PARALLEL ALGEBRAIC MULTIGRIDS FOR STRUCTURAL MECHANICS *

MARIAN BREZINA †, CHARLES TONG ‡, AND RICHARD BECKER §

Abstract.
This paper presents the results of a comparison of three parallel algebraic multigrid (AMG) preconditioners for structural mechanics applications. In particular, we are interested in investigating both the scalability and robustness of the preconditioners. Numerical results are given for a range of structural mechanics problems with various degrees of difficulty, and show that algebraic multigrid methods can be applied successfully even to difficult problems in structural mechanics.

Key words. algebraic multigrid, preconditioning, parallel computing

AMS subject classifications. 65F10, 65N20

1. Introduction. Krylov methods with algebraic multigrid preconditioners have been demonstrated to offer fast convergence rates as well as good parallel efficiency for the iterative solution of linear systems arising from discretizations of many elliptic partial differential equations. Two classes of algebraic multigrid methods (AMG) have gained popularity - the classical AMG method originally conceived in [5] and practically developed by Ruge and Stüben [24] (RS), and smoothed aggregation multigrid (SA) proposed by Vaněk, Mandel and Brezina [31, 29]. In addition, improvements to SA have been made to handle more difficult problems such as Maxwell's equations [4]. In this paper we investigate another improvement to SA to handle difficult problems in structural mechanics. This improvement involves locally enriching the prolongation operator using low energy eigenvectors of subdomains in a finite element framework. The idea of enriching the coarse space was already recognized in [31] in conjunction with treatment of high-order equations, and was later expanded into an adaptive procedure similar to the one presented here by Brezina, Heberton, Mandel and Vaněk in [7] as a general principle to eliminate convergence problems due to local irregularities of the finite element discretization. The latter method was, however, limited to two-levels. Our implementation generalizes this coarse-space enriching approach to multi-level, replacing the subdomain smoothers with more efficient multilevel procedure that allows for further coarsening. We call this method GSA, an acronym for generalized smoothed aggregation.

Parallelization issues for AMG preconditioners are also considered in this paper. The application of algebraic multigrids involves two phases: the setup phase and the solution phase. A typical setup phase consists of coarse grid selection, construction of

*This revision is dated May 15, 2004. This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

†Department of Applied Mathematics, Campus Box 526, University of Colorado at Boulder, Boulder, CO 80309–0526, and Front Range Scientific Computations, Inc. E-mail : mbrezina@math.cudenver.edu. This work is sponsored by the Department of Energy under grant numbers DE-FC02-01ER25479 and DE-FG02-03ER25574; Lawrence Livermore National Laboratory under contract number B533502; Sandia National Laboratories under contract number 15268; and the National Science Foundation under VIGRE grant number DMS-9810751.

‡Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, L-560, Box 808, Livermore, CA 94551. E-mail : chtong@llnl.gov.

§New Technology and Engineering Division, Lawrence Livermore National Laboratory, L-129, Box 808, Livermore, CA 94551. E-mail : becker13@llnl.gov.

prolongation operators, and creation of coarse grid matrices. The original coarse grid selection procedures in both RS and SA are sequential in nature. Efforts to parallelize the coarsening step have been reported in [26, 16, 19, 15]. The construction of coarse grid matrices using triple matrix products can be parallelized in a straightforward manner, but perfect scalability is difficult to achieve. Scalability issues in the solution phase are associated mainly with the parallel smoothers, and with solution of the coarse grid problem. In this paper we discuss some of these parallelization issues in detail and make recommendations given different options.

This paper is organized as follows: in Section 2, we give an introduction of the three AMG methods in question. Section 3 is dedicated to expounding the GSA methods and providing theoretical motivation for the method. Section 4 describes in more detail the parallel aspects of the setup and solution phases. Numerical experiments will be presented in Section 5, followed by a few afterthoughts in Section 6.

2. A Brief Review of Algebraic Multigrids. AMG methods can be used as independent linear solvers for solving linear systems of the form

$$Ax = b \tag{1}$$

where $A \in \Re^{n \times n}$ is a positive definite matrix and $x, b \in \Re^n$. In practice, they are often used as preconditioners in preconditioned conjugate gradient and/or generalized minimal residual methods. A typical V-cycle AMG is given as follows (by calling MG$(A_0, b, u, 0)$):

ALGORITHM MG(A_k, b_k, u_k, k)

1. If level k is the coarsest level, $u_k = A_k^{-1} b_k$, return
2. $u_k = S_k(A_k, b_k, u_k)$ /* pre-smoothing */
3. $r_k = b_k - A_k u_k$ /* residual calculation */
4. $b_{k+1} = R_k * r_k$ /* restriction */
5. $v_{k+1} = 0$
6. MG$(A_{k+1},\ b_{k+1},\ v_{k+1},\ k+1)$
7. $u_k = u_k + P_k v_{k+1}$ /* coarse grid correction */
8. $u_k = S_k(A_k, b_k, u_k)$ /* post-smoothing. */

Here S_k, R_k, and P_k are the smoothing, restriction, and prolongation operators, respectively.

Application of AMG methods consists of distinct setup and iteration phases. In the setup phase, the transfer operators P_k, R_k are created based on the operator A_k, and the multigrid smoothing operators S_k are set up. Once P_k becomes available, the restriction operator, R_k, is typically defined as $R_k = P_k^T$, and the coarse problem operator is constructed using the Galerkin formula, $A_{k+1} = R_k A_k P_k$. The hybrid Gauss-Seidel smoother [33] is chosen for our numerical study and its performance aspects will be discussed later.

The main idea of multigrid is based on the complementary roles of the multigrid smoothing and coarse-grid correction procedures. In order to have a well-convergent multigrid, the solution error not eliminated well by smoothing must be eliminated well by coarse grid correction. As the prolongation operators define the coarse spaces, they play a critical role in transmitting and eliminating the error modes between different grid levels. In the following subsections we briefly describe two different ways of selecting the coarse variables and the corresponding prolongation operators.

2.1. The Classical Algebraic Multigrid. Setting up the coarse grid structure in the classical algebraic multigrid [24] (RS) is guided by two principles:

P1: Errors not efficiently reduced by smoothing must be approximated well by the range of interpolation, and

P2: the coarse grid correction must efficiently eliminate fine grid error in the range of interpolation.

For problems with M-matrix property **P1** is satisfied by a two-pass coarse grid selection and the Ruge-Stüben interpolation formula [24], while construction of coarse problems using the Galerkin formula allows **P2** to be satisfied.

A typical setup phase for RS is as follows [11]:

RS Setup Phase:

1. Set k = 0.
2. Partition the level k set of points Ω^k into disjoint sets C^k and F^k.
   (a) Set $\Omega^{k+1} = C^k$.
   (b) Define interpolation P_k.
3. Set $R_k = P_k^T$ and $A_{k+1} = R_k A_k P_k$.
4. If Ω_{k+1} is small enough, stop. Otherwise, set $k = k + 1$ and go to step 2.

Step 2 begins by forming a set of dependencies for each point i (a point can be a single unknown in the scalar case or a set of unknowns in the system case) in the current grid level defined by

$$S_i = \{j \neq i : -a_{ij} \leq \alpha \max_{k \neq i}(-a_{ik})\}, \tag{2}$$

where α is the strength threshold for pruning weak couplings between variables. All points in the domain are chosen either as coarse or fine points. A set of interpolation points C_i is then selected for every fine point i. Since errors are more effectively smoothed in the direction of influence, a suitable criterion for selecting C_i is $C_i = S_i \cap C$ (C is the set of all C points.)

Two criteria for choosing the F and C sets are that

P3: For each $i \in F$, each $j \in S_i$ is either in C or $S_j \cap C_i \neq \emptyset$, and

P4: C is a maximal set with the property that if i and j are both coarse points, then $j \notin S_i$ and $i \notin S_j$.

Criterion **P3** says that if i is a fine point, then the points influencing i must either be coarse points or must share a common coarse point with i. **P4** is enforced to keep the coarse grid as small as possible. Since it may not be possible to satisfy both criteria simultaneously, a common practice is to enforce **P3**, guided by **P4**. This commonly takes the form of a two-pass coarse selection, which proceeds by first choosing a maximal independent set of coarse points, followed by patching up the C set to satisfy **P3**.

Once the coarse set C has been determined, the interpolation formula is prescribed as:

$$(P_k u_{k+1})_i = \begin{cases} (u_{k+1})_l & \text{if } i \in C \text{ and } i \text{ and } l \text{ are the same point} \\ \sum_{j \in C_i} \omega_{ij} (u_{k+1})_j & \text{if } i \in F. \end{cases}$$

Details of computing the quantities ω_{ij} can be found in [24].

Note that the RS method was originally developed for problems with M-matrix property, so the problems considered in this paper are out of its theoretical scope. However, the method has been successfully applied to a variety of problems violating the M-matrix property, and we demonstrate that it can, with some success, be applied to the problems under our consideration.

2.2. The Smoothed Aggregation Multigrid. The coarse grid construction for SA differs from that of RS in that, instead of choosing the coarse set as a subset of the current grid nodes (a node can be a single unknown in the scalar case or a set of unknowns in the system case), each fine node is assigned to a unique aggregate. Each aggregate thus corresponds to one or more equations in the coarse grid operator. The first coarsening step is thus to form a collection of mutually disjoint aggregates covering all the fine nodes (however, the equations corresponding to the essential boundary conditions are usually left out). The goal is to have uniformly shaped aggregates. The optimal aggregate size for the standard method has a diameter three, i.e., three nodes along the diameter (small aggregates may result in higher overall complexity, but very large aggregates may result in poor convergence rates unless a nonstandard prolongator smoother is applied, cf. [30]). A basic aggregation procedure proceeds as follows [31, 26]:

SA Aggregation step: (Given a graph representing the node connectivity)
Phase 1 : Form initial set of aggregates: for i = 1 to n,
- a. If node i has been aggregated or it is adjacent to an aggregated node, go to the next node.
- b. Otherwise, select it as a root node, and define a new aggregate as node i plus all its strongly coupled neighbors.

Phase 2 : Put unaggregated nodes into existing aggregates: for i = 1 to n,
- a. If node i has been aggregated, go to the next node.
- b. Otherwise, put it in the aggregate to which one of its neighbors belongs.

Actual implementation can add complexity to this aggregation algorithm. For example, more phases may be needed if aggregate size control is incorporated. Also, different criteria for choosing host aggregates in phase 2 may be prescribed to enhance performance.

The aggregation step is followed by forming the tentative prolongation operator $\tilde{P}_k$, the formula of which, in the simplest case of piecewise constant interpolation, is given as:

$$(\tilde{P}_k u_{k+1})_i = \begin{cases} (u_{k+1})_j & \text{if } i \in \text{aggregate}_j \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

For construction of the tentative prolongation in more general cases we refer to [31]. For solving systems of partial differential equations, aggregation is typically performed on the nodes in the computational grid. Three-dimensional elasticity problems, for example, have three unknowns per node corresponding to displacements in the three orthogonal directions. Thus, in the case of structured hexahedral mesh, each aggregate would ideally contain 27 nodes or 81 unknowns. In addition, instead of using piecewise constant interpolation for each aggregate, rigid body modes or near null space vectors can be provided for defining $\tilde{P}_k$. The number of coarse grid equations for each aggregate is then the number of null space vectors provided. For three-dimensional elasticity, six rigid body modes can be computed from the nodal coordinates which are readily available from most applications.

More robust prolongators can be constructed by smoothing the tentative prolongators. For example, applying a damped Jacobi smoother gives:

$$P_k = (I - \omega D_k^{-1} A_k)\tilde{P}_k, \tag{4}$$

where D_k is the diagonal of A_k, and ω is the damping factor related to the maximum eigenvalue of the scaled matrix $D_k^{-1} A_k$. Convergence theory [29] proves that this

prolongation smoother is needed to guarantee convergence rates only mildly dependent on the grid size.

3. Generalized Smoothed Aggregation. For many applications, the knowledge of a small number of near-kernel components suffices to construct a robust smoothed aggregation solver. However, for difficult problems, it may be beneficial or indeed necessary to generalize the method.

To motivate the GSA method, we first recall the theoretical convergence estimate derived in [30]. Throughout this section we drop the level indices in the notation for the multigrid operators when there is no danger of ambiguity. Let n, m denote the dimensions of the Euclidean spaces associated with the fine and coarse spaces, respectively. We further define $A_S = S^2 A$, where S denotes the prolongator smoother. Then for the two-level smoothed aggregation method employing a pre-smoother with a linear part given by S and a post-smoother with linear part given by

$$S' = I - \frac{\hat{\omega}}{\varrho(A_S)} A_S \tag{5}$$

the following result holds (cf., [30]),

THEOREM 3.1. *Assume that there exists a positive constant* C_{apx} *such that:*

1. *There is a linear mapping,* Q, *of* $\Re^n$ *onto* $\text{Range}(\hat{P})$ *such that*

$$\|(I - Q)u\|^2 \leq \frac{C_{\text{apx}}^2}{\bar{\varrho}(A_S)} \|u\|_A^2 \quad \forall u \in \Re^n. \tag{6}$$

2. *The prolongator smoother,* S, *is symmetric, commutes with* A, *and satisfies* $\varrho(S) \leq 1$.

Let e_i *denote the error after* i *iterations of SA with* $P = S\hat{P}$, *and* $e_i^S = Se_i$ *the final error smoothed by the prolongator smoother. Then it holds that*

$$\|e_i^S\|_A^2 \leq (1 - C)^i \|e_0\|_A^2,$$

where

$$C = \frac{C_{\text{apx}}^{-2} \hat{\omega}(2 - \hat{\omega})}{1 + C_{\text{apx}}^{-2} \hat{\omega}(2 - \hat{\omega})}, \text{ and } \hat{\omega} \text{ is the damping parameter from (5).}$$

As Assumption 2 of Theorem 3.1 is usually easy to satisfy, our attention is focused on minimizing the constant C_{apx} in (6). We note that the left-hand side of (6) depends only on the tentative prolongator, while the prolongator smoother is present only in the form of a bound on the spectral radius of A_S in the denominator of the right-hand side.

The smoothed aggregation method thus possesses two mechanisms for ensuring good convergence properties. One strategy leading to improved constant C_{apx} relies on developing more powerful prolongator smoothers, S, thus forcing $\varrho(S^2 A) \ll \varrho(A)$. This approach had been investigated in [30], where a two-level method with convergence properties independent of the size of the coarse grid was developed.

Another way to improve C_{apx} is based on appropriately enriching the range of the tentative prolongator $\hat{P}$, thus minimizing the left-hand side in (6). This approach can, naturally, be combined with the improvement of S for a compound benefit. Note, however, that in forming the prolongator $P = S\hat{P}$ to be used in the final method, all

columns of $\hat{P}$ must be multiplied by S. Hence, the effect of using an improved S is global in nature. The corresponding smoothing of the coarse space basis functions is thus appropriate for correcting global convergence phenomena tied to the differential equation solved, and the use of nonstandard prolongator smoothers may thus be an overkill if slow convergence of the method is rooted in local phenomena. In contrast, due to the disjoint nature of the aggregate decomposition, the enrichment of the range of $\hat{P}$ can be carried out locally one aggregate at a time.

As the emphasis of this paper is on parallel implementation, we are seeking a procedure which would modify the iterative solver locally in the problematic regions. We will thus focus on local enrichment of the range of the tentative prolongator $\hat{P}$, and refer interested reader to [30] for details on construction of S.

We will consider a decomposition of our computational domain, Ω, into J nonoverlapping subdomains, Ω_i,

$$\Omega = \bigcup_{i=1}^{J} \bar{\Omega}_i, \quad \Omega_i \cap \Omega_j = \emptyset \text{ if } i \neq j. \tag{7}$$

We assume that each $\bar{\Omega}_i$ is a connected cluster of elements. Denote by n_i the number of degrees of freedom in $\bar{\Omega}_i$, by A_i the $n_i \times n_i$ local stiffness matrix corresponding to $\bar{\Omega}_i$, and by N_i the zero-one matrix mapping the degrees of freedom associated with $\bar{\Omega}_i$ into the set of global degrees of freedom in matrix A. Obviously, the stiffness matrix A can be assembled from the local stiffness matrices A_i by:

$$A = \sum_{i=1}^{J} N_i A_i N_i^T. \tag{8}$$

We will utilize a system of large nodal aggregates $\{\mathcal{A}_i\}_{i=1}^{J}$ forming a disjoint covering of the set of all nodes such that all nodes in $\mathcal{A}_i$ lie in $\bar{\Omega}_i$. Such aggregates are easily formed by assigning all the nodes in $\bar{\Omega}_i$ to $\mathcal{A}_i$, and distributing any nodes shared by more than one $\bar{\Omega}_i$ to exactly one of the corresponding aggregates at will.

To formalize this distribution of nodes, for each subdomain $\bar{\Omega}_i$ with number of degrees of freedom n_i, we define a zero-one diagonal matrix I_i of dimension n_i as

$$(I_i)_{dd} = \begin{cases} 1 & \text{if degree of freedom } d \text{ corresponds to a node in } \mathcal{A}_i \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

We further define for each subdomain $\bar{\Omega}_i$ a set of vectors $\{w_j^{(i)}\}_{j=1}^{m_i}$, $w_j^{(i)} \in \text{Range}(I_i)$, and denote by W_i the $n_i \times m_i$ matrix consisting of orthogonal columns $\{w_j^{(i)}\}_{j=1}^{m_i}$. The tentative prolongator $\hat{P} : \Re^m \mapsto \Re^n$ being able to represent exactly all functions in $\{w_j^{(i)}\}_{j=1}^{m_i}$ over $\mathcal{A}_i$ can then be defined by

$$\hat{P}x = \sum_{i=1}^{J} N_i W_i x^{(i)}, \tag{10}$$

where $x^{(i)}$ denotes the segment of coarse degrees of freedom associated with aggregate $\mathcal{A}_i$.

Theorem 3.1 guarantees that the rate of convergence of the two-level method is bounded by $1 - \frac{C_{\text{apx}}^{-2}}{1+C_{\text{apx}}^{-2}}$ under the assumption that there exists $v \in \Re^m$ such that

$$\|u - \hat{P}v\|_{l^2(\bar{\Omega})}^2 \leq \frac{C_{\text{apx}}^2}{\varrho(S^2 A)} \|u\|_A^2 \quad \forall u \in \Re^n. \tag{11}$$

Using the sparsity structure of $\hat{P}$ stemming from disjointness of the aggregates, we can write for any $u \in \Re^n$,

$$\|u - \hat{P}v\|^2_{l^2(\bar{\Omega})} = \sum_{i=1}^{J} \|u - \hat{P}_{*,i} v_i\|^2_{l^2(\mathcal{A}_i)},$$

where $\|\cdot\|_{l^2(\mathcal{A}_i)}$ denotes the restriction of the Euclidean norm to the set of degrees of freedom corresponding to the node in the aggregate $\mathcal{A}_i$, and $\hat{P}_{*,i}$ denotes the super-column of $\hat{P}$ corresponding to aggregate $\mathcal{A}_i$. Similarly, using (8), we write

$$\|u\|^2_A = \sum_{i=1}^{J} \langle A_i N_i^T u, N_i^T u \rangle.$$

Thus, in order to satisfy (11) it suffices to satisfy over all aggregates $\mathcal{A}_i$ the inequality

$$\|u - \hat{P}v\|^2_{l^2(\mathcal{A}_i)} \leq \frac{C^2_{\text{apx}}}{\varrho(S^2 A)} \langle A_i N_i^T u, N_i^T u \rangle \quad \forall u \in \Re^n. \tag{12}$$

Assuming fixed prolongator smoother S, the satisfaction of (12) depends solely on the selection of columns of $\hat{P}$. We are thus led to constructing $\hat{P}$ so that

$$\inf_{v_i \in \Re^{m_i}} \|u - \hat{P}_{*,i} v_i\| = \langle (I - Q_i)u, u \rangle_{l^2(\mathcal{A}_i)} \leq \Lambda_i \|u\|^2_{A_i} \quad \forall u \in \Re^n, \tag{13}$$

where $Q_i = W_i (W_i^T W_i)^{-1} W_i^T$ is the $\Re^{n_i}$-orthogonal projection onto the Range(W_i). Equation (13) can be reformulated as a generalized eigenvalue problem for $I_i(I - Q_i)I_i$ preconditioned by A_i. The constant Λ_i in (13) can be decreased by finding the largest eigenvalue of the generalized eigenvalue problem and adding the corresponding eigenvector to the range of Q_i. This amounts to introducing new columns corresponding to aggregate $\mathcal{A}_i$ into the tentative prolongator $\hat{P}$. This procedure can be repeated until (13) is eventually satisfied with a uniform constant $\Lambda \geq \Lambda_i$ for all $i = 1, \ldots, J$.

To reduce cost associated with the solution of the local generalized eigenvalue problem, we instead solve the local eigenvalue problem for A_i. The following lemma [7] justifies this reduction in cost.

LEMMA 3.2. *Let $w_j^{(i)} = I_i \hat{w}_j^{(i)}$, where $\hat{w}_j^{(i)}$ are the eigenvectors of matrix A_i corresponding to the eigenvalues of A_i that are smaller than $\frac{1}{\Lambda} > 0$. Then*

$$\langle I_i(I - Q_i)I_i u, u \rangle_{\Re^{n_i}} \leq \Lambda \|u\|^2_{A_i} \quad \forall u \in \Re^{n_i}.$$

Note that we only compute the near null space vectors for the subproblems corresponding to subdomains Ω_i created by the domain partitioner (computing the near null space vectors for the global matrix would be prohibitively expensive). The setup cost is still relatively high, but it can be performed in parallel with no interprocessor communication. To keep cost down, the size of each subdomain is limited to a few thousand nodes.

The coarse-space enriching technique we have thus far described in this section is based on the two-level method proposed in [7], which utilizes aggregates $\mathcal{A}_i$ where each aggregate consists of all the nodes in the subdomain Ω_i. This necessitates the use of special prolongator smoothers and special multigrid relaxation procedure in

order to guarantee good convergence. Here we are interested in providing a multilevel generalization. Our current GSA method differs from that in [7] in that a standard aggregation coarsening will be applied on each $\bar{\Omega}_i$. Thus the nodes of Ω_i are subaggregated into a number of small, standard-sized aggregates, $\mathcal{N}_i = \bigcup \mathcal{A}_{ij}$, where $\mathcal{N}_i$ denotes the set of unique nodes corresponding to subdomain $\bar{\Omega}_i$. The eigenmodes of A_i are then restricted to these smaller aggregates, $\mathcal{A}_{ij}$, and the tentative prolongator used in the current GSA can be written as

$$\tilde{P}x = \sum_{i=1}^{J} \sum_{j=1}^{\mu_i} \hat{N}_{ij} W_{ij} x^{(ij)}, \tag{14}$$

where W_{ij} denotes restriction of W_i to $\mathcal{A}_{ij}$ and $\hat{N}_{ij}$ is a zero-one matrix mapping the degrees of freedom in $\mathcal{A}_{ij}$ into the set of global degrees of freedom. Symbol $x^{(ij)}$ denotes the segment of coarse degrees of freedom associated with aggregate $\mathcal{A}_{ij}$.

As the computation of eigenmodes on each subdomain is performed independently of one another, we do not allow aggregation across the subdomain boundary in order to facilitate alignment. Consequently, the number of unknowns on the coarsest grid is at least $J \times \nu$ where J is the number of subdomains and ν is the desired number of null space vectors. When both J and ν are large, the coarsest problem can become too large for direct solution.

Since the set of coarse grid basis functions generated by $\tilde{P}$ is a refinement of that generated by $\hat{P}$, we trivially obtain

$$\sum_{j=1}^{\mu_i} \min_{x^{(ij)}} \|u - \tilde{P}x^{(ij)}\|^2_{l^2(\mathcal{A}_{ij})} \leq \min_{x^{(i)}} \|u - \hat{P}x^{(i)}\|^2_{l^2(\mathcal{A}_i)} \leq \Lambda \|u\|^2_{A_i}.$$

Thus the standard convergence theory for the smoothed aggregation method [29] can be used to guarantee convergence of the resulting method. As a result of using the standard aggregates, $\tilde{P}$ can approximate well not only the computed set of eigenmodes, $\{w_j^{(i)}\}_{j=1}^{m_i}$, but also a much richer subspace consisting of any modes with locally similar character. This can reduce the number of eigenmodes that need to be computed. Another benefit of using GSA, compared to the method of [7], is the reduced amount of inter-processor communication. The method of [7] was primarily intended for serial environment. Its multigrid relaxation is based on polynomial smoothing, which makes it suitable for parallel implementation. But its use of the more powerful prolongation smoothing, required to achieve optimal convergence rates, results in larger cross-processor overlap in its coarse-level basis functions. This leads to increased communication necessary in setting up the coarse-level matrices. In contrast, GSA, with its use of the standard prolongation smoother, reduces communication and is certain to result in higher flop-rates in a massively parallel environment.

For completeness, we include a summary of the GSA multilevel coarsening procedure below. Let l denote the level index ($l = 0$ corresponds to the finest grid). Furthermore, let us denote by $\mathcal{N}_i^l$ the set of unique nodes on level l which is associated with subdomain $\bar{\Omega}_i$, and by μ_i^l its number. Symbol $\mathcal{A}_j^l$ denotes the aggregates on level l. Given a decomposition of the domain (7) and the corresponding subdomain matrices, A_i, GSA coarsening proceeds as follows:

1. Initialize one large aggregate, $\mathcal{A}_i^0 = \mathcal{N}_i$, per subdomain $\bar{\Omega}_i$, $\quad i = 1, \ldots, J$, distributing any shared nodes so that the resulting $\mathcal{N}_i$ are mutually disjoint. Set $\mu_i^0 = card(\mathcal{N}_i), i = 1, \ldots, J$.

2. For each local matrix $A_i, i = 1, \ldots, J$, compute a set of m_i eigenmodes, W_i^k, smaller than Λ. Set $l = 0$.
3. Perform standard SA aggregation independently over each $\mathcal{N}_i^l$ to produce $\{\mathcal{A}_{ij}^l\}$ such that $\mathcal{N}_i^l = \bigcup_j \mathcal{A}_{ij}^l$. Split W_i^l into segments, W_{ij}^l, corresponding to aggregates.
4. Construct operator $\tilde{P}_l$ as in (14). Simultaneously, compute W_i^{l+1}, the coarse-level representations of W_i^l such that $W_i^l = \tilde{P} W_i^{l+1}$.
5. Construct smoothed prolongator, $P_l = (I - \omega D_l^{-1} A_l)\tilde{P}_l$.
6. Construct coarse-level operator, $A_{l+1} = P_l^T A_l P_l$
7. If $l + 1$ is the coarsest level, stop.
   Otherwise, set $\mathcal{N}_i$ to be the set consisting of global numbers indexing the aggregates on level l, $\mu_i^{l+1} = card(\{\mathcal{A}_{ij}^l\})$, set $l = l + 1$, and go to step 3.

The construction of $\tilde{P}$ in step 4 (and in (14)) is modified in practice by orthogonalizing the columns of W_{ij}^l over the local aggregates $\mathcal{A}_{ij}^l$. Such modification does not change the range of the constructed tentative prolongator, but produces better-conditioned coarse-level matrices (cf. [31]). Also, it can be conveniently combined with the computation of the coarse-level representation W_i^{l+1} (cf. [29]).

We note that a coarse space enrichment similar to the one described here has been used in the context of classical AMG in the AMGe class of methods [6, 10], and also by Mandel [21, 22] in the context of two-level domain decomposition suitable for p-version finite elements. Reference [13] describes a coarse-space enrichment for a non-smoothed aggregation method using very small element agglomerates, under the assumption that the local element matrices are available. The recently introduced class of adaptive smoothed aggregation methods (αSA, cf. [8]) approaches local coarse-space enrichment by run-time evaluation of the error components of the iteration and subsequent incorporation of these modes into its multigrid transfer and coarse operators.

4. Parallel Implementation Issues. The original coarse grid selection algorithms for both RS and SA are inherently sequential. In the solution phase, the primary factor for parallel efficiency is the use of parallel and robust smoothers. This section briefly reviews efforts pertaining to these two issues.

4.1. Parallel Coarsenings for RS Algebraic Multigrid. Several parallel coarsening algorithms have been proposed and tested successfully - the parallel RS coarsening [14], the CLJP coarsening [11], and the hybrid (or Falgout) coarsening. The parallel RS coarsening first performs independent local coarsening within each processor, followed by coarsening on the processor boundaries. The CLJP coarsening is a form of a parallel maximal independent set algorithm [18, 20] modified to enforce the coarsening criteria. The hybrid coarsening uses the RS coarsening interior to the processors and CLJP coarsening on the boundaries. The relative performance of these algorithms varies dependent on the applications. Since the CLJP algorithm generally gives better results for unstructured grid problems, it is used in our numerical experiments. For other strategies which have been proved successful in parallelizing RS-based multigrid methods we refer the readers to [19, 15].

4.2. Parallel Coarsenings for SA Algebraic Multigrid. Reference [26] describes three parallel aggregation algorithms. The coupled aggregation scheme proceeds by having each processor first separating all nodes assigned to it into the interior and border nodes. Border nodes are those which share a grid edge with a nodes on

another processor. Subsequently, this scheme aggregates the border nodes before aggregating the interior nodes. This scheme has the disadvantage that some processors may have to wait until other processors have completed aggregating their border nodes. The worst case scenario would require a wait time proportional to $P^{1/3}$ where P is the number of processors. However, the scheme can give significantly better aggregates than the other approaches considered.

The second parallel aggregation scheme is based on maximally independent set (MIS). In brief, the widely studied parallel MIS algorithms can be applied to the square of the matrix representing grid connectivities. This scheme has the disadvantage that it requires computing the square of a matrix.

The third scheme, which is our choice in our numerical studies, is called the decoupled aggregation scheme. This scheme lets each processor form its set of aggregates independently of one another. It is very simple to implement and it generally gives good performance. However, its performance depends a great deal on the quality of the domain partitioner. Furthermore, not allowing the aggregate to cross the processor boundaries can result in slightly higher overall grid complexity (the total number of grid nodes in the grid hierarchy) and larger coarsest grid size when the number of processors is large. We remedy the latter by triggering a processor aggregation step (forming processor connectivity graph and grouping processors into aggregates) when the coarsest grid is too large.

4.3. Parallel Smoothers. Smoothers play a critical role in the AMG solution phase. On sequential computers, Gauss-Seidel (GS) is usually the smoother of choice. On massively parallel computers, however, the choice of smoother is less clear. The GS smoother can be modified for higher degree of parallelism by combining it with the Jacobi smoother. By preserving the sequential nature of GS within each processor and exploiting the parallel nature of Jacobi across processor boundaries, a hybrid GS-Jacobi smoother has been demonstrated to be quite effective on small number of processors. For massive parallelism, it may suffer from slow convergence rates. Another approach is to improve GS's parallelism by using multi-coloring [1]. A limitation of this approach is that there needs to be sufficient number of unknowns on each processor, and is thus not an effective smoother on the coarse grids. Other efforts include the use of polynomial smoothers [2] and Krylov smoothers preconditioned with Jacobi, block Jacobi, or overlapped Schwarz, all with limited success.

In our numerical experiments, we use a variant of the hybrid GS-Jacobi smoother given in [33]. The idea is to recover fast convergence rates by using under-relaxation in GS and/or damped Jacobi across processors. The relaxation parameters can be computed by a few conjugate gradient iterations.

5. Test Problems from Structural Mechanics. The test problems are taken from the ALE3D code developed at the Lawrence Livermore National Laboratory. The implicit integration algorithm in ALE3D for solid materials is based on an updated Lagrangian displacement-based formulation. A set of nonlinear equations is solved for equilibrium using the state and configuration at the end of a time step. During the time integration, nonlinearities arise because of the material response and configuration changes.

The basic equations are the momentum equations derived by starting with the local dynamic equilibrium relation in a form suitable for a solid body

$$\rho \mathbf{u}_{tt} = \nabla \cdot \boldsymbol{\sigma} + \rho \mathbf{b} \tag{15}$$
$$\mathbf{n} \cdot \boldsymbol{\sigma} = \mathbf{g}$$

where $\mathbf{u}$ is the displacement vector, $\boldsymbol{\sigma}$ is the stress tensor, $\mathbf{b}$ is the body force vector per unit mass, ρ is the density, and $\mathbf{g}$ is the surface traction. Applying the weak form and then integrating by parts yield

$$\int_V \rho \mathbf{u}_{tt} \cdot \delta\mathbf{u} dV + \int_V \boldsymbol{\sigma} : (\nabla \delta\mathbf{u})^T dV - \int_S \mathbf{n} \cdot \boldsymbol{\sigma} \cdot \delta\mathbf{u} dV - \int_V \rho \mathbf{b} \cdot \delta\mathbf{u} dV = 0 \tag{16}$$

where the colon signifies a double inner product, V is the volume, and S is the surface of the body. Time integration is performed by using the full Newton-Raphson iteration scheme represented by

$$\frac{dP}{d\mathbf{u}} \cdot d\mathbf{u} = -P \tag{17}$$

where P is the left hand side of Equation (16), and (A is the surface area)

$$dP = \int_V \left[d\boldsymbol{\sigma} - (\nabla d\mathbf{u})^T \cdot \boldsymbol{\sigma} + (\nabla \cdot d\mathbf{u}) \boldsymbol{\sigma} \right] : (\nabla \delta\mathbf{u})^T dV - \tag{18}$$

$$\int_S \left[\frac{d\mathbf{g}}{d\mathbf{u}} \cdot d\mathbf{u} + \frac{1}{A} \mathbf{g} \frac{dA}{d\mathbf{u}} \cdot d\mathbf{u} \right] \cdot \delta\mathbf{u} dS - \tag{19}$$

$$\int_V \rho \left[\frac{d\mathbf{b}}{d\mathbf{u}} \right] \cdot d\mathbf{u} + \int_V \rho \left[\frac{d\mathbf{u}_{tt}}{d\mathbf{u}} \right] \cdot \delta\mathbf{u} dV.$$

This displacement-based Jacobian equation is solved with the finite element method using linear elements on a hexahedral mesh. At each time step, corrections are added to the previous estimate of the nodal displacements. The nodal displacements are then used to calculate displacement gradients and strain increment tensors that are passed to the constitutive model.

The constitutive relation we use cannot be written simply in terms of stresses and elastic properties. It also depends on current loading. For a forward gradient stress integration, the material stiffness would take the form:

$$C_{ijkl} = L_{ijkl} - \frac{L_{ijrs} P_{rs} P_{tu} L_{tukl}}{h + P_{rs} L_{rstu} P_{tu}}, \quad i, j, k, l, r, s, t, u = 1 \cdots 3 \tag{20}$$

where L is the standard fourth order elasticity tensor, P is the derivative of the yield surface with respect to the stress tensor and h is the material strain hardening.

5.1. The Spherical Shell Problem. The domain for the first problem is an octant of a spherical shell depicted in Figure 1. The shell has three layers having steel on the outside and inside layers and polycarbonate in the middle. Both elasticity and plasticity are present in the constitutive model. The forcing function is the small amount of energy injected into the polycarbonate layer, causing the polycarbonate to expand against the inner and outer layers. Equation (20) can be simplified to $C = L$ in this case as there is no plasticity.

5.2. The Crystal Plasticity Problem. This test problem (called TBar and shown in Figure 2) uses a crystal plasticity constitutive model given by

$$C_{ijkl} = L_{ijkl} - \sum_{a=1}^{12} \sum_{b=1}^{12} L_{ijmn} Q^a_{mn} R^b_{rs} L_{rskl} \tag{21}$$

where Q^a and R^b are each second order tensors for each of the 12 slip systems. As a result, the global stiffness matrix is slightly nonsymmetric. Boundary conditions for this test problem are forces applied at one end in the direction of the bar length and the other end is tied down to prohibit displacement in the direction of the bar length.

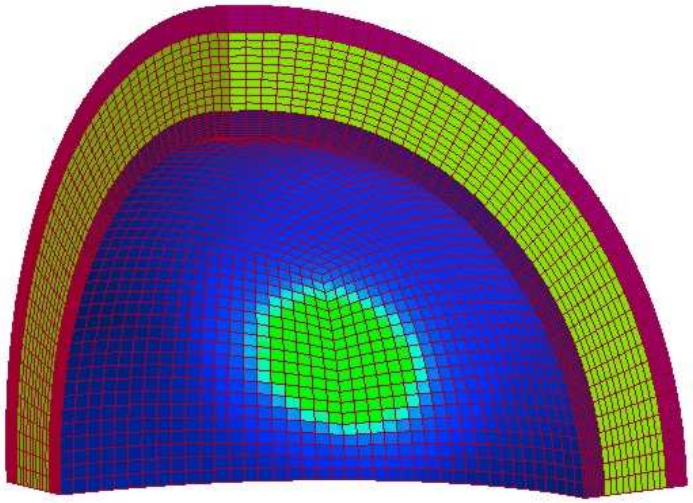

FIG. 1. *The Spherical Shell Problem*

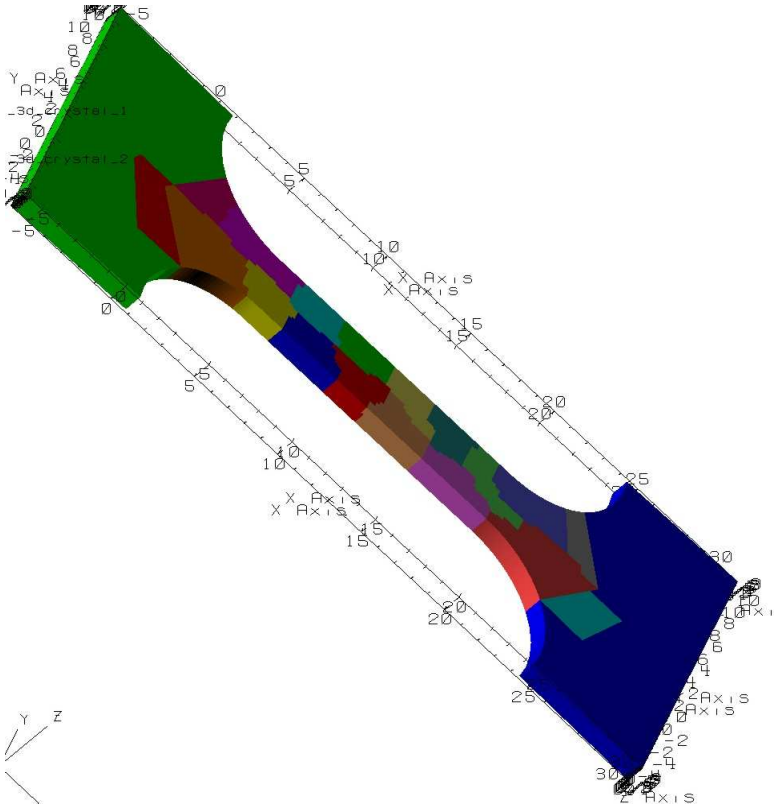

FIG. 2. *The TBar Problem (subdomains in different colors)*

5.3. The Bifurcation Problem. This test problem (called RBar and shown in Figure 3) involves bifurcation which occurs when the force required to stretch the perfectly uniform rod reaches a peak. The load is the axial stress times the cross sectional area. During deformation, the stress first increases linearly with strain as an elastic material. When stretched to a stress level known as the yield point, dislocation mechanisms in the metal allow rearrangement of the atoms and result in permanent deformation of the material. This is plastic flow. During plastic flow the rate of change of stress with strain is typically 2 to 3 orders of magnitude less than during the initial elastic deformation. This rate typically decreases continuously during deformation. Also, during plastic flow the volume of the material is roughly constant leading to a measurable decrease in the bar diameter as it is stretched.

Since the load is the stress times the cross section area, the load rate is comprised of two terms containing the increasing stress rate and the decreasing area rate. For most metals the area rate contribution overtakes the stress rate contribution in the strain range from 5% to 30% extension. This is the peak load and corresponds to the Considere [12] condition. Up to the peak load, energy considerations require that the bar deform uniformly. At the peak load the incremental energy increase applied to extend the bar is the same whether it continues stretching uniformly or some nonuniform strain rate pattern emerges. If a nonuniform strain rate pattern emerges, it is not unique.

It is the behavior at this instance of nonunique solution that is challenging to the solver. If everything is geometrically perfect and the low order digits are all the same, the bar can continue to deform uniformly. This has occasionally been seen with direct solvers. If the numerical perturbations are not negligible, a nonuniform deformation pattern will emerge triggered by the numerical imperfections. The energy increase and the load at the nodes of the finite element mesh are nearly the same regardless of the mode selected.

In the physical experiments, the real materials are not perfect and the effects of the grip ends holding the specimen generally force a nonuniform deformation pattern whereby the diameter reduces faster in one location. The second order work terms at the Considere point are such that these localized deformation modes are favored over continued uniform deformation. Once the localized modes begin, the bar is energetically favored to continue deforming in the localized mode. This eventually leads to what is known as a necking behavior and fracture of the bar (see Figure 4).

6. Numerical Experiments. The numerical experiments are run on two IBM SP2 computers (to be called SP2-White and SP2-Blue from this point on) at the Lawrence Livermore National Laboratory. The SP2-White computer has more than 500 nodes with 16 gigabyte of main memory per node. Each node has 16 Power3 processors running at 375 MHz each. The SP2-Blue computer has 264 nodes with 4 PowerPC 604e processors and 1.5 gigabytes of memory per node. Due to the scarce availability of large number of processors (1000 or more) and the large disk space requirement for storing the finite element meshes, our numerical experiments are limited to a maximum of 512 processors. For the spherical shell problem, some performance data were collected for RS and SA on 4000 SP2-White processors. The largest run for RS is about 300 millions unknowns which consumes 200 iterations and 2000 seconds for each solve. For SA we were able to successfully run even larger problem (610 million unknowns, which takes about 500 iterations and 1500 seconds) due to its lower memory requirements.

For both RS and SA, we prescribe one sweep of hybrid SSOR-Jacobi for both pre-

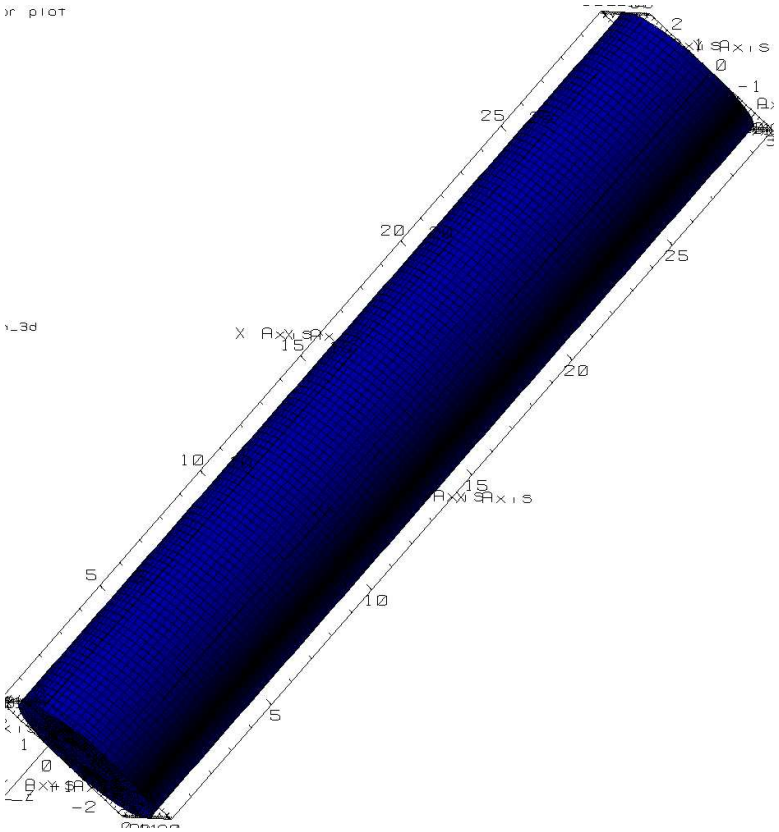

FIG. 3. *The RBar Problem (cylindrical computational domain)*

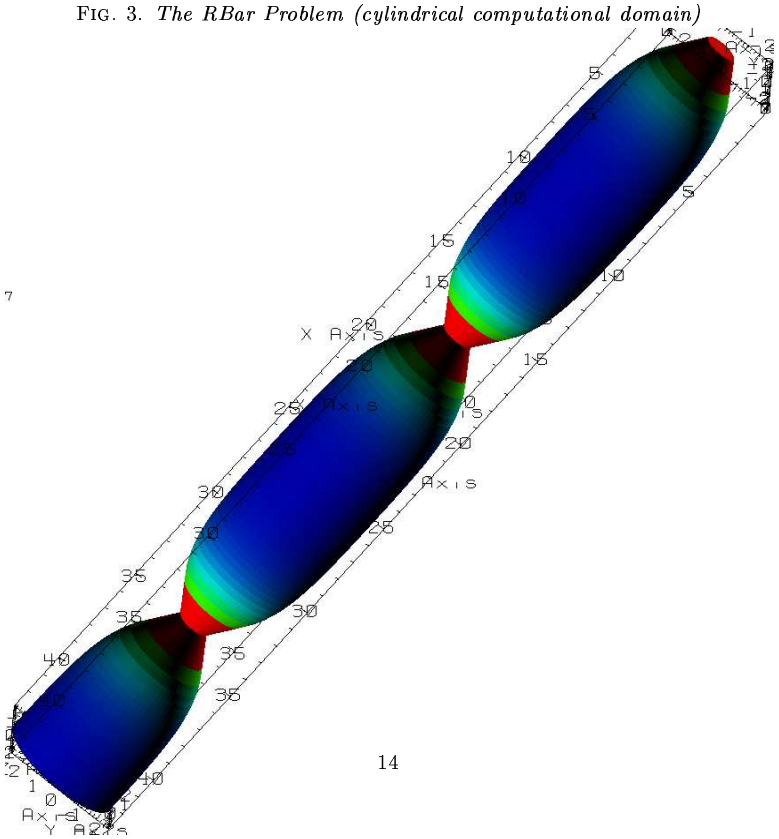

FIG. 4. *The RBar Problem (a deformation configuration)*

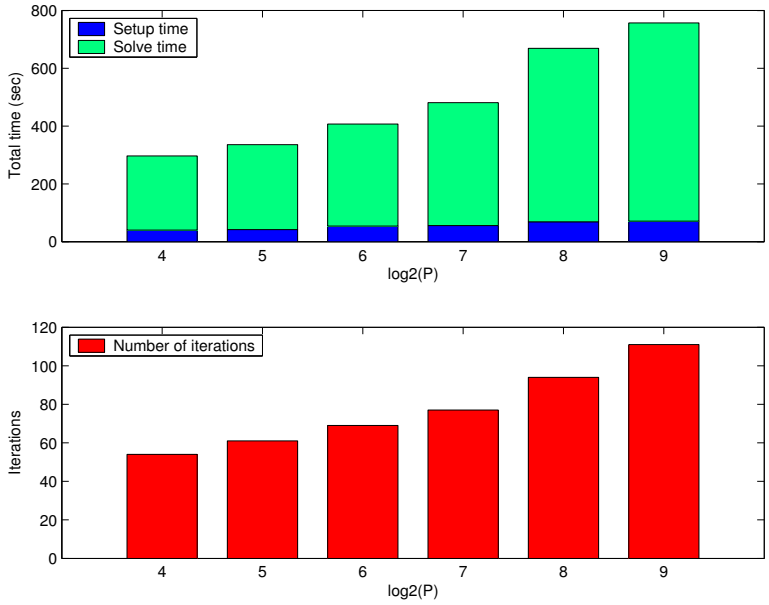

FIG. 5. *Spherical Shell Problem Using RS Preconditioner*

and post-smoothing. For RS, we choose different strength thresholds for best results on different test problems. For SA, the strength threshold is 0.0 and no prolongation smoothing is used for all test problems. In addition, the near null space vectors (the 6 rigid body modes) are computed from the nodal coordinates from our finite element package. For GSA, additional near null space information is calculated from the element stiffness matrices.

6.1. Scalability Study for the Spherical Shell Problem. We study the parallel performance of the spherical shell problem on the SP2-White computer using the conjugate gradient method with RS and SA as preconditioners. Each processor has about 100K unknowns, yielding more than 51 million unknowns on 512 processors. The entire simulation takes about 10 linear solves, and the timing results are the averages of all linear solves. Convergence criterion for the conjugate gradient is 10^{-6}. The best strength threshold for RS is around $0.9 - 0.99$ for this problem. We use 0.95 in our experiment. With this threshold the grid and operator complexities (grid complexity is the ratio of the total number of rows in all A_k's with respect to A_0 while the operator complexity is the ratio of the total number of nonzeros in all A_k's with respect to the number of nonzeros in A_0) are approximately 2.1 and 2.5, respectively. The corresponding grid and operator complexities for SA are 1.08 and 1.15, respectively. The timing results and iteration counts are shown in Figure 5 and 6.

These results show that SA is reasonably scalable (that is, the total solution time grows slowly with the number of processors given the same problem size per processor), while RS appears to be somewhat less scalable due mainly to the faster convergence rate deterioration with larger problem size (but RS gives much better solution times than SA when the number of processors is small.) For SA, we observe a substantial increase in the solution time going from $P = 128$ to $P = 256$ even though the iteration counts stay about the same. After eliminating the cause attributing to

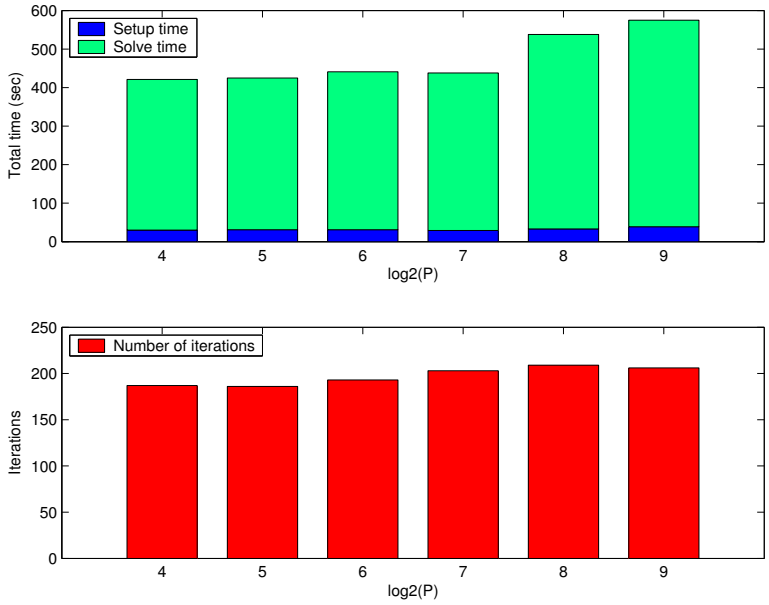

FIG. 6. *Spherical Shell Problem Using SA Preconditioner*

load imbalance, we conclude the performance deterioration is probably due to higher memory access conflicts for larger test problems.

6.2. Scalability/Robustness Study for the Tbar Problem. Numerical experiments for the Tbar problem are performed on the SP2-Blue computer. Each processor has about 30K unknowns, yielding a total of more than 14 million unknowns for 512 processors. The generalized minimal residual (GMRES [25]) method with a restart size of 200 is used for this problem due to nonsymmetry. The prescribed convergence criterion is 10^{-8}. We use 0.9 as strength threshold for RS. With this threshold the grid and operator complexities are about 2.1 and 3.4, respectively, for all processor configurations. The grid and operator complexities for SA are 1.09 and 1.13, respectively. The numerical results given in Figure 7 and 8 are for the first linear solve. We observe that starting from the second linear solve RS fails to converge (stagnation occurs when the relative residual norm drops to about 10^{-2}), while SA takes many more iterations. We demonstrate in Figure 9 and 10 that GSA helps to improve the robustness of SA.

From Figure 7 and 8, we observe that RS takes many more iterations than SA to converge even for the first linear solve. From examining the convergence history (not shown here) we further notice that the iteration count for RS would have been reduced by a factor of 4 if the convergence criterion is raised to 10^{-6}. We conclude that, for low accuracy runs, RS and SA give comparable performance. Similarly, the residual norm curves in Figure 10 shows that SA would have been competitive with GSA for low accuracy runs. For higher accuracy runs and in later time steps when the effect of crystal plasticity becomes more prominent, GSA performs the best.

6.3. Robustness Study for the Rbar Problem. For the Rbar problem, our focus is on the robustness of the preconditioners. Since similar scalability behaviors (as the spherical shell and Tbar problems) are observed here, we deliberately omit the

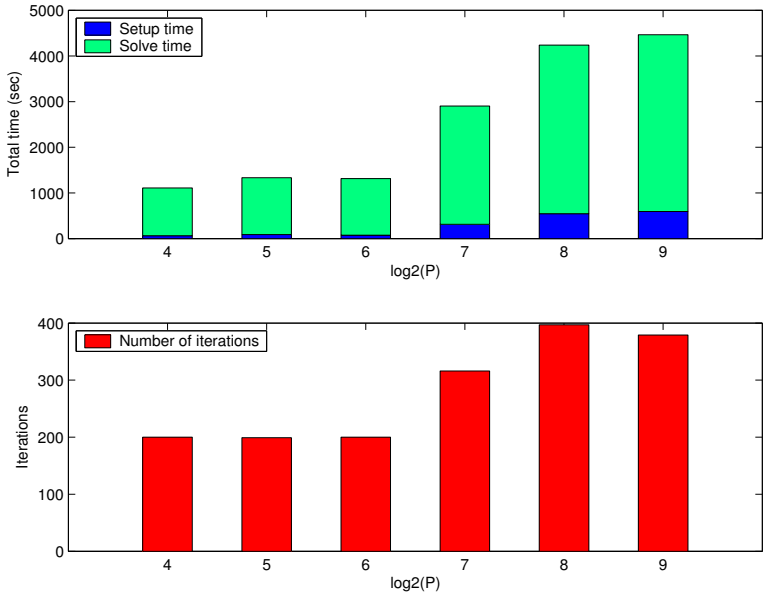

FIG. 7. *Tbar Problem Using RS Preconditioner*

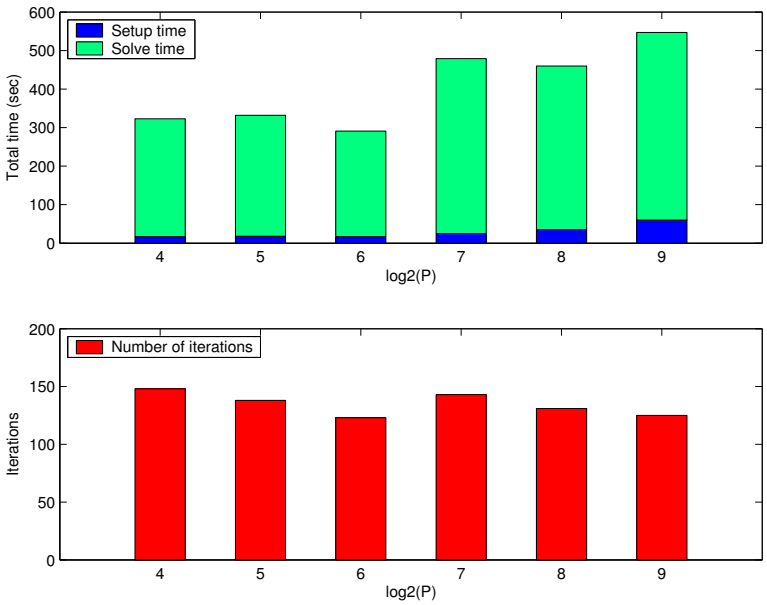

FIG. 8. *Tbar Problem Using SA Preconditioner*

scalability results. The concern for solver robustness stems from the observation that convergence deteriorates rapidly in later time steps when non-uniform deformation occurs. To illustrate the relative performance of the different solvers, we use a test problem with 380K unknowns running on 64 processors of the SP2-Blue computer. Again, we use GMRES(m) with $m = 100$ and convergence criterion 10^{-8}. To reduce

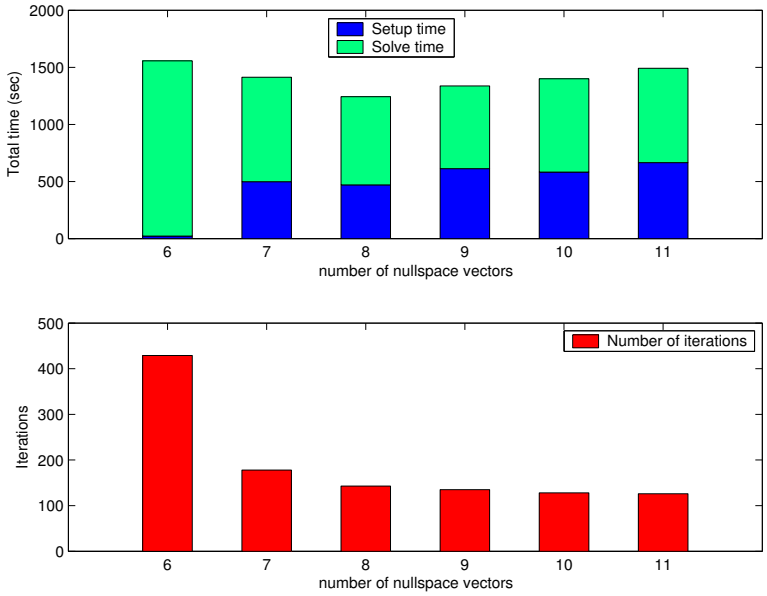

FIG. 9. *Tbar Problem Using GSA Preconditioner*

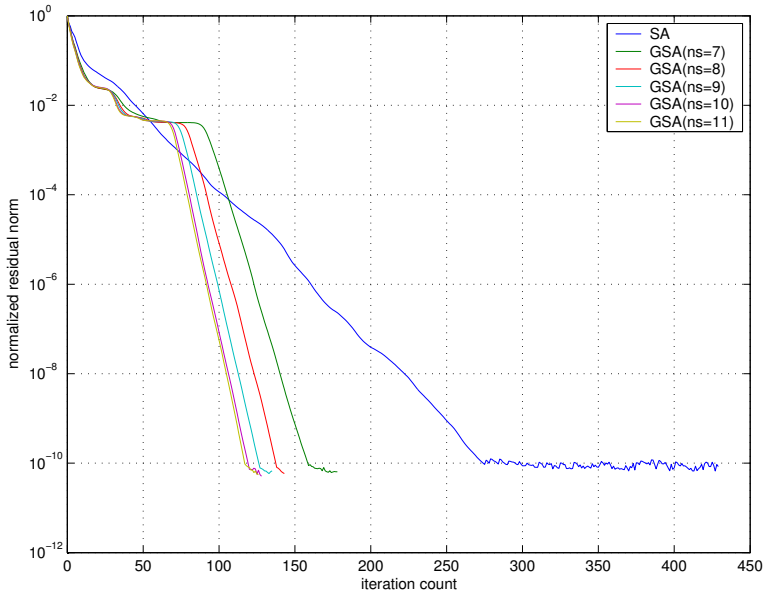

FIG. 10. *Convergence History of Tbar Problem Using GSA Preconditioner*

the eigendecomposition time in GSA, 128 subdomains are used, with each subdomain having roughly about 3000 unknowns. The number of grid levels and near null space vectors for GSA are 3 and 12 (ns=12), respectively, yielding the coarsest grid size of about 1500.

Timing results for all three AMGs are given in Figure 11 up to the 12th time step. Each time step involves 3 to 6 linear solves, and the data given here are the

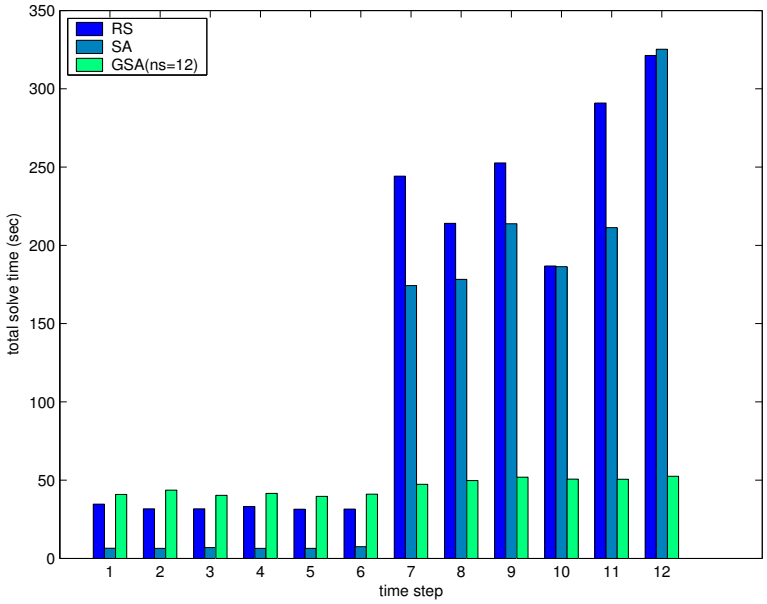

FIG. 11. *Rbar Problem With All Three Preconditioners*

averages. We observe that in the first 6 time steps when the problem is still well-behaved, SA without enriching the prolongation operator is adequate. Beyond the 12th time step, both RS and SA fail to converge, but GSA continues to march on in time with acceptable convergence rates, as depicted in Figure 12. We observe from Figure 12 that a large fraction of the setup time is in the eigendecomposition. We comment that this setup time can be improved by noting that the first 6 near null space vectors are already available and only 6 additional ones are to be computed. In the current implementation, we have not taken advantage of this fact.

7. Summary and Conclusion. In this paper we demonstrate the feasibility of using algebraic multigrid for solving large-scale structural mechanics problems on massively parallel processors.

For difficult structural mechanics problems, the smoothing and coarsening procedures used by the standard algebraic multigrid methods may fail to satisfactorily capture all significant low energy modes of the error. Although the standard SA can guarantee approximation of a general set of functions, it requires these functions to be known before its setup procedure commences. The RS coarsening faces an additional hurdle - even if the set of low energy modes were a priori known, an approximation of a general set of functions is an open problem for RS interpolation. Local prolongation enriching methods such as GSA help to alleviate these problems. Superior convergence has been observed for GSA compared to the RS and SA. Some shortcomings of GSA are the expensive eigendecomposition in the preprocessing step and the potentially large coarsest grid problem. Faster eigensolvers and faster parallel direct solvers are critical for this class of efficient and robust multigrid solvers. Future effort should focus on improving efficiency by reducing memory traffic.

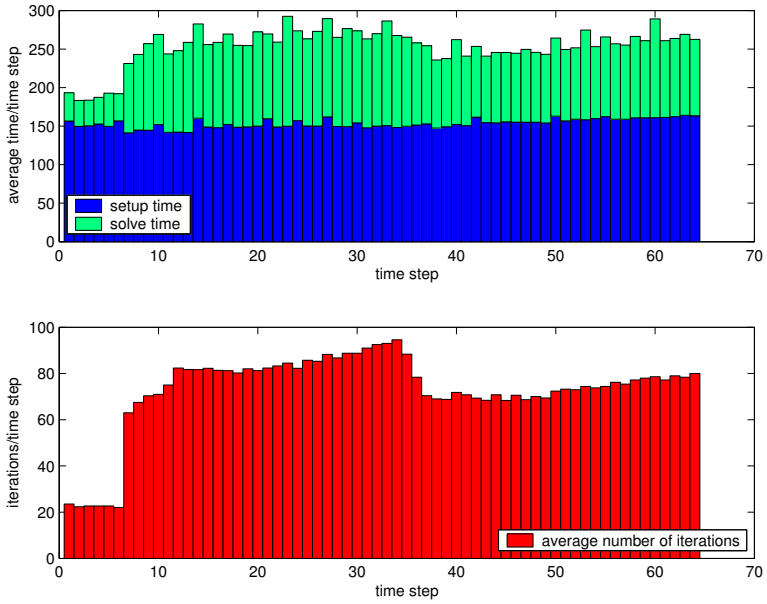

FIG. 12. *Rbar Problem Using GSA Preconditioner*

[1] M. F. Adams, *A Distributed Memory Unstructured Gauss-Seidel Algorithm for Multigrid Smoothers,* ACM/IEEE Proceedings of SC2001: High Performance Networking and Computing, Nov 2001.

[2] M. F. Adams, M. Brezina, J. J. Hu, and R. S. Tuminaro, *Parallel Multigrid Smoothing: polynomial versus GS,* J. Comp. Phys., Vol. 188, No. 2, pp. 593-610, 2003.

[3] *ALE3D Theory Manual,* UCRL-PRES-203313, Lawrence Livermore National Laboratory.

[4] P. B. Bochev, C. J. Garasi, J. J. Hu, A. C. Robinson, and R. S Tuminaro, *An Improved Algebraic Multigrid Method for Solving Maxwell's Equations,* SIAM J. Sci. Comput., Vol. 25, No. 2, pp. 623-642, 2003.

[5] A. Brandt, S. F. McCormick, and J. W. Ruge, *Algebraic Multigrid (AMG) for Automatic Multigrid Solutions with Application to Geodetic Computations*, Technical Report, Inst. for Computational Studies, Fort Collins, Colorado, October 1982.

[6] M. Brezina, A. J. Cleary, R. D. Falgout, V. E. Henson, T. A. Manteuffel, S. F. McCormick, and J. W. Ruge, *Algebraic Multigrid Based on Element Interpolation (AMGe)*, SIAM J. Sci. Comput., Vol. 22, No. 5, pp. 1570-1592, 2000.

[7] M. Brezina, C. Heberton, and P. Vaněk, *An Iterative Method with Convergence Rate Chosen A Priori,* UCD/CCM Report, No. 140, Center for Computational Mathematics, University of Colorado at Denver, Denver, CO., 1999.

[8] M. Brezina, R. D. Falgout, S. MacLachlan, T. A. Manteuffel, S. F. McCormick and J. W. Ruge. *Adaptive Smoothed Aggregation(α(SA)),* SIAM J. Sci. Comp., Vol. 25, No. 6, pp. 1896-1920, 2004.

[9] M. Brezina and P. Vaněk, *A Black-box Iterative Solver based on a Two-level Schwarz Method,* Computing, 63 (1999), pp. 233-263.

[10] T. Chartier, R. D. Falgout, V. E. Henson, J. E. Jones, T. A. Manteuffel, S. F. McCormick, J. W. Ruge and P. S. Vassilevski, *Spectral AMGe (ρAMGe)*, SIAM J. Sci. Comput., Vol. 25, No. 1, pp. 1-26, 2003.

[11] A. J. Cleary, R. D. Falgout, V. E. Henson and J. E. Jones, *Coarse Grid Selection for Parallel Algebraic Multigrid,* Proc. of the Fifth International Symposium on Solving Irregularly Structured Problems in Parallel, Vol 1457, Lecture Notes in Computer Science, pp. 104-115, 1998.

[12] M. Considère, *Die Anwendung von Eisen und Stahl bei Konstruktionen*, Gerold-Verlag, Wien 1888.

[13] J. Fish and V. Belsky, *Generalized Aggregation Multilevel Solver*, International Journal for Numerical Methods in Engineering, Vol. 40, pp. 4341-4361, 1997.

[14] R. D. Falgout and U. M. Yang, *hypre: a Library of High Performance Preconditioners,* Computational Science - CARS 2002 Part III, Lecture Notes in Computer Science, Vol. 2331, pp. 632-641, 2002.

[15] G. Haase, M. Kuhn, and S. Reitzinger, *Parallel Algebraic Multigrid Methods on Distributed Memory Computers,* SIAM J. Sci. Comput., Vol. 24, No. 2, pp. 410-427, 2002.

[16] V. E. Henson and U. M. Yang, *BoomerAMG: a Parallel Algebraic Multigrid Solver and Preconditioner,* Applied Numerical Mathematics, Vol. 41, pp. 155-177, 2002.

[17] E. W. Jenkins, C. E. Kees, C. T. Kelley, and C. T. Miller, *An Aggregation-based Domain Decomposition Preconditioner for Groundwater Flow,* SIAM J. Sci. Comput., Vol. 23, No. 2, pp. 430-441, 2001.

[18] M. Jones and P. Plassman, *A Parallel Graph Coloring Heuristic,* SIAM J. Sci. Comput. 14, pp. 654-669, 1993.

[19] A. Krechel and K. Stüben, *Parallel Algebraic Multigrid Based on Subdomain Blocking,* Parallel Computing, Vol. 27, pp. 1009-1031, 2001.

[20] M. Luby, *A Simple Parallel Algorithm for Maximal Independent Set Problem,* SIAM J. on Computing 15, pp. 1036-1053, 1986.

[21] J. Mandel, *Adaptive Iterative Solvers in Finite Elements,* in *Solving Large-scale Problems in Mechanics*, editor: M. Papandrakis, John Wiley & Sons Ltd., pp. 65-86, 1993.

[22] J. Mandel, *Iterative Methods for p-version Finite Elements: Preconditioning Thin Solids*, Comput. Methods Appl. Mech. Engrg., Vol. 133, pp. 247-257, 1996.

[23] G. Robinson, *Parallel Computational Fluid Dynamics on Unstructured Meshes Using Algebraic Multigrid*, in *Parallel Computational Fluid Dynamics*, Editor: R. B. Peltz, A. Ecer and J. Häuser, Elsevier Science Publishers B.V., Vol 92, 1993.

[24] J. W. Ruge and K. Stüben, *Algebraic Multigrid,* Multigrid Methods, Frontier in Applied Mathematics, pp. 73-130, SIAM, 1987.

[25] Y. Saad and M. H. Schultz, *GMRES : A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems,* SIAM J. Sci. Stat. Comput. 7, pp. 856-869 (1986).

[26] C. H. Tong and R. S. Tuminaro, *Parallel Smoothed Aggregation Multigrid : Aggregation Strategies on Massively Parallel Machines,* Proceedings of Supercomputing, November 2000.

[27] P. Vaněk, *Acceleration of Convergence of a Two-level Algorithm by Smooth Transfer Operators*, Appl. Math., Vol 37, pp 265-274, 1992.

[28] P. Vaněk, *Fast Multigrid Solvers*, Appl. Math., Vol 40, pp 1-20, 1995.

[29] P. Vaněk, M. Brezina, and J. Mandel, *Convergence of Algebraic Multigrid Based on Smoothed Aggregation,* Numerische Mathematik, Vol. 88, No. 3, pp. 559-579, 2001.

[30] P. Vaněk, M. Brezina, and R. Tezaur, *Two-grid Method for Linear Elasticity on Unstructured Meshes,* SIAM J. Sci. Comp., Vol. 21, No. 3, pp. 900-923, 1999.

[31] P. Vaněk, J. Mandel, and M. Brezina, *Algebraic Multigrid Based on Smoothed aggregation for Second and Fourth order problems*, Computing, Vol. 56, pp. 179-196, 1996.

[32] P. Vaněk, R. Tezaur, M. Brezina, and J. Křížková, *Two-level Method on Unstructured Meshes with Convergence Rate Independent of the Coarse-space Size,* Technical Report UCD-CCM-035, University of Colorado, 1995.

[33] U. M. Yang, *On the Use of Relaxation Parameters in Hybrid Smoothers,* Numer. Lin. Alg. Appl., Vol. 11, pp. 155-172, 2004.