CFD PPLTMG Using A Posteriori Error Estimates and Domain Decomposition

Randolph E. Bank, [1] Michael Holst, [2] Bertrand Mantel, [3]
Jacques Périaux, [4] and Chun Hua Zhou [5]

Abstract. This two-part paper examines two approaches for mesh adaptation, using combinations of *a posteriori* error estimates and domain decomposition.

In the first part, we consider a domain decomposition method applied to the generalized Stokes problem, with mesh adaptation in each subdomain using the *a posteriori* local error estimator as adaptation indicator. We apply domain decomposition without overlapping, and the condition of compatibility on the interface is enforced weakly via a Lagrange multiplier. The *a posteriori* error estimates for this problem are constructed corresponding to linear approximation for the velocity, density or pressure and the Lagrange multiplier associated with the constraint on the interface. All the errors are approximated in the space of bump quadratic functions, a hierarchical basis of the space of quadratic functions. The localization of the error estimation is based on solving the local problems, and an equivalence between the error estimators and the exact errors has been demonstrated.

In the second part, we outline a new approach for adaptive mesh generation which allows for the use of existing (sequential) adaptive mesh refinement algorithms and software in a distributed computing environment. Moreover, the primary components of this new algorithm are handled in parallel, with no communication requirements. We first outline the algorithm, and then give a numerical example using PLTMG as the sequential adaptive code used in the new algorithm.

[1] Department of Mathematics, University of California at San Diego, La Jolla, CA 92093, USA.
[2] Department of Mathematics, University of California at San Diego, La Jolla, CA 92093, USA.
[3] Dassault Aviation, DGT/DEA B.P. 300, 92214 St Cloud Cedex, France.
[4] Dassault Aviation, DGT/DEA B.P. 300, 92214 St Cloud Cedex, France.
[5] Department of Aerodynamics, NUAA, 92552 Nanjing, PRC.

1 Error Estimator for the Stokes Problem with Domain Decomposition

1.1 Domain decomposition for the generalized Stokes problem

1.1.1 *Problem definition*

Let Ω be a bounded open subset of R^2 with boundary Γ. We propose to solve in Ω the following generalized compressible Stokes problem with Dirichlet boundary conditions:

$$\left\{\begin{array}{ll} \alpha U - \mu\Delta U + \beta \bigtriangledown \rho = \vec{f} & \text{in } \Omega \\ \beta \bigtriangledown .U + \gamma\rho = h & \text{in } \Omega \\ U = \vec{g} & \text{on } \Gamma \end{array}\right. \tag{1}$$

where α, β and γ are positive constants.

Here we consider the decomposition of Ω into two non-overlapping subdomains, Ω_1 and Ω_2:

$$\Omega = \Omega_1 \cup \Omega_2$$
$$\Omega_1 \cap \Omega_2 = \emptyset$$

We denote the interface between Ω_1 and Ω_2 by S.

Now the global problem in Ω is replaced by the problems posed in each subdomain:

$$\left\{\begin{array}{ll} \multicolumn{2}{c}{\text{for } i = 1,2 \ (U_i, \rho_i) \text{ solves}} \\ \alpha U_i - \mu \bigtriangleup U_i + \beta \bigtriangledown \rho_i = \vec{f}_i & \text{in } \Omega_i \\ \beta \bigtriangledown .U_i + \gamma\rho_i = h & \text{in } \Omega_i \\ U_i = \vec{g} & \text{on } \Gamma_i \\ U_2 = U_1 & \text{on } S \end{array}\right. \tag{2}$$

with $\Gamma_i = \partial\Omega_i \cap \Gamma$, $\forall i = 1, 2$

1.1.2 *Lagrangian Formulation*

In order to solve problem (2), a Lagrange multiplier $\vec{\lambda}$ is introduced, associated with the interface condition $U_1 = U_2$, that is to say, we weakly impose continuity of the solution U across S.

We define the following spaces:

$X_g = X_{g1} \times X_{g2}$
and
$X_{gi} = \left\{ V_i \in (H^1(\Omega_i))^2; V_i \mid_{\Gamma_i} = \vec{g} \right\}$
$Y = L^2(\Omega_1) \times L^2(\Omega_2)$
$Z = \left(H^{-\frac{1}{2}}(S) \right)^2$

Therefore we search the solution of following Lagrangian formulation:

$$\left\{ \begin{array}{l} \text{Find } (U, \rho, \vec{\lambda}) \in X_g \times Y \times Z, \text{ such that:} \\ \sum_{i=1}^{2} \left[\alpha \int_{\Omega_i} U_i V_i dx + \mu \int_{\Omega_i} \nabla U_i \nabla V_i dx \right. \\ \left. -\beta \int_{\Omega_i} \rho_i \nabla . V_i dx \right] - \int_S \vec{\lambda}(v_2 - v_1) d\sigma \\ \qquad = \sum_{i=1}^{2} (\vec{f}_i, V_i) \qquad \forall V \in X_0 \\ -\sum_{i=1}^{2} \left[\beta \int_{\Omega_i} \nabla . U_i q_i dx + \gamma \int_{\Omega_i} \rho_i q_i dx \right] \\ \qquad = -\sum_{i=1}^{2} (h_i, q_i) \qquad \forall q \in Y \\ -\int_S \vec{\eta}\,(U_2 - U_1)\, d\sigma = \vec{0} \qquad \forall \vec{\eta} \in Z \end{array} \right. \tag{3}$$

Here for simplicity we denote by the integral on S, the product of duality. The left hand side of (3) defines a bilinear form subsequently denoted by $T_S((U, \rho, \vec{\lambda}), (V, q, \vec{\eta}))$.

Remark 1 When the interfaces S cross with Dirichlet boundaries, the variational formulation (3) is equivalent to original problem (2) only under the hypothesis $\vec{g} \in \left(H^{\frac{1}{2}+\epsilon}(\Gamma) \right)^2$ for $\epsilon \in \left[0, \frac{1}{2}\right]$.

Remark 2 The Lagrange multiplier $\vec{\lambda}$ can be interpreted as:

$$\vec{\lambda} = \left[\mu \frac{\partial U_i}{\partial \vec{n}_i} + \beta \rho_i \vec{n}_i \right]_S (-1)^i \qquad \text{in } \left(H^{-\frac{1}{2}}(S) \right)^2$$

The linear system of problem (3) can be solved by a preconditioned conjugate gradient algorithm. In each iteration of the algorithm, a generalized Stokes problem is solved in each subdomain. In this work, we solve it with a standard Petrov-Galerkin formulation.

1.2 Discretization

Now we consider the discretization of problem (3) using domain decomposition.

The subdomains Ω_1 and Ω_2 are approximated Ω_{h1} and Ω_{h2} which are polygonal domains de R^2. Let $\mathcal{T}_{h1}$ and $\mathcal{T}_{h2}$ be the domain discretization of Ω_{h1} and Ω_{h2}:

$$\overline{\Omega}_i = \bigcup_{\tau \in \mathcal{T}_{hi}} \tau \qquad \forall i = 1, 2$$

The interface S is approximated by a polygonal line S_h. The triangulations of of Ω_{h1} and Ω_{h2} are not required to match along the interface S_h.

Since the spaces X_g, Y and Z are not independent, the choice of the discrete Lagrange multiplier space Z_h for Z is influenced by the choice of the discrete spaces X_{gh} and Y_h for X_g and Y, respectively.

Since the Stokes equations are stable in the Petrov-Galerkin formulation, we can discretize velocity and density with piecewise polynomials of the same degree. The spaces X_g and Y are approximated spaces of Lagrange finite elements with degree $k = 1$ or 2:

$$\begin{array}{rcl} X_{gh} & = & X_{gh1} \times X_{gh2} \text{ and} \\ X_{ghi} & = & \{V_{hi} \in (C^0(\overline{\Omega}_i))^2; V_{hi} \mid_{\Gamma_i} = \vec{g}; \\ & & V_{hi} \in (P^k(\tau))^2, \forall \tau \in \mathcal{T}_{hi}\} \\ Y_h & = & Y_{h1} \times Y_{h2} \text{ and} \\ Y_{hi} & = & \{v_{hi} \in C^0(\overline{\Omega}_i); v_{hi} \in P^k(\tau), \forall \tau \in \mathcal{T}_{hi}\} \end{array}$$

Then we take a discretization for space Z with help of finite elements P^k on S_h, Z_h is defined by:

$$Z_h = \{V_h \in (C^0(\overline{S}))^2; V_h \mid_{B_j} \in (P^k(B_j))^2, \forall B_j \in S_h\}$$

$B_j = [p_j, p_{j+1}]$ is defined as follow:

The partition $p_1 < p_2 ... < p_m$ is the largest trace coming from the triangulations $\mathcal{T}_{h1}$ and $\mathcal{T}_{h2}$ on the interface (see figure 1).

1.3 A posteriori error estimates

1.3.1 *Formulation of a posteriori error*

Let $(U_L, \rho_L, \vec{\lambda}_L)$ denote the solution obtained using piecewise linear functions (k=1) and $(U, \rho, \vec{\lambda})$ is the solution of system (3). Let

$$(\vec{e}, \varepsilon, \vec{\theta}) = (U, \rho, \vec{\lambda}) - (U_L, \rho_L, \vec{\lambda}_L) \tag{4}$$

be the error of linear discretization of (3) and then define the spaces of piecewise H^1 functions:

$$\begin{array}{l} H^1_T = H^1_{T_1} \times H^1_{T_2} \text{ and} \\ H^1_{T_i} = \prod_{\tau \in \mathcal{T}_{hi}} H^1(\tau) = \{v, v \mid_\tau \in H^1(\tau), \tau \in \mathcal{T}_{hi}\} \end{array} \tag{5}$$

For $(V,q,\vec{\eta}) \in (H_T^1)^2 \times H_T^1 \times Z$, after integrating by parts over each triangle, we obtain the system for $(\vec{e},\varepsilon,\vec{\theta})$:

$$T_S((\vec{e},\varepsilon,\vec{\theta}),(V,q,\vec{\eta})) = L(V,q,\vec{\eta}) + J((\vec{e},\varepsilon),(V,q)) \tag{6}$$

with the linear forms $L(.)$ and bilinear forms $J(.,.)$:

$$\begin{array}{l} L(V,q,\vec{\eta}) \\ \sum_{i=1}^{2}[(\vec{f}_i,V_i)+(h_i,V_i)-\mu(\left[\frac{\partial U_{Li}}{\partial n}\right]_A,[V_i]_J)_{E_i} \\ +\beta(\rho_{Li}\vec{n},[V_i]_J)_{E_i}+\sum_{\tau\in\mathcal{T}_{hi}}\lambda_\tau(\vec{f}_i,\beta\bigtriangledown q_i-\alpha V_i)_\tau] \\ -T_S((U_L,\rho_L,\vec{\lambda}_L),(V,q,\vec{\eta})) \end{array} \tag{7}$$

and

$$J((\vec{e},\varepsilon),(V,q)) = \sum_{i=1}^{2}(-\mu\left[\frac{\partial \vec{e}_i}{\partial n}\right]_A + \beta\varepsilon_i\vec{n},[V_i]_J)_{E_i} \tag{8}$$

where the set E_i contains all interior edges of the triangles in $\mathcal{T}_{hi}$ and $[.]_A$, $[.]_J$ mean the average and jump values of a function across $e \in E_1 \cup E_2$.

1.3.2 Definition of local estimator of a posteriori error

We denote $X_{gh} \times Y_h \times Z_h$ by M_h^Q when $k=2$ and M_h^L when $k=1$. We have the hierarchical decomposition

$$M_h^Q = M_h^L \oplus M_h^K$$

where M_h^K is a space of the continuous quadratic bump functions.

If we resolve the problem (3) with M_h^Q using the hierarchical basis, one expects intuitively that the component of the new solution lying in M_h^L will change very little from the calculation with M_h^L. So, the component lying in M_h^K should be a good approximation to the error of the solution on the original space M_h^L.

Corresponding to system (6) and in order to improve the efficiency of the computation of a posteriori error estimator we use a space of discontinuous quadratic bump functions, $(K)^2 \times K \times (K_S)^2$, to approximate the error $(\vec{e},\varepsilon,\vec{\theta})$.

Note that none of the terms in the bilinear form $J(.,.)$ can be calculated from available data since the error terms $\vec{e}$ and ε are not known. On the contrary, all the terms in the linear form $L(.)$ can be calculated from the current solution $(U_L,\rho_L,\vec{\lambda}_L)$. Therefore, abandoning the term $J(.,.)$, we define the local a posteriori error estimator $(\vec{e}_h,\varepsilon_h,\vec{\theta}_h)$ by the following system:

$$\left\{ \begin{array}{l} \text{Find } (\vec{e}_h,\varepsilon_h,\vec{\theta}_h) \in (K)^2 \times K \times (K_S)^2 \text{ so that:} \\ \quad T_S((\vec{e}_h,\varepsilon_h,\vec{\theta}_h),(V,q,\vec{\eta}))_P = L(V,q,\vec{\eta})_P \\ \qquad \forall (V,q,\vec{\eta}) \in (K)^2 \times K \times (K_S)^2 \end{array} \right. \tag{9}$$

More explanation and details are given in the Zhou's thesis [3].

There are two cases for the definition of P:

Case 1

$$P = \bigcup_m \tau_m,\ \partial\tau_m \bigcap S_h \in B_j \text{ and } \tau_m \in \mathcal{T}_{hi}, \forall i = 1,2$$

where B_j is the support of $\vec{\theta}_h^j$, the error estimator of Lagrange multiplier. As an example in figure 1 P is an union of triangles with $*$. In this case the local Stokes problem (9) is solved by a conjugate gradient algorithm.

Case 2

$$P = \tau \text{ when } \partial\tau \bigcap S_h = \emptyset$$

In this case, no triangle has an interace edge, and the system (9) is the same as without domain decomposition, and reduces to the solution of a local Stokes problem on each triangle.

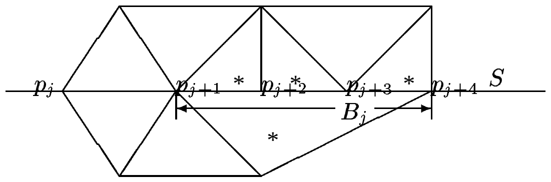

Figure 1. Interface partitioning and P definition in case 1

2 A Decoupled Strategy

The second strategy, still under development, is a new procedure for adaptive mesh generation which allows for the use of existing (sequential) adaptive mesh refinement algorithms and software in a distributed computing environment.

One of the most difficult obstacles to overcome in making effective use of parallel computers for adaptive finite element codes such as PLTMG [1] is the load balancing problem. As an adaptive method adjusts the mesh according to the features of the solution, elements

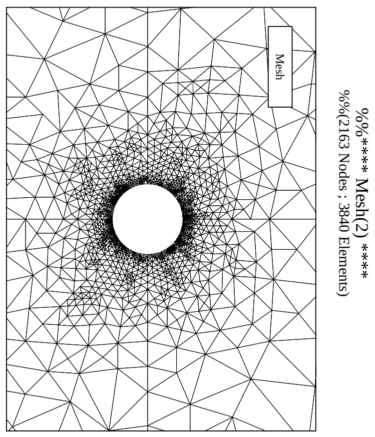

Figure 2. Density plot for the fourth mesh adaptation level

in some areas are refined, whereas others are not. If an initial mesh is distributed quite fairly among a number of processors, a very good error estimator (coupled with adaptive refinement) quickly produces a very bad work load imbalance among the processors. We propose an alternative approach that addresses the load balancing problem in a new way, requiring far less communication than current approaches. This approach also allows existing sequential adaptive PDE codes such as PLTMG to run in a parallel environment without a large investment in recoding.

Our approach has three main components:

1. We begin by solving a small (nonlinear) problem on an initial coarse mesh, and compute an *a posteriori* error estimate for the coarse grid solution. We partition the domain such that each subdomain has approximately equal *error* (although they can significantly differ in size and numbers of elements).
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 add the same number of nodes to its assigned subregion.
3. A final mesh is computed using the union of the refined partitions provided by each processor. This mesh could be regularized or left as a non-matching partitioned grid. In either case, a final solution computed, using a standard domain decomposition, mortar elements, or parallel multigrid technique, starting from the (very good) initial guess provided by the local solutions.

Our 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, using a sequential adaptive solver such as PLTMG, without requiring any modifications to the sequential solver. Second, the bulk of the calculation (step 2) takes place independently on each processor, and can also be performed with a sequential solver such as PLTMG with no modifications necessary for communication. (In PLTMG one line of code was added, which artificially multiplied *a posteriori* error estimates for elements outside a processor's partition by 10^{-6}.) Step 2 was motivated by recent work of Mitchell [4] on parallel multigrid methods.

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 local communication to produce a global conforming mesh, and (3) the final solution phase, which might require local communication (boundary exchanges). 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.

A more complete discussion of this method can be found in [2]. In the limited space available here, we are content to present an example. In this example we solve the full (nonlinear) potential flow problem about a NACA0012 airfoil.

The initial mesh, generated from the geometry description, had 384 triangles and 221 vertices (see Figure 3). The domain is partitioned into four subregions with approximately equal error using a recursive spectral bisection algorithm. The error is approximated using standard a posteriori error estimates [1].

Then four independent problems were solved with the mesh adaptively refined to one with approximately 5000 unknowns. Refined meshes for two of the four problems are shown in Figures 4-5.

The meshes from the four subproblems are combined to form a globally refined mesh with 37913 triangles and 18723 vertices. This mesh is conforming and is shown in Figure 6.

The initial guess for the global solution is computed by combining the solutions from the four subproblems. The local Mach number $M(\nabla u)$ computed from the initial guess for the the global solution is shown in Figure 7. (In our example, $M_\infty = .4$, with zero angle of attack.) Starting from this initial guess, the final global solution is computed; since the initial guess is very good, only a

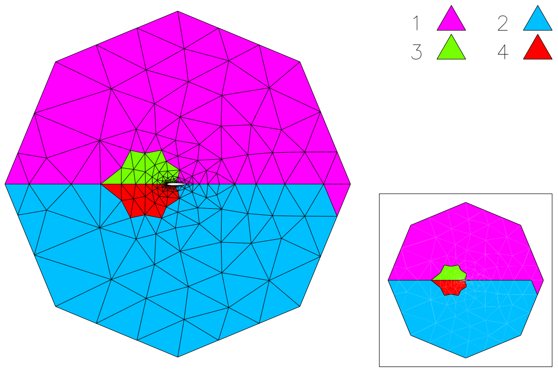

Figure 3. *The initial triangulation, partitioned into four subregions with approximately equal error.*

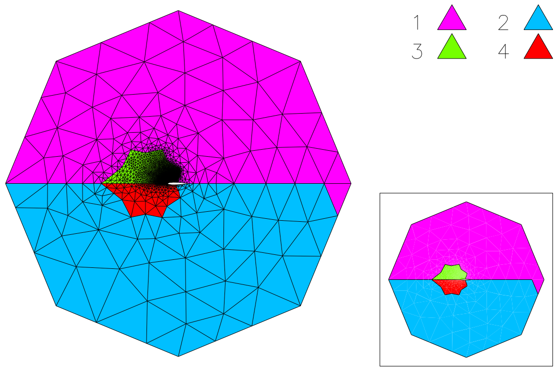

Figure 5. *The refined mesh for problem 3.*

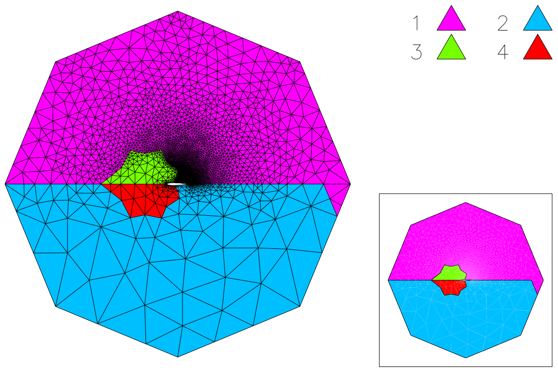

Figure 4. *The refined mesh for problem 1.*

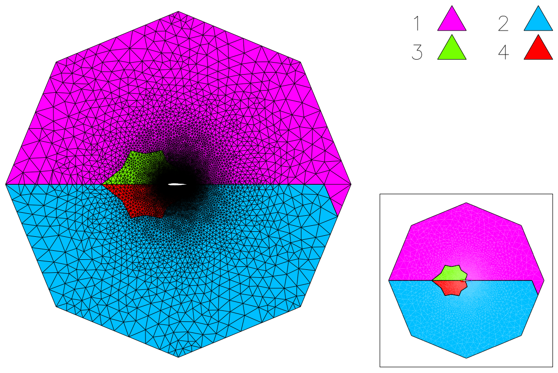

Figure 6. *The global refined mesh.*

small amount of smoothing is required. $M(\nabla u)$ for the final global solution is shown in Figure 7.

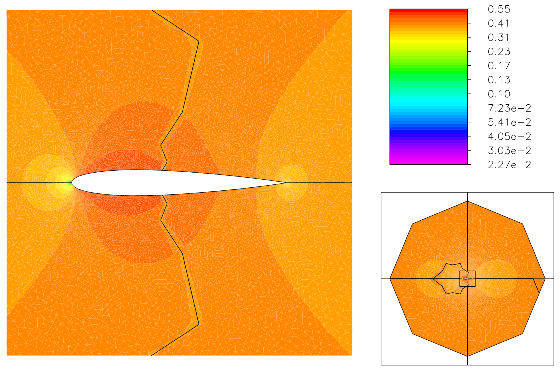

Figure 7. $M(\nabla u)$ computed from the initial guess for the global solution.

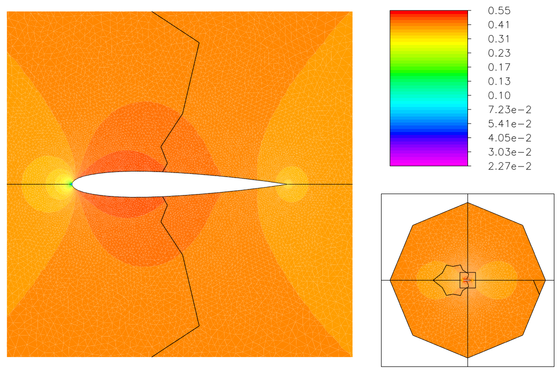

Figure 8. $M(\nabla u)$ computed from the final global solution.

In Table 1, we give the overall execution times (in seconds) for this calculation. The times are for an SGI Octane 195mhz R10000, using the f77 compiler options -O -32.

The initialization time includes generating a coarse mesh from the geometry description, solving the coarse mesh problem, computing a posteriori error estimates, and partitioning the domain. The initialization was done on one processor. The times for each of the subproblems include all aspects of the adaptive feedback loop, including matrix and right hand side assembly, solution of linear and nonlinear systems, a posteriori error estimation, and adaptive refinement and mesh smoothing. It also includes some relatively inexpensive clean-up, mainly removing unwanted parts of each mesh and solution in preparation for creating the global composite mesh. Postprocessing includes the creation of a global conforming mesh and forming an initial guess for the the solution on that mesh. In our present code, this is also done on one processor.

Initialization	0.82
Solve Subproblem 1	12.9
Solve Subproblem 2	13.0
Solve Subproblem 3	13.0
Solve Subproblem 4	12.9
Postprocessing	0.81

Table 1. Execution times (seconds) for the potential flow example.

From Table 1 we see that although the mix of calculations was different for each subproblem, the overall times do not vary too much. And since these problems were solved completely independently, there was no time spent in communication between processors, synchronization, etc.

REFERENCES

[1] Randolph E. Bank, *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations, Users' Guide 8.0*, Software, Environments and Tools, Vol. 5, SIAM, Philadelphia, 1998.

[2] Randolph E. Bank and Michael Holst, 'A new paradigm for parallel adaptive meshing algorithms', *in preparation.*

[3] Zhou Chun Hua, *Techniques de maillages éléments finis adaptés en parallèle par sous domaines non coincidents et estimation d'erreur a posteriori pour la résolution d'EDP en Mécanique des Fluides*, Université Pierre et Marie Curie, Jussieu, thèse doctorat de paris vi edn., 1998.

[4] William F. Mitchell, 'A parallel multigrid method using the full domain partition', *Electronic Transactions on Numerical Analysis*, **special issue for proceedings of the 8th Copper Mountain Conference on Multigrid Methods**, (submitted).