step, we generate an intermediate mesh $\Omega_{h_1}$ by using only $h$-refinement, keeping $p = p_0$ fixed and constructing a mesh with varying mesh size ($1 \leq f \leq N_1$) that will produce the intermediate error $\eta_i = \|e\|_1/\|u\|_1$.

Step 2. For an optimal $h$-adaptive mesh to achieve $\eta_i$, we need to
(i) reduce global error
(ii) equidistribute the error [19].

We achieve this by computing for each element in the ‘old’ (initial) mesh, the numbers,

$$ n_K = \left[ \frac{\Lambda_k^2 N_1 h_0^{2\mu} \gamma^{1/3}}{p_0^{2\mu} \theta_i} \right]^{1/3} $$

(23)

where $n_K$ is the number of new elements required in the area covered by the $K$th element of the initial mesh. $N_1$ is the number of elements in the intermediate mesh. $\beta$ is a parameter depending on the dimension of the problem, and $\theta_i = \eta_i \|U_0\|_i$ is the estimated global error for the intermediate mesh $\Omega_{h_i}$. The parameter $\beta = 2$ for one-dimension, $\beta = 1$ for two-dimensions and $\beta = 2/3$ for three-dimensions. The derivation of (23) is given in Appendix 1.

The principal difficulty in computing $n_K$ is that $N_1$ will be known only after the intermediate mesh is known. However, the $n_K$ computed must add up to $N_1$

$$ \sum_{K=1}^{N_0} n_K = N_1 $$

(24)

Eqs. (23) and (24) can now be solved iteratively for $N_1$ and $n_K$. Having obtained $n_K$, we redistribute inside the element using a weighting scheme based on errors in partitions of the initial element. If $n_K < 1$, we can consider unrefinements. Now solve the problem on the $h$-adapted mesh and compute a posteriori error estimate $\theta_i$ at the global level and $\theta_f$ at the element level.

**Step 3.** We now construct the third mesh by computing a distribution of polynomial degrees $p_T$ to form a final mesh $\Omega_{h_f}$ which should deliver the target error $\eta_T$. At this stage, an intermediate mesh $\Omega_{h_i}$ is available with non-uniform mesh sizes $h_f$ and with $N_1$ elements on which the intermediate error $\theta_i$ is achieved and equidistributed. Towards this end, we use the actual error distribution $\theta_i$ to compute $\Lambda_f$ on the $h$-adapted mesh $\Omega_{h_i}$ as

$$ \Lambda_f^2 = \frac{h_0^{2\mu} A^2}{\gamma^2 \theta_f^2} $$

(25)

The target error must satisfy

$$ \theta_T^2 = (\eta_T \|U_0\|^2) = \sum_{f=1}^{N_1} \frac{h_f^{2\mu} A_f^2}{p_f^{2\mu}} $$

(26)

Now use $p$ refinements to achieve the target error by computing $p_T$ using $\Lambda_f$ and

$$ p_{fT}^{2\mu} = \frac{h_f^{2\mu} A_f^2 N_1}{\theta_T^2} $$

(27)

Thus, we enrich $p$ on each element of $\Omega_{h_i}$ to obtain $\Omega_{h_f}$. We then solve the problem on $\Omega_{h_f}$ and compute an estimate of the final error index $\eta_T$.

**Step 4. Post Loop Calculation.** If $\eta_T \leq \eta_T$ the computation is terminated; otherwise Steps 2–4 are repeated with higher $\gamma$ starting with $\Omega_{h_f}$ as the initial mesh for the next 3 step series.

**REMARK 2.1.** The choice of the initial mesh should be made with some care. It must be fine enough to ensure that the error is already in the ‘asymptotic range’ but at this stage there is no way of knowing if this is achieved. Higher initial $p_0$ may be used to achieve this. Special knowledge, if any, of the solution may also be used in this step to grade the mesh for better performance of the algorithm. All the examples presented in this study do not use any special grading.

**REMARK 2.2.** It is not necessary to use a uniform $p = p_0$ in Step 1; a non-uniform $p = p_0k$ can be used; then (18) must be modified appropriately.
3.3. Algorithm 2

In numerical experiments that we discuss in subsequent sections, we show that the performance of this algorithm is very good on problems with smooth solutions and point type singularities. On problems with sharp gradients along a line the performance can deteriorate, the primary reason being that in the second stage higher-order elements are used to resolve a sharp gradient. This yields poor convergence rates with respect to \( p \) on elements that contain the gradient. This deficiency is overcome by modifying the original algorithm by using estimated convergence rates computed locally on an element by element basis, from coarse mesh solutions. We outline this modified algorithm below:

**Step 0.** As before, we define target and intermediate errors

\[
\eta_T = \eta_{\text{TARGET}} \approx \frac{\|e\|}{\|u\|}, \quad \eta_I = \gamma \eta_T
\]

with \( \gamma \) a parameter generally ranging from 5 to 10.

**Step 1.** Introduce an initial mesh \( \Omega_{h_0} \) of \( N_0 \) elements with generally (but not necessarily) uniform \( p = p_0 \) and an almost-uniform mesh size distribution \( h_0(h_0 \approx h_{\theta K}, 1 \leq K \leq N_0) \) sufficiently fine to fall in the asymptotic part of the convergence curve for \( h \)-refinements. Solve the problem on the initial mesh and use an a posteriori error estimator to estimate the global and local errors \( \theta_0 \) and \( \theta_{0K} \):

\[
\|e\|_I^2 = \theta_0^2 = \sum_{K=1}^{N_0} \theta_{0K}^2
\]

where \( N_0 \) is the number of elements in the initial mesh and \( \theta_{0K} \) is the element level error indicator. We proceed as before, computing \( \|U_0\|_I^2 = \|u\|_I^2 \) and \( \eta_0 \) using (16), (17) and (18). Now

\[
\theta_0^2 = \|e\|_I^2 = \sum_{K=1}^{N_0} \|e_K\|_I^2 = \sum_{K=1}^{N_0} \frac{h_0^{2\mu_K}}{p_0^{2\nu_K}} \Lambda_K^2
\]

By definition (30), (20) and (21) suggest that locally we can approximate \( \Lambda_K \) by

\[
\Lambda_K = \frac{p_0^{2\nu_K} \theta_K^2}{h_0^{2\mu_K}}
\]

At this stage, we need the rates \( \mu_K \) and \( \nu_K \). To compute these, we generate two other coarse uniform meshes, one at a value of \( h \) slightly finer and another at a \( p = p_1 \), level one greater than \( p_0 \) (see for instance Fig. 1). Solve the problem on these meshes and using the local errors on these problems compute for all elements in the initial mesh

\[
\mu_K = \frac{\log \theta_{K0} - \log \sqrt{\sum_{K=1}^{N_0} \theta_{K0}^2}}{\log h_0 - \log h_1}
\]

\[
\nu_K = \frac{\log \theta_{K1} - \log \theta_{K0}}{\log p_1 - \log p_0}
\]

The remainder of the algorithm is similar to the previous algorithm with two differences. Firstly, the local rates \( \mu_K \) and \( \nu_K \) are used instead of the global rates \( \mu_0 \) and \( \nu_0 \). To prevent \( p \) enrichments being used to resolve sharp gradients, we use the element \( p \) convergence rate to isolate elements that contain such gradients. As before, we generate an intermediate \( h \) mesh using only \( h \)-refinement and follow with a \( p \)-refinement on all but the elements flagged as unresponsive to \( p \) enrichments.
Step 2. As before, we attempt to reduce the global error to \( \eta_1 \) and equidistribute it. We achieve this by computing for each element in the old mesh

\[
N_K = \left[ \frac{2^{2p_K^2} \eta_1}{p_0^{2p_K} \theta_1^2} \right]^{1/2p_K+1}
\]  

where \( N_K \) is the number of new elements required in the area covered by the \( K \)th element of the initial mesh. \( N_1 \) is the number of elements in the intermediate mesh. \( \beta \) is a parameter depending on the dimension of the problem, and \( \theta_1 = \eta_1 \|U_0\|_1 \) is the estimated global error for the intermediate mesh \( \Omega_{h_1} \). The parameter \( \beta = 2 \) for one dimension, one for two dimensions and \( 2/3 \) for three dimensions. The derivation of (34) is similar to that for (23) given in Appendix 1.

Again, we must solve (34) together with

\[
\sum_{K=1}^{N_1} N_K = N_1
\]  

iteratively for \( N_1 \) and \( N_K \). Having obtained \( N_K \), we redistribute inside the element using a weighting scheme based on errors in partitions of the initial element. If \( n_K < 1 \) we can consider unrefinements. After solving the problem on this mesh, we do adaptive \( p \) refinement on the \( h \)-adapted mesh using a posteriori error estimates on the \( h \)-adapted mesh.

Step 3. We use the actual error distribution to compute \( \Lambda_K \) on the \( h \)-adapted mesh \( \Omega_{h_1} \), according to

\[
\Lambda_K^2 = \frac{p_0^{2p_K \theta_1^2 \Lambda_1^2}}{h_1^{2p_K \theta_1^2 k}}
\]  

The convergence rates \( \nu_K \) with respect to \( p \), computed on the initial mesh are now passed on to the refined mesh. Each element in the new mesh inherits the convergence rate of its parent element.

To isolate elements that are unresponsive to \( p \) enrichment, define a cutoff convergence rate \( \nu_{cut} \). A choice of 0.5 appears to work well. A priori estimates, where available, can be used to define this.

Let \( N_D \) be the number of such elements. The target error must then satisfy

\[
\theta_1^2 = (\eta_1 \|U_0\|_1^2) = \sum_{K=1}^{N_1-N_D} \frac{h_1^{2p_K} \Lambda_K^2}{p_0^{2p_K}} + \sum_{K=N_D}^{N_1} \frac{h_1^{2p_K} \Lambda_K^2}{p_0^{2p_K}}
\]  

Now use \( p \) refinements to achieve the target error by computing \( p_{ref} \) using \( \Lambda_1 \) and

\[
p_{ref}^{2p_K} = \frac{h_1^{2p_K} \Lambda_1^2 N_1}{\theta_1^2}
\]  

Thus, we enrich \( p \) on each element of \( \Omega_{h_1} \) to obtain \( \Omega_{h_f} \).
Step 4. Post Loop Calculation. Solve the problem on $\Omega_h$ and compute an estimate of the final error index $\eta_F$. If $\eta_F \leq \eta_T$ the computation is terminated; otherwise Steps 3–5 are repeated with higher $\gamma$.

4. Numerical results

In this section we describe a few sample applications of the algorithms primarily to demonstrate their performance. For more complex applications see also the work of Legat and Oden [12], Wu et al. [15] and Bey [2]. Problems with both point- and line-singularities are solved. Suitable choices of boundary conditions and data $f$ are made to obtain variations in solution regularity properties.

**EXAMPLE 1.** The first test problem is that of Poisson's equation on a L-shaped domain with Dirichlet boundary conditions. This problem has a point singularity at the reentrant corner of order

$$\alpha = (\pi/\omega) = \pi/(3\pi/2) = 2/3$$

This problem has been studied by several researchers [10] and some comparison data is available. First, conventional uniform $h$ and $p$ refinements are carried out. The uniform refinements achieve the theoretical convergence rates of $1/3$ and $2/3$ for uniform $h$ and $p$, respectively. Now implement the first algorithm outlined in the previous section.

**Step 1.** A target error of $\eta_T = 0.01$ and an intermediate error of $0.06$ ($\gamma = 6$) are set.

**Step 2.** An initial mesh of 12 bilinear ($p_B = 1$) elements (Fig. 2) is used to solve the problem. The normalized $H^1$ error norm $\eta_0$ computed for this solution is 0.12323. $A$ and $A_K$ are also computed.

**Step 3.** Element numbers $n_K$ and $N_i$ are iteratively computed. Four iterations are sufficient for convergence at an intermediate mesh with 26 elements. The problem is solved on this intermediate mesh. The $H^1$ norm of the error computed is 0.0678.

![Initial mesh for Poisson's equation on L-shaped domain.](image)

![Final mesh for Poisson's equation on L-shaped domain with Dirichlet boundary conditions and solution $u = r^{2/3} \sin(2\theta/3)$.](image)
Step 4. Compute $A_f$, and using this, compute $p_f$ for each element in the $h$-adapted mesh. Enriching the mesh to the levels desired leads to $\Omega_h$ (see Fig. 3).

Step 5. Solve the problem on this final mesh, which is shown in Fig. 3. The final error index $\eta_F$ is seen to be 0.107 i.e. $\eta_F = \eta_T$.

The mesh resulting from the 3-step algorithm has 3 levels of $h$-refinement and 5 levels of $p$-refinement. It produces the finest $h$ near the singularity but, contrary to expectations, it also applies the highest level of $p$, $p = 5$, also in the two elements next to the singularity. Conventionally, an optimal mesh for a problem of this type [7] has been hypothesized to have a geometrically graded mesh near the singularity and a $p$ distribution increasing away from the singularity with the largest element possessing the highest degree of $p$. The result of this experiment seems to indicate that good $hp$ meshes can be generated that do not have this structure.

Further, the choice of initial mesh and intermediate error also affect the final mesh obtained. The 'mesh density' i.e. the distribution of the number of dof, seems to be inversely proportional to the error distribution. This observation could be applied when designing a strategy to obtain a load balanced domain decomposition scheme for $hp$ adaptivity.

In Fig. 4, the convergence rate of the 3-step scheme and of uniform $h$ and $p$ refinements against dof is plotted. The 3-step method achieves an order of magnitude improvement in the convergence rate against $h$ and $p$ refinements ($10^{2.5}$ degrees of freedom for the 3 step as opposed to $10^7$ for $p$ refinements and $10^{4.7}$ for $h$ refinements to achieve an error index of $\eta = 0.1$).

In Fig. 5, convergence rates are plotted versus cpu-time for results obtained on an Alliant FX/8 computer. Similar improvements in convergence rates are observed. However, the steeper slope observed for the $p$ extension in Fig. 4 are not seen and both uniform $h$ and $p$ extensions appear to have the same performance when measured against cpu-time. This deterioration of performance of the $p$ extensions may in part be due to the choice of a frontal solver for the problem. This does, however, indicate that the minimization of the number of dof need not necessarily produce the best mesh in an $hp$ adaptive refinement situation. The computational effort required for $p$ version calculations is strongly affected by the lack of sparsity of the matrices, the form of the polynomial bases, and the type of the solver used. Our results show that for the direct solver used, the additional computation effort associated with a dof added due to an increase in $p$ and that associated with an increase in $h$ refinement are not the same. Use of faster preconditioned iterative parallel solvers [5] may reduce the solution time and bring the convergence more in line with that seen in Fig. 4.

Fig. 4. Convergence rates for 3 step and uniform refinements for Poisson's equation on L shaped domain with Dirichlet boundary conditions and solution $u = r^{1/3} \sin(2\theta/3)$.
Fig. 5. Convergence rates for 3 step and $h$ adaptive refinements for Poisson's equation on L shaped domain with Dirichlet boundary conditions and solution $u = r^{2/3} \sin(2\pi/3)$.

In Figs. 6–11, we compare the performance of the 3 step method against the $h$, $p$, and $hp$ adaptive strategies built into PHLEX. In Fig. 6, we observe that the final $hp$ mesh in the 3-step method has almost an order of magnitude fewer dof than the $h$-adaptive method ($10^{2.5}$ against $10^{3.2}$). Further, the $h$-adaptive method requires many more resolutions than the 3 step method. The computational cost of this is seen in Fig. 8 where the cpu-time anticipated for the $h$-adaptive method is about 1000 s compared to the 50 s for the 3-step method to achieve the same error level. In Figs. 7 and 9, the 3 step and $p$-adaptive procedures are compared. The $p$-adaptive procedures achieve almost the same number of dof in the final mesh, but the number of resolutions is 5 as opposed to 3 in the new algorithm. This larger number of resolutions results in higher cpu times for $p$-adaptive methods.

Fig. 6. Convergence rates for 3 step and $p$ adaptive refinements for Poisson's equation on L shaped domain with Dirichlet boundary conditions and solution $u = r^{2/3} \sin(2\pi/3)$. 
In Figs. 10 and 11 the performance of an \textit{hp} adaptive algorithm available in PHLEX is compared to the 3-step method. The results indicate that the same convergence rate is achieved by the 3-step algorithm with three fewer solution steps. The effect of this is seen in Fig. 10 where the cpu time used is much more, even though the convergence rate with respect to dof is the same.

\textbf{EXAMPLE 2.} A problem with a sharp internal layer is now examined. A rectangular domain with Dirichlet boundary conditions and exact solution of the form $u = \tan^{-1}(\alpha(\xi - \xi_0))(1 - x)x(1 - y)y$ is solved, where $\xi = (x + y)/\sqrt{2}$, $\alpha$ and $\xi_0$ are parameters that control the sharpness and location of the layer. Here, $\alpha = 17.5$ and $\xi_0 = 0.7$ are used. The error is reduced an order of magnitude and almost...
equidistributed, as we desired. Figs. 12 and 13 show the convergence rates for this problem using Algorithm 1. The behavior is similar to that observed in Example 1.

However, for higher values of $\alpha (\alpha = 50)$, Algorithm 1 fails. Algorithm 2 is then applied to this problem with local convergence rates $\nu_k$ used for the $p$ convergence rates and $\mu_k = \mu_0$ for the $h$ convergence rates. The convergence rates are shown in Fig. 14. The effect of higher $p$ on the elements with sharper gradients is immediately obvious as the uniform $p$ enrichment actually increases the error for $p$ larger than 2. The initial mesh is also relatively coarse and does not quite catch the high gradient, but subsequent $h$ refinements do reduce the error. The 3-step algorithm with the local rates for $p$
convergence performs quite well, consistently reducing the error even from the very coarse initial mesh. The mesh finally generated is shown in Fig. 15.

5. Parameter selection for optimum performance

The efficiency of the 3-step algorithm depends on both the initial mesh and the prescribed intermediate error. These, at present, are heuristic choices. Performance is clearly superior for certain combinations of initial mesh and intermediate mesh. The effect of $\gamma$ is particularly interesting, as it loosely determines the relative amounts of error reduction due to $h$ and $p$ refinements. Very low $\gamma$ results in an almost totally $p$-adaptive mesh while a high $\gamma$ results in more $h$-adaptivity.

It is our experience that for best mesh generation we must
(i) pick an optimal $\gamma$ (the intermediate error selector)
(ii) start with an appropriate initial mesh.
(iii) compute the convergence rates $\mu$ and $\nu$ with some care.
Fig. 13. Convergence rates for 3 step on Poisson's equation with solution \( u = \tan^{-1}(x + y - x_0)x(1 - x)y(1 - y) \) for \( \alpha = 50 \) using Algorithm 2.

Fig. 14. \( hp \) mesh generated for 3 step on Poisson's equation with solution \( u = \tan^{-1}(x + y - x_0)x(1 - x)y(1 - y) \) for \( \alpha = 50 \) using Algorithm 2.
To obtain some additional insight into this procedure, the model problem is solved with several choices of $\gamma$ and different types of initial meshes. First, we analyze the results from three types of initial meshes. In Fig. 16, we start with 3 uniformly $h$-refined meshes. While the slopes attained look similar, the coarsest mesh produces the desired error with lowest number of dof. We then compare (Fig. 17) three graded meshes with the area closer to the singularity graded with $h = 1/2$ and $h = 1/4$, respectively. Again this does not appear to aid the convergence rates. However, starting with a finer mesh does cause some unnecessary refinement and additional dof in the final mesh.

In Figs. 18 and 19, the effect of starting with a finer initial mesh and using unrefinement and initial meshes with quadratic and cubic elements are compared. Based on these results, it appears that the convergence rate is relatively insensitive to the initial mesh chosen. However, the underlying assumption in developing the algorithm that all meshes used, lead to solutions for which the errors are
in the asymptotic range. If this assumption is not satisfied, the prediction of meshes based on convergence rates in a-priori estimates is invalid and poor results are obtained. It is also easy to see that the coarsest mesh satisfying this assumption will lead to the least amount of over-refinement.

The choice of $\gamma$ is more interesting. We show the effect of various values of $\gamma$ on the convergence rates and on the overall cpu time used for solving the problem in Figs. 20 and 21. There appears to be an optimal value of $\gamma$ for which the solution time is minimum. This is perhaps to be expected as $\gamma$ is a measure of the amount of $h$ versus $p$ refinement used in designing the mesh. This optimal value of $\gamma$ also appears to be problem dependent and possibly related to the convergence rates $\mu$ and $v$. More or less, $h$-refinement will cause excessive $p$-refinement and much larger solution costs. The optimal selection of $\gamma$ will also depend on the nature of singularities present in the solution. A problem with a sharp internal layer, as in Example 2, needed a much higher $\gamma$ (i.e. more $h$ refinement than a problem with a point singularity, as in Example 1). On both examples, a choice of $\gamma = \mu / v$ appears to be optimal.

![Fig. 17. Convergence rates for 3 step with different initial meshes.](image1)

![Fig. 18. Convergence rates for 3 step with different initial meshes.](image2)
6. Parallel implementation

In this section, the parallelization of the 3-step adaptive strategy described earlier is discussed. This necessarily involves implementation of techniques for partitioning and parallel iterative solution described in Oden and Patra [14] and Oden, Patra and Feng [14] into one comprehensive package.

Two main paradigms for implementing the adaptive strategy in parallel are considered: a master-slave strategy and a SPMD (single program multiple data) model. The ease of implementation of a particular model and its efficiency is obviously dependent on the chosen parallel architecture. In particular, the relative cost of communication to computation will determine the granularity at which the overhead of parallelizing will be matched by the gain in speed in running on multiple processors.

For the distributed memory multiprocessors with large numbers of processors that are used in this study, the bottleneck in communication between a single master node and many slave nodes rules out that approach. So we propose an approach based on the SPMD approach.
The largest obstacle encountered in the preliminary work described here is in the partitioning of the complex data structure necessary to support \( hp \) adaptivity. While several ad hoc patches have been made to produce a functioning parallel code, it was determined that a comprehensive new data structure is needed to support partitionability. This will be an important subject of future work.

In analyzing physical systems, there appear to be two approaches to introducing concurrency. In the first, explicit substructuring techniques of various kinds are used. The parallelism then is generally at a very high level and such methods are normally made up of explicit subdomain solves followed by some communication and the solution of an interface problem. In the second method, no explicit substructuring is used but different phases of the analysis are carried out in parallel at the operation level, e.g. the parallel solver and parallel matrix generation. The explicit substructuring techniques have the advantage of lower communication costs, but the resulting interface problem is often very expensive to both formulate and solve.

6.1. SPMD model

In this model, the underlying philosophy is to eliminate excessive communication by duplication of some of the less intensive computations on all the processors. Thus, the data structure information is duplicated across all processors. Further, each processor is required to carry a copy of the solution and error information for the whole problem and not just the subdomain assigned to it. Consequently, the adaptivity and partitioning computations can be duplicated on each processor as is the initial mesh solution. However, the compute intensive parts, creation of stiffness and load vectors for both intermediate and final meshes and the actual solution, are carried out in parallel. This model is fairly easy to implement within existing sequential FEM codes. The parallel 3 step algorithm follows:

*Parallel 3 step Algorithm*

*Step 1.* On all processors, the problem is solved on a sufficiently fine initial mesh and the error is estimated. Convergence rates, if they are not known from a-priori estimates, are computed.
Error vs. CPU time on Intel i860 for Poisson's Problem

Fig. 22. Parallel 3-step algorithm on example problem 1 on iPSC/860.

(recall (32) and (33)). The intermediate \( h \)-adaptive mesh is computed and, using one of the partitioning algorithms, the domain is partitioned into the required number of subdomains. Note that both the solution and partitioning algorithms can be parallelized, however for the relatively coarse initial meshes it is rarely worth the effort.

**Step 2.** Based on the partitioning in the earlier step, the matrices are generated in parallel; i.e. on any processor only the elements associated with the partition assigned to that processor are generated and assembled. Then the problem is solved using the appropriate parallel solver and errors are recomputed and updated in parallel on each processor. Compute the \( h \) adapted mesh for the next step on each processor and partition the new \( h \)-adapted mesh.

**Step 3.** Based on the partitioning in the earlier step, the matrices are generated in parallel, i.e. on any processor only the elements associated with the partition assigned to that processor are generated and assembled. Now the problem is solved using the appropriate parallel solver and errors are recomputed and updated in parallel on each processor.

The first implementation of this algorithm is on an Intel IPSC-860 machine with 16 processors for the model problem. Results obtained using this algorithm with one to four processors are shown in Figs. 22 and 23. Note that the characteristic knee and sharp convergence for the \( p \)-adaptive phase is not seen in Fig. 23 because of the linear \( x \) scale used. This does show the large drop in CPU speed when using 4 processors. Due to the many sequential portions of the algorithm that must be repeated on every processor in the SPMD model, the parallel efficiency is only 60%.

7. Conclusions

The focus of this work is to design an \( hp \)-adaptive strategy to produce good \( hp \) meshes with a minimum number of resolutions carried out on multiprocessor computers. A simple strategy that equidistributes the errors over all the elements, and uses a priori estimates to predict the refinement
required to reduce the error to a desired level locally and globally is presented. This is followed by the presentation of an efficient decomposition procedure and parallel solution as described in [14].

Numerical experiments carried out to test the algorithms indicate good performance for several test cases, even in problems with singularities. Exponential convergence rates of the $hp$ adaptive schemes appear to be attained with only three resolutions for any given target error. Since the strategy is driven by a specific target error, unnecessary refinements are generally avoided. Singularities in the solution do not appear to present additional difficulties in the refinement strategy. The strategy also seems to be fairly successful on more complex problems as has been shown in the preliminary work of Wu, Oden and Legat [15] on Navier-Stokes equations and Bey [2] on hyperbolic systems using a discontinuous Galerkin formulation.

Acknowledgment

The financial support of DARPA under contract no. DABT63-92-C-0042 is gratefully acknowledged. The many helpful discussions with Dr. K. Bey are also acknowledged.

Appendix A: 3-step formulation

For an intermediate mesh which achieves $\theta_i = \eta_i \|U_0\|_1$ with an adaptive $h$ and $p$ held at $p = p_0$, we have

$$\Lambda^2 = p_0^{2p} \sum_{k=1}^{N_1} \frac{\theta_k^2}{h_f^2}$$  \hspace{1cm} (A.1)

where, if the error is equidistributed

$$\theta_f^2 = \frac{\theta_1^2}{N_1}$$  \hspace{1cm} (A.2)
We exploit the idea (which is true only asymptotically as $h_K \to 0$) that the intermediate error $\eta_i$ is equidistributed so that each element $\Omega_K$ in the initial mesh $\Omega_h$ must be replaced by $n_K$ elements each carrying an equal error. Thus,

$$\frac{A_i^2 N_i}{p_0^2 \theta_1^2} = \sum_{K=1}^{N_0} \sum_{j=1}^{n_K} h_j^{-2\mu_0}$$

(A.4)

Here, both $h_j$ and $N_i$ are unknown. There exists for each element in the initial mesh, a mean mesh parameter $h_K$ for each element such that

$$\frac{n_K}{h_K^{-2\mu_0}} = \sum_{j=1}^{n_K} \left( \frac{1}{h_j} \right)^{2\mu_0}$$

(A.5)

Using (A.5) in (A.3)

$$\frac{N_i}{p_0^2 \theta_1^2} A_i^2 = \sum_{K=1}^{N_0} \frac{n_K}{h_K^{-2\mu_0}}$$

(A.6)

or, passing to the element level

$$\frac{N_i}{p_0^2 \theta_1^2} A_K = \frac{n_K}{h_K^{-2\mu_0}}$$

(A.7)

or

$$\frac{h_K^{-2\mu_0}}{n_K} = \frac{p_0^2 \theta_1^2}{A_K^2 N_i}$$

(A.8)

For two dimensions

$$\frac{h_0}{h_K} = \sqrt{n_K}$$

(A.9)

or

$$h_K^{-2\mu_0} = \frac{h_0^{-2\mu_0}}{n_K}$$

(A.10)

Using (A.10) in (A.8), we obtain

$$n_K^{\mu_0+1} = \frac{A_K^2 N_i h_0^{-2\mu_0}}{p_0^2 \theta_1^2}$$

(A.11)

Further

$$\sum_{K=1}^{N_0} n_K = N_i$$

(A.12)

Eqs. (A.11) and (A.12) can be solved iteratively for $n_K$ and $N_i$. Having computed $n_K$ for each element, we obtain a new mesh by subdividing the original elements, assumed here to be quadrilaterals. Suitable grading is obtained by estimating error in partitions of the initial element.

For one and three dimensions, (A.9) must be suitably modified. In one dimension

$$\frac{h_0}{h_K} = n_K$$

(A.13)

and (A.11) now becomes
Similarly, for three dimensions

\[ n_k^{2/3m_0 + 1} = \frac{A_k^2 N h_k^{2m_0}}{\rho_0^{2m_0} \theta_k^1} \]  \hspace{1cm} (A.15)

or, in general

\[ n_k^{m_0 + 1} = \frac{A_k^2 N h_k^{2m_0}}{\rho_0^{2m_0} \theta_k^1} \]  \hspace{1cm} (A.16)

where \( \beta = 2 \) for one dimension, one for two dimensions, \( 2/3 \) for three dimensions.

References

INFORMATION FOR CONTRIBUTORS

Manuscripts should be sent in triplicate to one of the Editors. All manuscripts will be refereed. Manuscripts should preferably be in English. They should be typewritten, double-spaced, first copies (or clear Xerox copies thereof) with a wide margin. Abstracts, footnotes and lists of references should also be double-spaced. All pages should be numbered (also those containing references, tables and figure captions). Upon acceptance of an article, author(s) will be asked to transfer copyright of the article to the publisher. This transfer will ensure the widest possible dissemination of information.

Abstracts
The text of a paper should be preceded by a summary in English. This should be short, but should mention all essential points of the paper.

Figures and tables
The drawings for the figures must be originals, drawn in black India ink in large size and carefully lettered, or printed on a high-quality laser printer. The lettering as well as the details should have proportionate dimensions, so as not to become illegible or unclear after the usual reduction by the printers; in general, the figures should be designed for a reduction factor of two or three. Mathematical symbols should be entered in italics, where appropriate. Each figure should have a number and a caption; the captions should be collected on a separate sheet. The appropriate place of a figure should be indicated in the margin. Tables should be typed on separate sheets. Each table should have a number and a title. The appropriate places for the insertion of tables should be indicated in the margin. Colour illustrations can be included and will be printed at no charge if, in the opinion of the Editors, the colour is essential. If this is not the case, the figures will be printed in black and white unless the author is prepared to pay the extra costs arising from colour reproduction.

Formulate
Displayed formulæ should be numbered and typed or clearly written by hand. Symbols should be identified in the margin, where they occur for the first time.

References
In the text, reference to other parts of the paper should be made by section (or equation) number, but not by page number. References should be listed on a separate sheet in the order in which they appear in the text.

COMPLETE INSTRUCTIONS TO AUTHORS are published in every last issue of a volume, and copies can also be obtained from the Editors and the Publisher, Elsevier Science B.V., P.O. Box 951, 1000 AE Amsterdam, The Netherlands.

INFORMATION FOR SUBSCRIBERS

COMPUTER METHODS IN APPLIED MECHANICS AND ENGINEERING (ISSN 0045-7825) is published in ten volumes (40 issues) a year. For 1995 Volumes 119-128 are scheduled for publication. Subscription prices are available upon request from the publisher. Subscriptions are accepted on a prepaid basis only and are entered on a calendar year basis. Issues are sent by surface mail except to the following countries where airmail charges are available upon request. Claims for missing issues will be honoured free of charge within six months after the publication date of issues. Mail orders and inquiries to: Elsevier Science S.A., P.O. Box 564, CH-1001 Lausanne, Switzerland.

INFORMATION FOR ADVERTISERS

Advertising orders and inquiries can be sent to the Advertising Manager, Elsevier Science B.V., Journal Division, P.O. Box 211, 1000 AE Amsterdam, The Netherlands.

Special regulations for authors
Upon acceptance of an article by the journal, the author(s) will be asked to transfer copyright of the article to the publisher. This transfer will ensure the widest possible dissemination of information.

Special regulations for readers in the USA
This journal has been registered with the Copyright Clearance Center, Inc. Consent is given for copying of articles for personal or internal use, or for the personal use of specific clients. This consent is given on the condition that the copier pays through the Center the per-copy fee stated in the code on the first page of each article for copying beyond that permitted by Sections 107 or 108 of the US Copyright Law. The appropriate fee should be forwarded with a copy of the first page of the article to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, USA.

If no code appears in an article, the author has not given broad consent to copy and permission to copy must be obtained directly from the author. The fees indicated on the first page of an article in this issue will apply retroactively to all articles published in the journal, regardless of the year of publication. This consent does not extend to other kinds of copying, such as for general distribution, resale, advertising and promotion purposes, or for creating new collective works. Special written permission must be obtained from the publisher for such copying.