A Parallel *hp*-Adaptive Finite Element Method

Randolph E. Bank and Hieu Nguyen

Abstract. This paper describes a combination of automatic *hp*-adaptive finite elements and domain decomposition. The combination is based on the Bank-Holst parallel adaptive meshing paradigm. The *hp*-adaptivity is based on a derivative recovery technique while the domain decomposition method is formulated using a mortar-like formulation with Dirac δ functions for the mortar element space. Numerical results show that the approach can scale to hundreds of processors and the convergence of the domain decomposition method is independent of the number of processors as well as the distribution of element sizes and element degrees among the local meshes.

1. Introduction

hp-adaptive finite elements and domain decomposition are popular methods for numerically solving PDEs. The former is known for its fast rate of convergence and the latter is recognized as an efficient way to solve large problems using parallel computation. In the software package $PLTMG$ [Ban11], we combine these two methods to take advantages of their strong points. In this paper we describe some of the challenges in forming an effective combination, based on extensions of the Bank-Holst paradigm [BH00, BH03, Ban06b].

In Bank-Holst paradigm, the domain is partitioned according to a posteriori error estimates obtained on a coarse mesh of the whole domain solved on a single processor. In this partition, each subdomain has approximately equal error even though they can greatly vary in size, number of elements and degrees of freedom. At the conclusion of the adaptive *hp* refinement, each subdomain will have (approximately) the same number of degrees of freedom. This is achieved by giving each processor the same target number of degrees of freedom and allowing them to perform adaptive meshing independently (with focus on their own subdomain). The domain decomposition solver is employed only when adaptive enrichment has concluded.

1991 *Mathematics Subject Classification.* 65N50, 65N55.

Key words and phrases. hp adaptivity, *hp*-FEM, domain decomposition, Bank-Holst paradigm.

The work of this author was supported by the National Science Foundation under contract DMS-0915220.

The work of this author was supported by the National Science Foundation under contract DMS-0915220 and a grant from the Vietnam Education Foundation (VEF)..

The *hp*-finite element method in this work is based on gradient/derivative recovery technique introduced in [BX03a, BX03b, BXZ07]. High order derivatives of the exact solution are recovered by superconvergent approximations. These approximations are used to formulate a posteriori error estimates that guide the adaptive meshing. The approach is also able to address a crucial question in *hp*-adaptivity; that is whether it is more advantageous to refine a given element into several child elements (h-refinement) or increase its degree. A superconvergent result is used as a consistency check of the a posteriori error estimates. In the region where the exact solution is smooth enough, the result should be consistent. In the regions near singularities and low regularity, we anticipate that the recovery scheme will have difficulties approximating high order derivatives of the exact solution and the superconvergent result is no longer valid. Consequently, the consistency check can be used to identify regions near singularities regions with rapid changes, and signal the adaptive procedure to use h-refinement in these regions and p-refinement elsewhere.

2. *hp*-Finite Element

2.1. Basis functions. In our study, we use nodal basis functions, rather than a more traditional hierarchical family of functions as in [MM11]. For a standard element of degree p, basis functions are defined by their values at nodal points of degree p (equals 1 at the associated nodal point and 0 at all the others) as illustrated in Figure 1 (left).

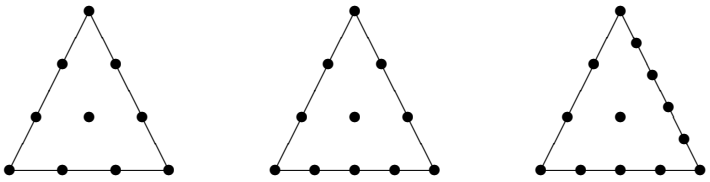

FIGURE 1. A standard cubic element (left), a cubic element with one quartic edge (middle) and a cubic element with one quartic and one quintic edge (right).

Along edges shared by elements of different degrees, the element of the 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 Figure 1. Special treatment is utilized for basis functions associated with the transition edges. A transition basis function of degree $p + k$ is sought in the form of a linear combination of the standard basis functions of degree p associated with the transition edge and special polynomials of degree $p + 1, p + 2, \ldots, p + k$. These special polynomials can be chosen to be ones that equal zero at all of nodal points of degree p (see [Ngu10, BN12])

This construction of basis functions guarantees there are no hanging nodes and the global finite element space is continuous. In addition, the construction can be generalized to three dimensions (see the Appendix in [BN12]).

2.2. Error estimates. Let Ω be the domain of the PDE and $\mathcal{T}_h$ be a shape regular triangulation of Ω of size h. Denote by $\mathcal{V}_h^{(p)}$ the space of continuous piecewise polynomials of degree p associated with $\mathcal{T}_h$. Define Q be the L^2 projection from the space of discontinuous piecewise constants functions associated with $\mathcal{T}_h$ into the space of continuous piecewise linear polynomials $\mathcal{V}_h^{(1)}$. Finally, let S be a multigrid smoother for the Laplace operator and m be a small integer, typically one or two.

Our error estimate is inspired by the work on gradient/derivative recovery techniques of Bank, Xu, and Zheng ([BX03a, BX03b, BXZ07]), where it is shown that $\|\partial^p(u-S^mQu_{hp})\|_\Omega$ is an asymptotically exact approximation of $\|\partial^p(u-u_{hp})\|_\Omega$ with better than first order of accuracy. In addition, $\partial(S^mQ(\partial^p u_{hp}))$ is proved to be a superconvergent approximation of $\partial^{p+1}u$. Here u is the exact solution and u_{hp} is the finite element solution of the PDE.

To describe our a posteriori estimate for the case of element τ of degree p, let $\mathcal{P}_p(\tau)$ be the space of polynomials of degree p defined on τ. We write

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

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

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

where c_i is the i-th barrycentric coordinate function. Using this basis, we approximate $u - u_{hp}$ on element τ as

$$u - u_{hp}|_\tau \approx \epsilon_\tau = \alpha_\tau \sum_{k=0}^{p+1} \frac{\partial_{c_1}^k \partial_{c_2}^{p+1-k}\hat{u}}{k!(p+1-k)!}\psi_{p+1,k} \tag{2.1}$$

where $\hat{u}$ is a hierarchical extension of degree $p+1$ of u_{hp}.

The partial derivatives of order $p+1$ appearing in (2.1) 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 τ, computed by differentiating the recovered p-th derivatives of u_{hp}, which are linear polynomials on element τ.

The constant α_τ is chosen such that

$$\|\partial^p(u - S^mQu_{hp})\|_{0,\tau} = \|\partial^p(\epsilon_\tau)\|_{0,\tau}.$$

Normally, one should expect $\alpha_\tau \approx 1$, except for elements where the true solution u is not smooth enough to support p derivatives. Therefore, the size of α_τ can be used to formulate *hp*-refinement indicator that decide whether element τ should be refined in h or in p. Then our *local error indicator* is defined by $\eta_\tau = |\epsilon_\tau|_{1,\tau}$. For more details on our formulation of a posteriori error estimates and *hp* refinement indicator, see [BN12].

2.3. *hp* Refinement Procedure. The *hp*-Refinement procedure is outlined in Figure 2. In this procedure, the target number of degrees of freedom for the new mesh, denoted by $NDTRGT_0$, is given by

$$NDTRGT_0 = \min(NDTRGT, NDF \times 4^{1/p_{ave}}). \tag{2.2}$$

Here *NDTRGT* is the target number given by user and the average degree of elements in the current mesh is estimated by the following formula

$$p_{ave} = \sqrt{\frac{NDF}{NVF}} \tag{2.3}$$

where *NDF* and *NVF* are the number of degrees of freedom and the number of vertices of the current mesh respectively .

The use of (2.2) tries to force a geometric increase in the number of degrees of freedom in each refinement step. This is an empirical formula based on the observation that the dimensions of the subspaces (i.e. *NDF*) must grow more slowly for higher order elements or the higher rates of convergence promised by these elements might not be achieved.

R1 Create a heap with respect to η_τ with the largest error estimate $\eta_{\tau_{\max}}$ at the root;
R2 If $NDF \approx NDTRGT_0$ or $\eta^2_{\tau_{max}} \le \eta^2_{ave}/2$, then **exit**.
R3 Decide to refine $\tau_{\max}$ in h or p based on $\alpha_{\tau_{\max}}$
R4 Refine element $\tau_{\max}$, and possibly others as required.
R5 Update error indicators for affected elements.
Add new elements as needed.
Remake the heap. Go to **R2**.

FIGURE 2. *hp*-refinement procedure

While we normally expect the refinement loop to exit when the target number of degrees of freedom is approximately achieved, we can also exit if the largest error in the current mesh is sufficiently small. In particular,

$$\eta^2_{\tau_{\max}} \le \frac{\eta^2_{ave}}{2} = \frac{1}{2N} \sum_{t \in \Omega_I} \eta_t^2 \tag{2.4}$$

where Ω_I is the fine subregion associated with processor I in the case of parallel computation, and $\Omega_I \equiv \Omega$ otherwise; N is the number of triangles in Ω_I.

In *hp*-refinement procedure, the most interesting test is to decide between h-refinement and p-refinement for element $\tau_{\max}$. In our implementation, we use h-refinement if

$$\alpha_{\tau_{\max}} > 2\alpha_{ave} \tag{2.5}$$

and use p-refinement otherwise. Here α_{ave} is the average of all of α_τ in the mesh before the refinement. If the scaling factor $\alpha_{\tau_{\max}} \approx 1$, then the recovered derivatives and the error estimate are consistent, and we assume that the solution is locally smooth, which in turn justifies p-refinement. Large values of $\alpha_{\tau_{\max}}$ empirically correspond to locally non-smooth behavior of the solution, and this in turn suggests h-refinement.

While (2.5) is the main test for *hp*-refinement, we also check for round-off error problems, and for the maximum degree (limited by the available quadrature rules), and change the decision suggested by (2.5) if necessary. Finally, for very coarse meshes we always choose h-refinement. In particular, we choose h-refinement until the relative error

$$RELERP \le \frac{1}{5} \tag{2.6}$$

is satisfied. Condition (2.6) tries to insure that the mesh contains enough elements that hp-refinement is a viable option. We have also observed that our error estimates are sometimes unreliable on extremely coarse meshes, perhaps due to data oscillations, or more generally that h is not sufficiently small for the asymptotic error behavior underlying our derivative recovery procedures to hold.

3. Parallel hp-Adaptive Paradigm

3.1. Bank-Holst paradigm. In this work, we use the Bank-Holst paradigm [BH00, BH03, Ban06b, BO07, Ova04, Mit98a, Mit98b], that addresses the load balancing problem in a new way, requiring far less communication. Another important point is that our approach allows serial adaptive finite element codes to run in a parallel environment without a large investment in additional coding. This approach has three main components:

Step 1: A small (nonlinear) problem is solved on an initial coarse mesh, and *a posteriori* error estimates are computed for the coarse grid solution. The triangulation is partitioned such that each subdomain has approximately equal *error* (although they can significantly differ in size, numbers of elements and degrees of freedom).

Step 2: Each processor is provided the complete coarse mesh and solution, and instructed to solve the *entire* (nonlinear) problem, with the stipulation that its adaptive refinement should be limited largely to its own partition. Load balancing is achieved by instructing each processor to create a refined mesh with the same number of degrees of freedom.

Step 3: A final mesh is computed using the union of the refined partitions provided by each processor. This mesh is reconciled such that the (virtual) mesh made up of the union of the refined subregions would be conforming. A final solution is computed, using a domain decomposition method. An initial guess is provided by the local solutions.

The above approach has several interesting features. First, the load balancing problem (Step 1) is reduced to the numerical solution of a small problem on a single processor. It can be done using any serial adaptive finite element code without any modifications. Second, the adaptive mesh generation calculation (Step 2) takes place independently on each processor, and can also be performed with no communication.

The only parts of the calculation requiring communication are

(1) the initial fan-out of the mesh distribution to the processors, once the decomposition is determined by the error estimator.
(2) the mesh regularization, requiring communication to produce a global conforming mesh.
(3) the final solution phase. Note that a good initial guess for Step 3 is provided in Step 2 by taking the solution from each subregion restricted to its partition.

3.2. Bank-Holst paradigm and hp-refinement. Even though hp-refinement can be used directly with Bank-Holst paradigm without any restriction, a careful use of hp-refinement can make the combination smoother and more effective. The following are the practices that we employ

(1) In Step 1, only h-refinement is used. In this step, having a good approximate solution and accurate a posteriori error estimate is not the first priority. It is more critical to create a mesh with a sufficient number of elements so that a good partition of the domain can be obtained. In addition, avoiding p- and hp-refinement in this step is also appropriate as these strategies are not effective on extremely coarse meshes.
(2) At the beginning of Step 2 on each processor, if the number of elements in the local subdomain is small, adaptive h-refinement or local uniform h-refinement should be used first. After that, automatic hp-refinement can be used.

Since the adaptive enrichment on each processor (Step 2) is completely independent of what happens on other processors, the global refined mesh, constructed from the meshes associated with the refined regions on each of the processors, is initially non-conforming along the interface system. With the use of hp-refinement, the task to reconcile the local meshes can be challenging. The meshes are unstructured in geometry (in h) and have variable degree (variable p). In addition, there is no refinement tree, and nonconformity exists in both h and p. Thus, we need to efficiently identify and resolve these nonconformities, and ultimately to establish links between degrees of freedom on the fine mesh interface system on a given processor and the corresponding degrees of freedom on other processors which share its interface. For detail descriptions of this task, we refer to [BNar].

4. Domain Decomposition Solver

In this section, we describe the domain decomposition algorithm implemented in *PLTMG* for Step 3 of the Bank-Holst paradigm presented in Subsection 3.1. This algorithm is described in detail in [BN11, BL03, Ban06a, Lu04, BV08].

For simplicity in our discussion, we restrict attention to the case of just two subdomains. In our scheme, each subregion contributes equations corresponding all fine degrees of freedom, including its interface. Thus in general there will be multiple unknowns and equations in the global system corresponding to the interface degrees of freedom. This is handled by equality constraints that impose continuity at all degrees of freedom on the interface. The result is a mortar-element like formulation, using Dirac δ functions for the mortar element space. In any event, with a proper ordering of unknowns, the global system of equations has the block 5×5 form

$$\begin{pmatrix} A_{11} & A_{1\gamma} & & & \\ A_{\gamma 1} & A_{\gamma\gamma} & & & I \\ & & A_{\nu\nu} & A_{\nu 2} & -I \\ & & A_{2\nu} & A_{22} & \\ & I & -I & & \end{pmatrix} \begin{pmatrix} \delta U_1 \\ \delta U_\gamma \\ \delta U_\nu \\ \delta U_2 \\ \Lambda \end{pmatrix} = \begin{pmatrix} R_1 \\ R_\gamma \\ R_\nu \\ R_2 \\ U_\nu - U_\gamma \end{pmatrix}. \tag{4.1}$$

Here A_{11} and A_{22} correspond to the fine degrees of freedom on processors 1 and 2, respectively, that are not on the interface, while $A_{\gamma\gamma}$ and $A_{\nu\nu}$ correspond to interface points. The fifth block equation imposes continuity, and its corresponding Lagrange multiplier is Λ. The identity matrix appears because the global fine mesh is conforming. The introduction of the Lagrange multiplier and the saddle point formulation (4.1) are only for expository purposes; indeed, Λ is never computed or updated.

On processor 1, we develop a similar but "local" saddle point formulation. That is, the fine mesh subregion on processor 1 is "mortared" to the remaining coarse mesh on processor 1. This leads to a linear system of the form

$$\begin{pmatrix} A_{11} & A_{1\gamma} & & & \\ A_{\gamma 1} & A_{\gamma\gamma} & & & I \\ & & \bar{A}_{\nu\nu} & \bar{A}_{\nu 2} & -I \\ & & \bar{A}_{2\nu} & \bar{A}_{22} & \\ & I & -I & & \end{pmatrix} \begin{pmatrix} \delta U_1 \\ \delta U_\gamma \\ \delta \bar{U}_\nu \\ \delta \bar{U}_2 \\ \Lambda \end{pmatrix} = \begin{pmatrix} R_1 \\ R_\gamma \\ R_\nu \\ 0 \\ U_\nu - U_\gamma \end{pmatrix}, \tag{4.2}$$

where quantities with a bar (e.g., $\bar{A}_{22}$) refer to the coarse mesh. A system similar to (4.2) can be derived for processor 2. With respect to the right hand side of (4.2), the interior residual R_1 and the interface residual R_γ are locally computed on processor 1. We obtain the boundary residual R_ν, and boundary solution U_ν from processor 2; processor 2 in turn must be sent R_γ and U_γ. The residual for the coarse grid interior points is set to zero. This avoids the need to obtain R_2 via communication, and to implement a procedure to restrict R_2 to the coarse mesh on processor 1. Given our initial guess, we expect $R_1 \approx 0$ and $R_2 \approx 0$ at all iteration steps. R_γ and R_ν are not generally small, but $R_\gamma + R_\nu \to 0$ at convergence.

As with the global formulation (4.1), equation (4.2) is introduced mainly for exposition. The goal of the calculation on processor 1 is to compute the updates δU_1 and δU_γ, which contribute to the global conforming solution. To this end, we formally reorder (4.2) as

$$\begin{pmatrix} & -I & & I & \\ -I & \bar{A}_{\nu\nu} & & & \bar{A}_{\nu 2} \\ & & A_{11} & A_{1\gamma} & \\ I & & A_{\gamma 1} & A_{\gamma\gamma} & \\ & \bar{A}_{2\nu} & & & \bar{A}_{22} \end{pmatrix} \begin{pmatrix} \Lambda \\ \delta \bar{U}_\nu \\ \delta U_1 \\ \delta U_\gamma \\ \delta \bar{U}_2 \end{pmatrix} = \begin{pmatrix} U_\nu - U_\gamma \\ R_\nu \\ R_1 \\ R_\gamma \\ 0 \end{pmatrix}. \tag{4.3}$$

Block elimination of the Lagrange multiplier Λ and $\delta \bar{U}_\nu$ in (4.3) leads to the block 3×3 Schur complement system

$$\begin{pmatrix} A_{11} & A_{1\gamma} & \\ A_{\gamma 1} & A_{\gamma\gamma} + \bar{A}_{\nu\nu} & \bar{A}_{\nu 2} \\ & \bar{A}_{2\nu} & \bar{A}_{22} \end{pmatrix} \begin{pmatrix} \delta U_1 \\ \delta U_\gamma \\ \delta \bar{U}_2 \end{pmatrix} = \begin{pmatrix} R_1 \\ R_\gamma + R_\nu + \bar{A}_{\nu\nu}(U_\nu - U_\gamma) \\ \bar{A}_{2\nu}(U_\nu - U_\gamma) \end{pmatrix}. \tag{4.4}$$

The system matrix in (4.4) corresponds to the final adaptive refinement step on processor 1, with possible modifications due to global fine mesh regularization. It is exactly the matrix used in the preliminary local solve to generate the initial guess for the global domain decomposition iteration. In the solution of (4.4), the components δU_1 and δU_γ contribute to the global solution update, while $\delta \bar{U}_2$ is discarded. We remark that the global iteration matrix corresponding to this formulation is not symmetric, even if all local system matrices are symmetric.

The domain decomposition algorithm is incorporated as the solver for the approximate Newton method which is used to solve the discretized equations [1]. In particular, only one domain decomposition iteration (a so-called *inner iteration*) is used in each approximate Newton step. Thus, loosely speaking, each solve of (4.4) alternates with a line search step in which the global solution is updated. The

[1] Even when the original problem is linear, we formally apply the approximate Newton method and the linearity is realized after one Newton step.

Newton line search procedure requires global communication to form some norms and inner products, as well as the boundary exchange described above.

In this paper, the convergence of the domain decomposition algorithm is determined using either one of the the following criteria:

$$\frac{\|\delta\mathcal{U}^k\|_G}{\|\mathcal{U}^k\|_G} \le \max\left(\frac{\|\delta\mathcal{U}^0\|_G}{\|\mathcal{U}^0\|_G}, \frac{\|\nabla e_h\|_{0,\Omega}}{\|\nabla u_h\|_{0,\Omega}}\right) \times 10^{-1}. \tag{4.5}$$

or

$$\frac{\|\delta\mathcal{U}^k\|_G}{\|\mathcal{U}^k\|_G} \le \frac{\|\delta\mathcal{U}^0\|_G}{\|\mathcal{U}^0\|_G} \times 10^{-4}. \tag{4.6}$$

Here $\|\delta\mathcal{U}^k\|_G$ and $\|\mathcal{U}^k\|_G$ are the discrete global norm of the approximate solution and the update respectively, at iteration k, while $\|\nabla e_h\|_{0,\Omega}$ and $\|\nabla u_h\|_{0,\Omega}$ are the a posteriori error estimate and the norm of the initial solution. Normally, (4.5) is sufficient for the purposes of computing an approximation to the solution of the partial differential equation. However, we also use the more stringent criterion (4.6) to illustrate the behavior of the domain decomposition solver as an iterative method for solving linear systems of equations.

5. 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 (2.33GHz) and 16GB 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* [Ban11] are written in FORTRAN; the *gfortran* compiler was used in these experiments, invoked using the script *mpif90* and optimization flag *-O*.

In our 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 surrounding an airfoil-shaped object (see Figure 3, on the left).

At the beginning, an adaptive mesh of size $N_c = 50K$ was created on one processor. 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, first in h and then automatic hp, creating a mesh of size N_P. In this experiment, N_P was chosen to be $400K, 600K$, and $800K$. This resulted in global meshes varying in size from approximately $750K$ to $161M$. These global meshes were regularized to be h-conforming and p-conforming by applying appropriate refinement and unrefinement in both h and p to the local meshes. For the case $N_P = 800K$, $P = 32$, the solution and the load balance is shown in Figure 3. The mesh density and degree density of the global mesh and one local mesh are shown in Figure 4 and Figure 5. As expected, both the mesh density and the degree density are high in the local subdomain and much lower elsewhere in the local mesh.

After the global meshes are regularized, a global DD solve was made to obtain the global solution. The results are summarized in Table 1. For the case $N_P =$

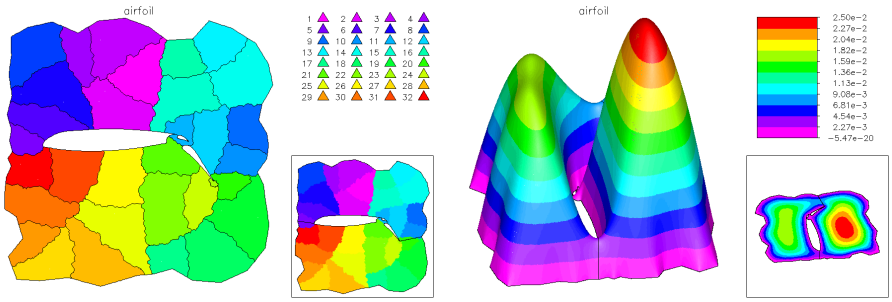

FIGURE 3. The load balance (left) and solution (right) in the case $N_P = 800K$, $P = 32$.

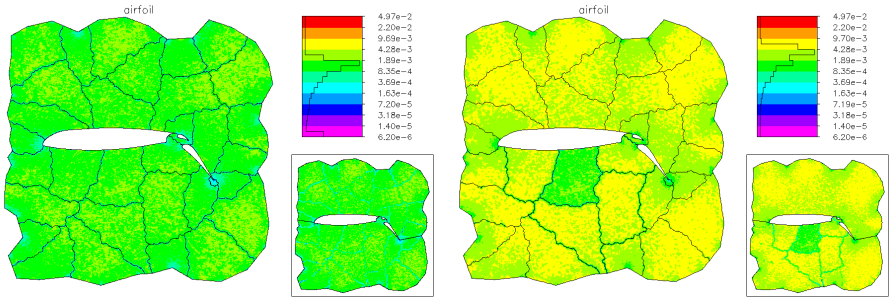

FIGURE 4. The mesh density for the global mesh (left) and for one of the local meshes (right) in the case $N_P = 800K$, $P = 32$.

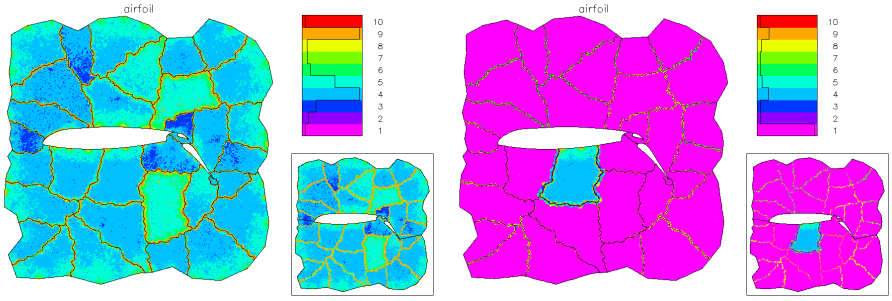

FIGURE 5. The degree density for the global mesh (left) and for one of the local meshes (right) in the case $N_P = 800K$, $P = 32$.

$800K, P = 256$, the convergence history when using the strict criterion (4.5) is shown in Figure 6.

For this approach of domain decomposition, the number of degrees of freedom of the global mesh is predicted by

$$N \approx PN_P - (P-1)N_c. \tag{5.1}$$

Equation (5.1) 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=50K, N_P=800K, $P = 256$, (5.1) predicts $N \approx 192050000$ when actually $N = 161009439$.

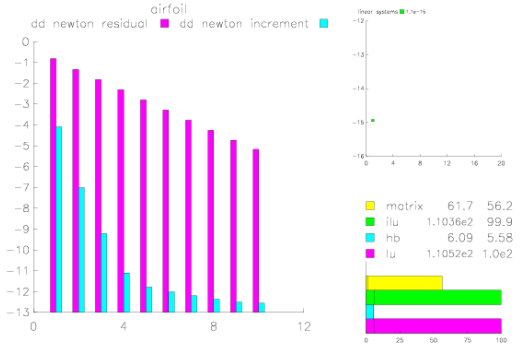

FIGURE 6. Convergence history of the domain decomposition solver for the case $N_P = 800K, P = 256$ when the criterion (4.6) is used.

TABLE 1. Convergence Results. Numbers of iterations needed to satisfy convergence criteria are given in the column labeled DD. The numbers in parentheses are the number of iterations required to satisfy (4.5) which is the default convergence criterion used by *PLTMG*.

	$N_P = 400K$		$N_P = 600K$		$N_P = 800K$	
P	N	DD	N	DD	N	DD
2	750247	14 (4)	1150487	14 (4)	1550418	14 (4)
4	1450884	13 (4)	2252526	19 (5)	3050963	20 (5)
8	2851662	19 (6)	4442935	19 (5)	6101079	20 (5)
16	5670458	20 (4)	8975651	20 (3)	12269476	20 (3)
32	11315140	20 (4)	17721124	20 (4)	24019421	20 (4)
64	20260417	20 (4)	31612049	19 (4)	44327632	18 (4)
128	34750517	10 (3)	56984391	8 (3)	83764874	8 (3)
256	61949578	13 (3)	108554486	10 (3)	161009439	10 (4)

References

[Ban06a] Randolph E. Bank, *A domain decomposition solver for a parallel adaptive meshing paradigm*, Domain Decomposition Methods in Science and Engineering XVI (Olof B. Widlund and David E. Keyes, eds.), Lecture Notes in Computational Science and Engineering, vol. 55, Springer-Verlag, 2006, pp. 3–14.

[Ban06b] Randolph E. Bank, *Some variants of the Bank-Holst parallel adaptive meshing paradigm*, Comput. Vis. Sci. **9** (2006), no. 3, 133–144. MR MR2271791

[Ban11] ———, *PLTMG: A software package for solving elliptic partial differential equations, users' guide 11.0*, Tech. report, Department of Mathematics, University of California at San Diego, 2011.

[BH00] Randolph E. Bank and Michael Holst, *A new paradigm for parallel adaptive meshing algorithms*, SIAM J. Sci. Comput. **22** (2000), no. 4, 1411–1443 (electronic). MR MR1797889 (2002g:65117)

[BH03] ———, *A new paradigm for parallel adaptive meshing algorithms*, SIAM Rev. **45** (2003), no. 2, 291–323 (electronic), Reprinted from SIAM J. Sci. Comput. **2**2 (2000), no. 4, 1411–1443 [MR1797889]. MR MR2010380

[BL03] Randolph E. Bank and Shaoying Lu, *A domain decomposition solver for a parallel adaptive meshing paradigm*, SIAM J. on Scientific Computing **45** (2003), 292–323.

[BN11] Randolph E. Bank and Hieu Nguyen, *Domain decomposition and hp-adaptive finite elements*, Domain Decomposition Methods in Science and Engineering XIX (Berlin) (Yunqing Huang, Ralf Kornhuber, Olof Widlund, and Jinchao Xu, eds.), Lecture Notes in Computational Science and Engineering, Springer-Verlag, 2011, pp. 3–13.

[BN12] ———, *hp adaptive finite elements based on derivative recovery and superconvergence*, Comput. Vis. Sci. (2012), 1–10.

[BNar] ———, *Mesh regularization in Bank-Holst parallel hp-adaptive meshing*, Domain Decomposition Methods in Science and Engineering XX, Lecture Notes in Computational Science and Engineering, Springer-Verlag, to appear.

[BO07] Randolph E. Bank and Jeffery S. Ovall, *Dual functions for a parallel adaptive method*, SIAM J. on Scientific Computing **29** (2007), 1511–1524.

[BV08] Randolph E. Bank and Panayot S. Vassilevski, *Convergence analysis of a domain decomposition paradigm*, Computing and Visualization in Science **11** (2008), 333–350.

[BX03a] Randolph E. Bank and Jinchao Xu, *Asymptotically exact a posteriori error estimators. I. Grids with superconvergence*, SIAM J. Numer. Anal. **41** (2003), no. 6, 2294–2312 (electronic). MR MR2034616 (2004k:65194)

[BX03b] ———, *Asymptotically exact a posteriori error estimators. II. General unstructured grids*, SIAM J. Numer. Anal. **41** (2003), no. 6, 2313–2332 (electronic). MR MR2034617 (2004m:65212)

[BXZ07] Randolph E. Bank, Jinchao Xu, and Bin Zheng, *Superconvergent derivative recovery for Lagrange triangular elements of degree p on unstructured grids*, SIAM J. Numer. Anal. **45** (2007), no. 5, 2032–2046 (electronic). MR MR2346369 (2009b:65293)

[Lu04] Shaoying Lu, *Parallel adaptive multigrid algorithms*, Ph.D. thesis, Department of Mathematics, University of California at San Diego, 2004.

[Mit98a] William F. Mitchell, *The full domain partition approach to distributing adaptive grids*, Applied Numerical Mathematics **26** (1998), 265–275.

[Mit98b] ———, *A parallel multigrid method using the full domain partition*, Electronic Transactions on Numerical Analysis **6** (1998), 224–233.

[MM11] W.F. Mitchell and M.A. McClain, *A survey of hp-adaptive strategies for elliptic partial differential equations*, Recent Advances in Computational and Applied Mathematics (2011), 227–258.

[Ngu10] Hieu Nguyen, *p- and fully automatic hp- adaptive finite element methods for elliptic partial differential equations methods*, Ph.D. thesis, University of California, San Diego, 2010.

[Ova04] Jeffrey S. Ovall, *Duality-based adaptive refinement for elliptic pdes*, Ph.D. thesis, Department of Mathematics, University of California at San Diego, 2004.

Department of Mathematics, University of California, San Diego, La Jolla, California 92093-0112.
E-mail address: rbank@ucsd.edu

Department of Computer Science, University of California, Davis, Davis, California 95616.
E-mail address: htrnguyen@ucdavis.edu