Domain Decomposition and hp-Adaptive Finite Elements

Randolph E. Bank [*1] and Hieu Nguyen [†2]

[1] Department of Mathematics, University of California, San Diego, La Jolla, CA 92093-0112, USA, rbank@ucsd.edu.
[2] Department of Mathematics, University of California, San Diego, La Jolla, CA 92093-0112, USA, htn005@math.ucsd.edu

1 Introduction

In this work, we report on an ongoing project to implement an hp-adaptive finite element method. The inspiration of this work came from the development of certain a posteriori error estimates for high order finite elements based on superconvergence [7, 8, 9]. We wanted to create an environment where these estimates could be evaluated in terms of their ability to estimate global errors for a wide range of problems, and to be used as the basis for adaptive enrichment algorithms.

Their use in a traditional h-refinement scheme for fixed degree p is straightforward, as is their use for mesh smoothing, again with fixed p. What is less clear and thus more interesting is their use in a traditional adaptive p-refinement scheme. One issue we hope to resolve, at least empirically, is the extent to which the superconvergence forming the foundation of these estimates continues to hold on meshes of variable degree. If superconvergence fails to hold globally (for example, in our preliminary experiments, superconvergence seems to hold in the interiors of regions of constant p but fails to hold along interfaces separating elements of different degrees), we would still like to determine if they remain robust enough to form the basis of an adaptive p-refinement algorithm.

As this is written, we have implemented in the PLTMG package [2] adaptive h-refinement/coarsening, adaptive p-refinement/coarsening, and adaptive mesh smoothing. These three procedures can be used separately, or mixed in arbitrary combinations. For example, one could compose an adaptive algorithm consisting of alternating steps of h and p-refinement. Since this requires that all procedures are able to process meshes with both variable h and p, many of the internal data structures

[*] The work of this author was supported by the U.S. National Science Foundation under contract DMS-0915220. The Beowulf cluster used for the numerical experiments was funded by NSF SCREMS-0619173.
[†] The work of this author was supported in part by a grant from the Vietnam Education Foundation (VEF).

and existing algorithms in the PLTMG package had to be generalized and extended. However, at present there remains open the more delicate and challenging issue of hp-refinement; that is, how to use these error estimates to decide if it is better to refine a given element into several child elements (h-refinement), or increase its degree (p-refinement). We hope to be able to report progress on this point at some time in the future.

Since PLTMG has options for parallel adaptive enrichment, this aspect also needs to be addressed. Fortunately, the parallel adaptive meshing paradigm implemented in PLTMG, see [1, 3, 4], formally works as well for p and hp-adaptivity as it does for h-adaptivity for which it was originally developed. As its final step, the paradigm requires the solution of a large global set of equations. A special DD algorithm (see [5, 6]) taking advantage of the structure of the parallel adaptive procedure was developed for this purpose.

2 A Posteriori Error Estimate

In the case of two dimensions, we consider an element t with vertices ν_i, and edges e_i, $1 \le i \le 3$, with e_i opposite ν_i. Let h_t denote the diameter of t. The barycentric coordinates for element t are denoted c_i, $1 \le i \le 3$. The restriction of the $\mathcal{C}^0$ piecewise polynomial space of degree $p \ge 1$ to element t consists of the $(p+1)(p+2)/2$-dimensional space $\mathcal{P}_p$ of polynomials of degree p, with degrees of freedom given by nodal values at the barycentric coordinates

$$(c_1, c_2, c_3) = (j/p, k/p, (p-j-k)/p)$$

for $0 \le j \le p$, $0 \le k \le p-j$.

Superconvergent derivative recovery schemes for this family of elements were developed in [7, 8, 9]. For the continuous piecewise polynomial space of degree p, let $\partial^p u_h$ denote any of the (discontinuous piecewise constant) p-th derivatives. The recovered p-th derivative is denoted by $R(\partial^p u_h) \equiv S^m Q(\partial^p u_h)$. Here Q is the L^2 projection from discontinuous piecewise constants into the space of continuous piecewise linear polynomials, and S is a multigrid smoother for the Laplace operator; m is a small integer, typically one or two. Under appropriate smoothness assumptions, it was shown that $\|\partial^p u - R(\partial^p u_h)\|$ has better than the first order convergence of $\|\partial^p (u - u_h)\|$.

To describe our a posteriori estimate for the case of an element of degree p, we write

$$\mathcal{P}_{p+1}(t) = \mathcal{P}_p(t) \oplus \mathcal{E}_{p+1}(t)$$

where the hierarchical extension $\mathcal{E}_{p+1}(t)$ consists of those polynomials in $\mathcal{P}_{p+1}(t)$ that are zero at all degrees of freedom associated with $\mathcal{P}_p(t)$. In the case of two dimensions, this is a subspace of dimension $p+2$, with a convenient basis given by

$$\psi_{p+1,k} = \prod_{j=0}^{k-1} (c_1 - j/p) \prod_{m=0}^{p-k} (c_2 - m/p)$$

for $0 \le k \le p+1$. Using this basis, we approximate the error $u - u_{h,p}$ on element t as

$$u - u_{h,p} \approx e_{h,p} \equiv \alpha_t \sum_{k=0}^{p+1} \frac{\partial_{c_1}^k \partial_{c_2}^{p+1-k} \hat{u}}{k!(p+1-k)!} \psi_{p+1,k}. \tag{1}$$

The partial derivatives of order $p+1$ appearing in (1) are formally $O(h_t^{p+1})$ when expressed in terms of ∂_x and ∂_y. The derivative $\partial_x^k \partial_y^{p+1-k} \hat{u}$ is constant on element t, computed by differentiating the recovered p-th derivatives of u_h, which are linear polynomials on element t.

$$\partial_x^k \partial_y^{p+1-k} \hat{u} = \begin{cases} \partial_y R(\partial_y^p u_h), & k = 0, \\ (\partial_x R(\partial_x^{k-1} \partial_y^{p+1-k} u_h) + \partial_y R(\partial_x^k \partial_y^{p-k} u_h))/2, & 1 \le k \le p, \\ \partial_x R(\partial_x^p u_h), & k = p+1. \end{cases}$$

The constant α_t is chosen such that

$$\sum_{k=0}^{p} \|\partial_x^k \partial_y^{p-k} e_{h,p}\|_t^2 = \sum_{k=0}^{p} \|\partial_x^k \partial_y^{p-k} u_h - R(\partial_x^k \partial_y^{p-k} u_h)\|_t^2$$

Normally, one should expect $\alpha_t \approx 1$, except for elements where the true solution u is not smooth enough to support p derivatives.

3 Basis Functions

One aspect of our study that is a bit unconventional is our use of nodal basis functions, rather than a hierarchical family of functions. The standard element of degree p uses standard nodal basis functions, as illustrated in Fig. 1, left. Along edges shared by elements of different degrees, the element of lower degree inherits the degrees of freedom of the higher degree element. This results in elements of degree p with one or two *transition edges* of higher degree. Some typical cases are illustrated in Fig. 1.

To illustrate the construction of the nodal basis for transition elements, consider the case of an element t of degree p with one transition edge of degree $p+1$. Without loss of generality take this to be edge three. We define one special polynomial of degree $p+1$, zero at all nodes of the standard element of degree p, and identically zero on edges one and two, by

$$\tilde{\phi}_{p+1} = \begin{cases} \prod_{k=0}^{(p-1)/2} (c_1 - k/p)(c_2 - k/p), & \text{for } p \text{ odd}, \\ (c_1 - c_2) \prod_{k=0}^{(p-2)/2} (c_1 - k/p)(c_2 - k/p), & \text{for } p \text{ even}. \end{cases}$$

The polynomial space for the transition element is given by $\mathcal{P}_p \oplus \{\tilde{\phi}_{p+1}\}$. We form linear combinations of $\tilde{\phi}_{p+1}$ and the $p+1$ standard nodal basis functions associated

with edge three to form the $p+2$ nodal basis functions for the transition edge. Because each of these $p+2$ polynomials is zero on edges one and two, and zero at all internal nodes for element t, all linear combinations of them also satisfy these properties, so the required calculation effectively reduces to a simple one-dimensional change of basis. If the edge is of degree $p+k$, the polynomial space is given by $\mathcal{P}_p \oplus \{\tilde{\phi}_{p+1}(c_1 - c_2)^m\}_{m=0}^{k-1}$, and a similar construction yields the required nodal basis for the transition edge. If a second transition edge is present, it is treated analogously. Because of our construction, each transition edge can be treated independently. It is also easy to see that the global finite element space constructed in this fashion is $\mathcal{C}^0$.

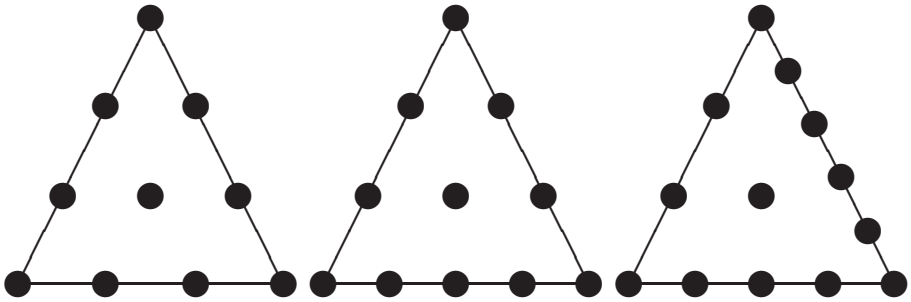

Fig. 1. A standard cubic element (*left*), a cubic element with one quartic edge (*middle*) and a cubic element with one quartic and one quintic edge (*right*).

4 Parallel Adaptive Algorithm

A general approach to parallel adaptive discretization for systems of elliptic partial differential equations was introduced in [3, 4]. This approach was motivated by the desire to keep communications costs low, and to allow sequential adaptive software such as PLTMG to be employed without extensive recoding.

The original paradigm has three main components:

Step I: Load Balancing. We solve a small problem on a coarse mesh, and use a posteriori error estimates to partition the mesh. Each subregion has approximately the same error, although subregions may vary considerably in terms of numbers of elements, or polynomial degree.

Step II: Adaptive Meshing. Each processor is provided the complete coarse problem and instructed to sequentially solve the *entire* problem, with the stipulation that its adaptive enrichment (h or p) should be limited largely to its own partition. The target number of degrees of freedom for each processor is the same. At the end of this step, the mesh is regularized such that the global finite element space described in Step III is conforming.

Step III: Global Solve. The final global problem consists of the union of the refined partitions provided by each processor. A final solution is computed using domain decomposition.

A variant of the above approach, in which the load balancing occurs on a much finer space, was described in [1]. The motivation was to address some possible problems arising from the use of a coarse grid in computing the load balance. This variant also has three main components.

Step I: Load Balancing. On a single processor we adaptively create a *fine* space of size N_P, and use a posteriori error estimates to partition the mesh such that each subregion has approximately equal error, similar to Step I of the original paradigm.
Step II: Adaptive Meshing. Each processor is provided the complete adaptive mesh and instructed to sequentially solve the *entire* problem. However, in this case each processor should adaptively *coarsen* regions corresponding to other processors, and adaptively enrich its own subregion. The size of the problem on each processor remains N_P, but this adaptive rezoning strategy concentrates the degrees of freedom in the processor's subregion. At the end of this step, the global space is made conforming as in the original paradigm.
Step III: Global Solve. This step is the same as in the original paradigm.

Using the variant, the initial mesh can be of any size. Indeed, our choice of N_P is mainly for convenience and to simplify notation; any combination of coarsening and refinement could be allowed in Step II.

5 DD Solver

Let $\Omega = \cup_{i=1}^{P} \Omega_i \subset \mathcal{R}^2$ denote the domain, decomposed into P geometrically conforming subdomains. Let Γ denote the interface system. The degree of a vertex x lying on Γ is the number of subregions for which $x \in \bar{\Omega}_i$. A cross point is a vertex $x \in \Gamma$ with degree$(x) \geq 3$. We assume that the maximal degree at cross points is bounded by the constant δ_0. The connectivity of Ω_i is the number of other regions Ω_j for which $\bar{\Omega}_i \cap \bar{\Omega}_j \neq \emptyset$. We assume the connectivity of Ω_i is bounded by the constant δ_1.

In our algorithm, we employ several triangulations. The mesh $\mathcal{T}$ is a globally refined, shape regular, and conforming mesh of size h. We assume that the fine mesh $\mathcal{T}$ is aligned with the interface system Γ. The triangulations $\mathcal{T}^i \subset \mathcal{T}$, $1 \leq i \leq P$ are partially refined triangulations; they coincide with the fine triangulation $\mathcal{T}$ within Ω_i, but are generally much coarser elsewhere, although as in the case for the variant paradigm, along the interface system Γ, $\mathcal{T}^i$ may have some intermediate level of refinement.

Let $\mathcal{S}$ denote the hp space of piecewise polynomials, associated with the triangulation $\mathcal{T}$, that are continuous in each of the Ω_i, but can be discontinuous along the interface system Γ. Let $\bar{\mathcal{S}} \subset \mathcal{S}$ denote the subspace of globally continuous piecewise polynomials. The usual basis for $\mathcal{S}$ is just the union of the nodal basis functions

corresponding to each of the subdomains Ω_i; such basis functions have their support in $\bar{\Omega}_i$ and those associated with nodes on Γ will have a jump at the interface. In our discussion, we will have occasion to consider another basis, allowing us to write $\mathcal{S} = \bar{\mathcal{S}} \oplus \mathcal{X}$, where $\mathcal{X}$ is a subspace associated exclusively with jumps on Γ. In particular, we will use the global conforming nodal basis for the space $\bar{\mathcal{S}}$, and construct a basis for $\mathcal{X}$ as follows. Let z_k be a node lying on Γ shared by two regions Ω_i and Ω_j (for now, z_k is not a crosspoint). Let $\phi_{i,k}$ and $\phi_{j,k}$ denote the usual nodal basis functions corresponding to z_k in Ω_i and Ω_j, respectively. The continuous nodal basis function for z_k in $\bar{\mathcal{S}}$ is $\phi_k \equiv \phi_{i,k} + \phi_{j,k}$, and the "jump" basis function in $\mathcal{X}$ is $\hat{\phi}_k \equiv \phi_{i,k} - \phi_{j,k}$. The direction of the jump is arbitrary at each z_k, but once chosen, will be used consistently. In this example, at point z_k we will refer to i and the "master" index and j as the "slave" index. At a cross point where $\ell > 2$ subregions meet, there will be one nodal basis function corresponding to $\bar{\mathcal{S}}$ and $\ell - 1$ jump basis functions. These are constructed by choosing one master index for the point, and making the other $\ell - 1$ indices slaves. We can construct $\ell - 1$ basis functions for $\mathcal{X}$ as $\phi_{i,k} - \phi_{j,k}$, where i is the master index and j is one of the slave indices.

For each of the triangulations $\mathcal{T}^i$, $1 \le i \le P$ we have a global nonconforming subspace $\mathcal{S}^i \subset \mathcal{S}$, and global conforming subspace $\bar{\mathcal{S}}^i \subset \bar{\mathcal{S}}$. In a fashion similar to $\mathcal{S}$, we have $\mathcal{S}^i = \bar{\mathcal{S}}^i \oplus \mathcal{X}^i$.

For simplicity, let the continuous variational problem be: find $u \in \mathcal{H}^1(\Omega)$ such that

$$a(u, v) = (f, v) \tag{2}$$

for all $v \in \mathcal{H}^1(\Omega)$, where $a(u, v)$ is a self-adjoint, positive definite bilinear form corresponding to the weak form of an elliptic partial differential equation, and $|\!|\!| u |\!|\!|_\Omega^2 = a(u, u)$ is comparable to the usual $\mathcal{H}^1(\Omega)$ norm.

To deal with the nonconforming nature of $\mathcal{S}$, for $u, v \in \mathcal{S}$, we decompose $a(u, v) = \sum_{i=1}^{P} a_{\Omega_i}(u, v)$. For each node z lying on Γ there is one master index and $\ell - 1 > 0$ slave indices. The total number of slave indices is denoted by K, so the total number of constraint equations in our nonconforming method is K. To simplify notation, for each $1 \le j \le K$, let $m(j)$ denote the corresponding master index, and z_j the corresponding node. We define the bilinear form $b(v, \lambda)$ by

$$b(v, \lambda) = \sum_{j=1}^{K} \{v_{m(j)} - v_j\} \lambda_j \tag{3}$$

where $\lambda \in \mathcal{R}^K$. In words, $b(\cdot, \cdot)$ measures the jump between the master value and each of the slave values at each node on Γ. The nonconforming variational formulation of (2) is: find $u_h \in \mathcal{S}$ such that

$$\begin{aligned} a(u_h, v) + b(v, \lambda) &= (f, v) \\ b(u_h, \xi) \qquad\quad &= 0 \end{aligned} \tag{4}$$

for all $v \in \mathcal{S}$ and $\xi \in \mathcal{R}^K$. Although this is formally a saddle point problem, the constraints are very simple; in particular, (4) simply imposes continuity at each of

the nodes lying on Γ, which in turn, implies that $u_h \in \bar{\mathcal{S}}$. Thus u_h also solves the reduced and conforming variational problem: find $u_h \in \bar{\mathcal{S}}$ such that

$$a(u_h, v) = (f, v)$$

for all $v \in \bar{\mathcal{S}}$.

Let $\mathcal{K}_i$ denote the index set of constraint equations in (3) that correspond to nodes present in $\mathcal{T}^i$. Then

$$b_i(v, \lambda) = \sum_{j \in \mathcal{K}_i} \{v_{m(j)} - v_j\} \lambda_j.$$

We are now in a position to formulate our domain decomposition algorithm. Our initial guess $u_0 \in \mathcal{S}$ is generated as follows: for $1 \le i \le P$, we find (in parallel) $u_{0,i} \in \bar{\mathcal{S}}^i$ satisfying

$$a(u_{0,i}, v) = (f, v) \tag{5}$$

for all $v \in \bar{\mathcal{S}}^i$. Here we assume exact solution of these local problems; in practice, these are often solved approximately using iteration. The initial guess $u_0 \in \mathcal{S}$ is composed by taking the part of $u_{0,i}$ corresponding to the fine subregion Ω_i for each i. In particular, let χ_i be the characteristic function for the subregion Ω_i. Then

$$u_0 = \sum_{i=1}^{P} \chi_i u_{0,i}$$

To compute $u_{k+1} \in \mathcal{S}$ from $u_k \in \mathcal{S}$, we solve (in parallel): for $1 \le i \le P$, find $e_{k,i} \in \mathcal{S}^i$ and $\lambda_{k,i} \in \mathcal{R}^K$ such that

$$\begin{aligned} a(e_{k,i}, v) + b_i(v, \lambda_{k,i}) &= (f, v) - a(u_k, v) \\ b_i(e_{k,i}, \xi) &= -b_i(u_k, \xi) \end{aligned} \tag{6}$$

for all $v \in \mathcal{S}^i$ and $\xi \in \mathcal{R}^K$. We then form

$$u_{k+1} = u_k + \sum_{i=1}^{P} \chi_i e_{k,i}.$$

Although the iterates u_k are elements of the nonconforming space $\mathcal{S}$, the limit function $u_\infty = u_h \in \bar{\mathcal{S}}$. In some sense, the purpose of the iteration is to drive the jumps in the approximate solution u_k to zero. Also, although (6) suggests a saddle point problem needs to be solved, by recognizing that only $\chi_i e_{k,i}$ is actually used, one can reduce (6) to a positive definite problem of the form (5). In particular, the Lagrange multipliers $\lambda_{k,i}$ need not be computed or updated.

The information required to be communicated among the processors is only the solution values and the residuals for nodes lying on Γ, which is necessary to compute the right hand sides of (6). This requires one all-to-all communication step at the beginning of each DD iteration.

6 Numerical Results

In this section, we present some numerical results. Our examples were run on a LINUX-based Beowulf cluster, consisting of 38 nodes, each with two quad core Xeon processors (2.33 GHz) and 16 GB of memory. The communication network is a gigabit Ethernet switch. This cluster runs the NPACI ROCKS version of LINUX and employs MPICH2 as its MPI implementation. The computational kernels of PLTMG [2] are written in FORTRAN; the *gfortran* compiler was used in these experiments, invoked using the script *mpif90* and optimization flag *-O*.

In these experiments, we used PLTMG to solve the boundary value problem

$$\begin{aligned} -\Delta u &= 1 \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned}$$

where Ω is a domain shaped like Lake Superior.

In our first experiment, the variant strategy was employed. A mesh of N_P degrees of freedom was created on a single processor using h-adaptive and p-adaptive refinement. Elements on this mesh had different sizes and degrees. This mesh was then broadcast to P processors, where a strategy of combined coarsening and refinement in both h and p was used to transfer approximately $N_P/2$ degrees of freedom from outside Ω_i to inside Ω_i. The global fine mesh was then made h-conforming (geometrically conforming) as described in [3, 4] and p-conforming (degrees agree on shared edges along the interface Γ). Note that the adaptive strategies implemented in PLTMG allow mesh moving and other modifications that yield meshes $\mathcal{T}_i$ that generally are *not* submeshes of the global conforming mesh $\mathcal{T}$ (by definition they are identical on Ω_i and $\partial\Omega_i$). However, PLTMG does insure that the partitions remain geometrically conforming, even in the coarse parts of the domain, and in particular, that the vertices on the interface system in each $\mathcal{T}_i$ are a subset of the vertices of interface system of the global mesh $\mathcal{T}$.

In this experiment, three values of N_P (400, 600, and 800 K), and eight values of P (2^k, $1 \le k \le 8$) were used, yielding global fine meshes ranging in size from about 626 K to 96.5 M unknowns. Because our cluster had only 38 nodes, for larger values of P, we simulated the behavior of a larger cluster in the usual way, by allowing nodes to have multiple processes.

In these experiments, the convergence criterion was

$$\frac{\|\delta\mathcal{U}^k\|_G}{\|\mathcal{U}^k\|_G} \le \frac{\|\delta\mathcal{U}^0\|_G}{\|\mathcal{U}^0\|_G} \times 10^{-3}. \tag{7}$$

This is more stringent than necessary for purposes of computing an approximation to the solution of the partial differential equation, but it allows us to illustrate the behavior of the solver as an iterative method for solving linear systems of equations.

Table 1 summarizes this computation. The columns labeled DD indicate the number of domain decomposition iterations required to satisfy the convergence criteria (7). For comparison, the number of iterations needed to satisfy the actual convergence criterion used in PLTMG, based on reducing the error in the solution of the

linear system to the level of the underlying approximation error, is given in parentheses. From these results it is clear that the number of iterations is stable and largely independent of N and P over this range of values. The size of the global mesh for the variant strategy can be estimated from the formula

$$N \approx \theta P N_P + N_P \tag{8}$$

where $\theta = 1/2$. Equation (8) predicts an upper bound, as it does not account for refinement outside of Ω_i and coarsening inside Ω_i, needed to keep the mesh conforming and for other reasons. For $N_P = 800\,\mathrm{K}$, $P = 256$, (8) predicts $N \approx 103,200,000$, where the observed $N = 96{,}490{,}683$.

Table 1. Convergence results for variant algorithm. Numbers of iterations needed to satisfy (7) are given in the column labeled DD. The numbers in parentheses are the number of iterations required to satisfy the actual convergence criterion used by PLTMG.

	$N_P = 400$ K		$N_P = 600$ K		$N_P = 800$ K	
P	N	DD	N	DD	N	DD
2	625,949	10 (3)	776,381	8 (3)	1,390,124	12 (4)
4	1,189,527	13 (4)	1,790,918	11 (4)	2,288,587	9 (3)
8	1,996,139	10 (4)	2,990,807	13 (4)	3,993,126	10 (3)
16	3,569,375	14 (4)	5,220,706	13 (4)	6,920,269	12 (3)
32	6,723,697	13 (3)	9,736,798	16 (4)	13,142,670	11 (3)
64	12,978,568	11 (4)	18,905,909	14 (4)	25,326,662	11 (3)
128	25,155,124	12 (3)	37,148,571	10 (4)	48,841,965	10 (3)
256	48,874,991	11 (3)	72,902,698	14 (4)	96,490,683	11 (3)

In our second experiment we solved the same problem using the original paradigm. On one processor, an adaptive mesh of size $N_c = 50\,\mathrm{K}$ was created. All elements on this mesh were linear elements. This mesh was then partitioned into P subregions, $P = 2^k$, $1 \le k \le 8$. This coarse mesh was broadcast to P processors (simulated as needed) and each processor continued the adaptive process in both h and p, creating a mesh of size N_P. In this experiment, N_P was chosen to be 400, 600, and $800\,\mathrm{K}$. This resulted in global meshes varying in size from approximately $750\,\mathrm{K}$ to $189\,\mathrm{M}$. These global meshes were regularized to be h-conforming and p-conforming, and a global DD solve was made as in the first experiment. As in the first experiment, the usual convergence criteria was replaced by (7) in order to illustrate the dependence of the convergence rate on N and P. The results are summarized in Table 2.

For the original paradigm the size of the global mesh is predicted by

$$N \approx P N_P - (P-1) N_c. \tag{9}$$

Similar to Eq. (8), Eq. (9) only predicts an upper bound, as it does not account for refinement outside of Ω_i, needed to keep the mesh conforming and for other reasons. For example, for $N_c = 50\,\mathrm{K}$, $N_P = 800\,\mathrm{K}$, $P = 256$, (9) predicts

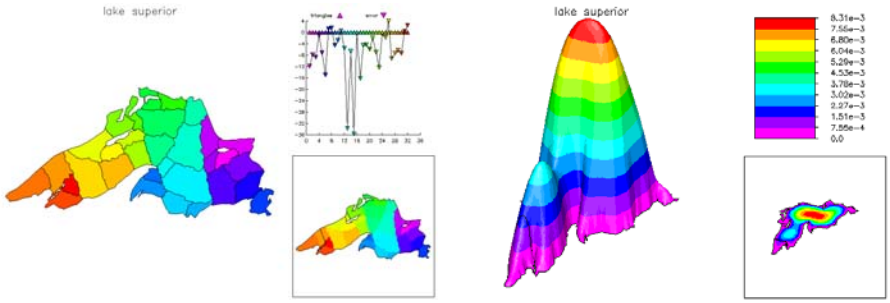

Fig. 2. The load balance (*left*) and solution (*right*) in the case $N_P = 800\,\mathrm{K}$, $P = 32$.

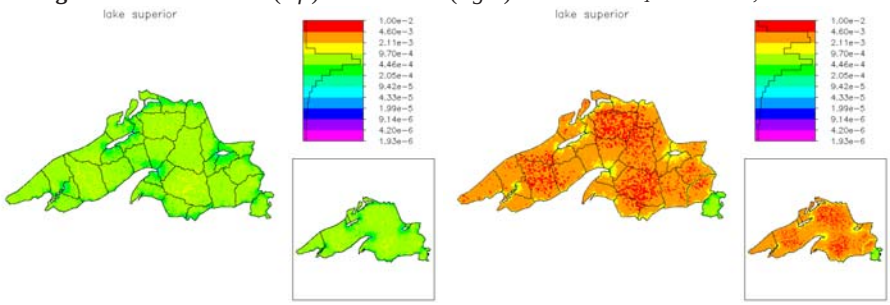

Fig. 3. The mesh density for the global mesh (*left*) and for one of the local meshes (*right*) in the case $N_P = 800\,\mathrm{K}$, $P = 32$.

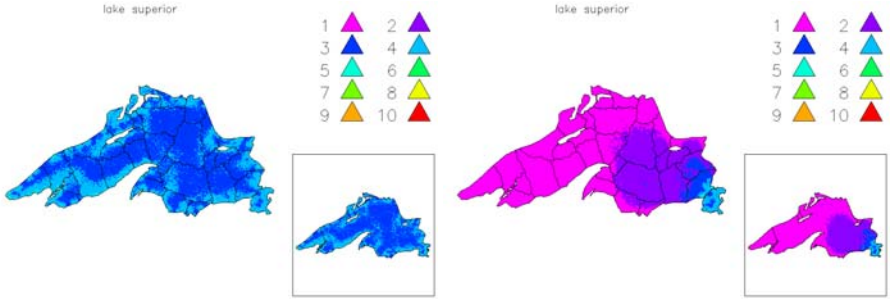

Fig. 4. The degree density for the global mesh (*left*) and for one of the local meshes (*right*) in the case $N_P = 800\,\mathrm{K}$, $P = 32$.

$N \approx 192,050,000$ when actually $N = 189,363,322$. For the case $N_P = 800\,\mathrm{K}$, $P = 32$, the solution and the load balance is shown in Fig. 2. The mesh density and degree density of the global mesh and one local mesh are shown in Figs. 3 and 4. As expected, both the mesh density and the degree density are high in the local region and much lower elsewhere in the local mesh.

Table 2. Convergence results for original Algorithm. Numbers of iterations needed to satisfy (7) are given in the column labeled DD. The numbers in parentheses are the number of iterations required to satisfy the actual convergence criterion used by PLTMG.

	$N_P = 400\,\mathrm{K}$		$N_P = 600\,\mathrm{K}$		$N_P = 800\,\mathrm{K}$	
P	N	DD	N	DD	N	DD
2	750,225	13 (4)	1,150,106	13 (4)	1,549,915	13 (4)
4	1,450,054	13 (4)	2,248,841	13 (4)	3,047,906	13 (4)
8	2,846,963	9 (3)	4,442,665	9 (4)	6,039,743	9 (3)
16	5,635,327	11 (4)	8,821,463	10 (4)	12,010,188	11 (4)
32	11,204,214	12 (4)	17,564,640	10 (4)	23,930,867	11 (4)
64	22,301,910	14 (4)	34,983,543	13 (4)	47,693,190	13 (4)
128	44,408,605	11 (4)	69,696,605	12 (4)	95,026,759	11 (4)
256	88,369,503	11 (3)	138,790,801	11 (3)	189,363,322	11 (4)

References

1. R.E. Bank. Some variants of the Bank-Holst parallel adaptive meshing paradigm. *Comput. Vis. Sci.*, 9(3):133–144, 2006. ISSN 1432-9360. URL http://dx.doi.org/10.1007/s00791-006-0029-6.
2. R.E. Bank. PLTMG: A software package for solving elliptic partial differential equations, users' guide 10.0. Technical Report, Department of Mathematics, University of California at San Diego, 2007. URL http://ccom.ucsd.edu/~reb.
3. R.E. Bank and M. Holst. A new paradigm for parallel adaptive meshing algorithms. *SIAM J. Sci. Comput.*, 22(4):1411–1443 (electronic), 2000. ISSN 1064-8275. URL http://dx.doi.org/10.1137/S1064827599353701.
4. R.E. Bank and M. Holst. A new paradigm for parallel adaptive meshing algorithms. *SIAM Rev.*, 45(2):291–323 (electronic), 2003. ISSN 0036-1445. URL http://dx.doi.org/10.1137/S003614450342061. Reprinted from SIAM J. Sci. Comput. 22 (2000), no. 4, 1411–1443 [MR1797889].
5. R.E. Bank and S. Lu. A domain decomposition solver for a parallel adaptive meshing paradigm. *SIAM J. Sci. Comput.*, 26(1):105–127 (electronic), 2004. ISSN 1064-8275. URL http://dx.doi.org/10.1137/S1064827503428096.
6. R.E. Bank and P.S. Vassilevski. Convergence analysis of a domain decomposition paradigm. *Comput. Vis. Sci.*, 11(4–6):333–350, 2008. ISSN 1432-9360. URL http://dx.doi.org/10.1007/s00791-008-0103-3.
7. R.E. Bank and J. Xu. Asymptotically exact a posteriori error estimators. I. Grids with superconvergence. *SIAM J. Numer. Anal.*, 41(6):2294–2312 (electronic), 2003. ISSN 0036-1429. URL http://dx.doi.org/10.1137/S003614290139874X.
8. R.E. Bank and J. Xu. Asymptotically exact a posteriori error estimators. II. General unstructured grids. *SIAM J. Numer. Anal.*, 41(6):2313–2332 (electronic), 2003. ISSN 0036-1429. URL http://dx.doi.org/10.1137/S0036142901398751.
9. R.E. Bank, J. Xu, and B. Zheng. Superconvergent derivative recovery for Lagrange triangular elements of degree p on unstructured grids. *SIAM J. Numer. Anal.*, 45(5):2032–2046 (electronic), 2007. ISSN 0036-1429. URL http://dx.doi.org/10.1137/060675174.