SUBMITTED TO SIAM JOURNAL ON MATRIX ANALYSIS AND APPLICATIONS, SEPTEMBER 2004

REDUCING COMPLEXITY IN PARALLEL ALGEBRAIC MULTIGRID PRECONDITIONERS

HANS DE STERCK[†§], ULRIKE MEIER YANG[‡¶], AND JEFFREY J. HEYS[‖*]

Abstract. Algebraic multigrid (AMG) is a very efficient iterative solver and preconditioner for large unstructured linear systems. Traditional coarsening schemes for AMG can, however, lead to computational complexity growth as problem size increases, resulting in increased memory use and execution time, and diminished scalability. Two new parallel AMG coarsening schemes are proposed, that are based on solely enforcing a maximum independent set property, resulting in sparser coarse grids. The new coarsening techniques remedy memory and execution time complexity growth for various large three-dimensional (3D) problems. If used within AMG as a preconditioner for Krylov subspace methods, the resulting iterative methods tend to converge fast. This paper discusses complexity issues that can arise in AMG, describes the new coarsening schemes and examines the performance of the new preconditioners for various large 3D problems.

Key words.

AMS subject classifications.

1. Introduction.

The Algebraic Multigrid (AMG) algorithm [1, 11, 12, 2] is one of the most efficient algorithms for solving large unstructured sparse linear systems that arise in a wide range of science and engineering applications. One of AMG's most desirable properties, especially in the context of large scale problems and massively parallel computing, is its potential for algorithmic scalability: for a matrix problem with n unknowns, the number of iterative V-cycles required for convergence is ideally independent of the problem size n (resulting from error reduction per cycle with convergence factors bounded away from one that are constant in terms of the problem size n), and the work in the setup phase and in each V-cycle is, in the ideal case, linearly proportional to the problem size n. A brief overview of the basic AMG algorithm is given in Sec. 2. Familiarity with the basic AMG algorithm is assumed in the remainder of this introductory section. For real-life problems, the AMG algorithm has shown to deliver consistent near-optimal algorithmic scalability for a wide range of applications [5, 12]. Various parallel versions of AMG have been developed [4, 6, 9, 8].

Traditional coarsening schemes for AMG that are based on the coarsening heuristics originally proposed by Ruge and Stueben (RS) [11], tend to work well for problems that arise from the discretization of elliptic partial differential equations (PDEs) in two spatial dimensions (2D). For many 2D problems, a solver with optimal scalability can be obtained, with the number of iterations that is required for convergence independent of the problem size n, and memory use, setup time, and solution time per iteration linearly proportional to n. However, when traditional AMG algorithms are applied to three-dimensional (3D) problems, numerical tests show that in many

[†]Department of Applied Mathematics, University of Waterloo, Waterloo, ON N2L 3G1, Canada

[‡]Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, PO Box 808, Livermore, CA 94551, USA

[‖]Chemical and Materials Engineering Department, Box 876006, Arizona State University, Tempe, Arizona 85287-6006, USA

[§]hdesterck@uwaterloo.ca

[¶]umyang@llnl.gov

[*]heys@asu.edu

cases scalability is lost: while the number of iterations required may remain constant, computational complexities, particularly stencil size, may grow significantly leading to increased execution times and memory usage. This loss of scalability is already an issue on serial machines, but for larger problem sizes on parallel machines the effects are even more severe, with additional degradation due to increased communication. Complexity issues have been considered before. In [12] Stueben suggests the application of aggressive coarsening (which can be implemented as applying standard coarsening twice) to the finer levels, which leads to reduced complexities. Aggressive coarsening is also used in [8]. This approach also requires the use of long-range interpolation, which in combination with interpolation truncation is efficient for sequential computations. It is, however, difficult to implement in parallel, since it requires information that goes beyond immediate neighbors and might even be located on processors that are not neighbor processors according to the graph of the original matrix. In search for a simpler and possibly more efficient method, we tried to avoid the need for long-range interpolation in the present work.

We present here two new, simplified parallel coarsening algorithms. The first one, which we call the Parallel Modified Independent Set (PMIS) algorithm, is a modification of an existing parallel maximal independent set algorithm by Luby [10, 7]. This algorithm can be implemented in such a way that the resulting coarse grid is independent of the number of processors and the distribution of the points across processors. The second one, the Hybrid Modified Independent Set (HMIS) algorithm, is obtained by combining the PMIS algorithm with a one-pass RS scheme. We find that, using PMIS and HMIS coarsening, complexity growth problems largely disappear. Not surprisingly, we observe that, using regular AMG interpolation combined with PMIS and HMIS coarsening, AMG convergence factors deteriorate as a function of problem size, resulting in a loss of scalability. However, this convergence degradation can be counteracted effectively by using Krylov subspace acceleration of the AMG solvers. We show that the resulting preconditioners, with regard to total execution times and memory use, often significantly outperform preconditioners that use parallel coarsening schemes based on the classical sequential RS coarsening (e.g., the CLJP coarsening algorithm [6]). For large problems on parallel machines in the several 1,000 processor class, the newly proposed methods typically use less than half the memory, and require less than half the computer time. Also, for the problem sizes and machines tested, the scalability in memory and execution time (especially AMG setup times) is generally much better than for the existing approaches. We investigate whether these improvements hold for a large class of structured problems arising from scalar elliptic 3D PDEs, including anisotropic and convection-diffusion problems, and investigate whether these improvements also hold for 3D PDE systems on unstructured grids, in particular, First-Order System Least-Squares finite element discretizations of Stokes flow.

The paper is organized as follows. In the following section a brief overview is given of the basic AMG algorithm. In Section 3 a more detailed discussion of the arising complexity issues is presented. Section 4 describes the new PMIS and HMIS coarsening algorithms. In Section 5 the performance of the PMIS and HMIS schemes is compared with that of RS-based parallel coarsening strategies (namely CLJP and its hybrid variant, Falgout coarsening), for a large class of structured and unstructured problems arising from scalar elliptic 3D PDEs. Section 6 describes scaling studies for a 3D PDE system modeling Stokes flow. Finally, conclusions are formulated in Section 7.

2. Algebraic Multigrid. In this section we give an outline of the basic principles and techniques that comprise AMG, and we define terminology and notation. Detailed explanations may be found in [11, 12, 2]. Consider a problem of the form

$$Au = f, \tag{2.1}$$

where A is an $n \times n$ matrix with entries a_{ij}. For convenience, the indices are identified with grid points, so that u_i denotes the value of u at point i, and the grid is denoted by $\Omega = \{1, 2, \ldots, n\}$. In any multigrid method, the central idea is that "smooth error," e, that is not eliminated by relaxation must be removed by coarse-grid correction. This is done by solving the residual equation $Ae = r$ on a coarser grid, then interpolating the error back to the fine grid and using it to correct the fine-grid approximation by $u \leftarrow u + e$.

Using superscripts to indicate level number, where 1 denotes the finest level so that $A^1 = A$ and $\Omega^1 = \Omega$, the components that AMG needs are as follows:

1. "Grids" $\Omega^1 \supset \Omega^2 \supset \ldots \supset \Omega^M$.
2. Grid operators $A^1, A^2, \ldots, A^M$.
3. Grid transfer operators:
   Interpolation $P^k, k = 1, 2, \ldots M-1$,
   Restriction $R^k, k = 1, 2, \ldots M-1$.
4. Smoothers $S^k, k = 1, 2, \ldots M-1$.

These components of AMG are determined in a first step, known as the *setup phase*.

AMG Setup Phase:

1. Set $k = 1$.
2. Partition Ω^k into disjoint sets C^k and F^k.
   (a) Set $\Omega^{k+1} = C^k$.
   (b) Define interpolation P^k.
3. Define R^k (often $R^k = (P^k)^T$).
4. Set $A^{k+1} = R^k A^k P^k$ (Galerkin condition).
5. Set up S^k, if necessary.
6. If Ω^{k+1} is small enough, set $M = k+1$ and stop. Otherwise, set $k = k + 1$ and go to step 2.

Once the setup phase is completed, the *solve phase*, a recursively defined cycle, can be performed as follows:

Algorithm: $MGV(A^k, R^k, P^k, S^k, u^k, f^k)$.
If $k = M$, solve $A^M u^M = f^M$ with a direct solver.
Otherwise:
Apply smoother S^k μ_1 times to $A^k u^k = f^k$.
Perform coarse grid correction:
Set $r^k = f^k - A^k u^k$.
Set $r^{k+1} = R^k r^k$.
Apply $MGV(A^{k+1}, R^{k+1}, P^{k+1}, S^{k+1}, e^{k+1}, r^{k+1})$.
Interpolate $e^k = P^k e^{k+1}$.
Correct the solution by $u^k \leftarrow u^k + e^k$.
Apply smoother S^k μ_2 times to $A^k u^k = f^k$.

The algorithm above describes a V(μ_1, μ_2)-cycle; other more complex cycles such as W-cycles are described in [2, 13]. In every V-cycle, the error is reduced by a certain factor, which is called the convergence factor. A sequence of V-cycles is executed until the error is reduced below a specified tolerance. For a scalable AMG method,

the convergence factor is bounded away from one and independent of the problem size n, and the computational work in both the setup and solve phases is linearly proportional to the problem size n.

3. Complexity Issues. While scaling is often excellent for matrices arising from 2D PDE problems, numerical tests (see Sec. 5) show that for many 3D problems computational complexities may grow significantly for increasing problem size. this growth can lead a large memory requirement and a significantly increased amount of computational work, which may result in a severe loss of scalability that becomes prohibitive when large problem sizes are attempted. The loss of scalability manifests itself in non-scalable execution times for both the setup phase and the solve phase of the AMG algorithm.

There are two types of complexities that need to be considered: the *operator complexity* and the *stencil size*. The operator complexity, C_{op}, is defined as the sum of the number of nonzero matrix elements in the operator matrices on all grid levels, divided by the number of nonzero matrix elements of the fine grid operator matrix, A. It is a measure for the memory use and execution time in the solve phase. The stencil size on a given level is defined as the average number of coefficients per matrix row, and strongly influences the setup time, as growing stencil sizes substantially increase the number of operations required in the coarsening and interpolation process due to the need of accessing neighbors of neighbors. So, large stencil sizes can lead to large setup times, even when the operator complexity is small. Both C_{op} and the stencil sizes on all levels should ideally be independent of the problem size n. In large 3D tests using existing AMG coarsenings of RS-type [11, 6], however, it can be seen that both the operator complexity, C_{op}, and the stencil size may grow strongly, resulting in severe loss of scalability, see also Section 5. For instance, the RS coarsening [11], and its parallel variants, the CLJP and Falgout coarsenings [4, 6], generally give good results in terms of convergence factors for 3D problems, but may result in large operator complexities and severe stencil size growth. It is important to note that complexity growth can be kept under control for some 3D problems. For elasticity problems, for instance, low complexities can be obtained by judiciously choosing the AMG strength parameter, which defines which matrix elements are considered in the coarse point selection process, while retaining good convergence. For some 3D problems, interpolation truncation can be quite effective to decrease operator complexitites and stencil sizes.

We consider the classical RS-based coarsening strategies in more detail, in order to gain some more insight into why these complexity growth problems arise. The AMG coarsening process partitions the grid points on a given grid level into disjoint sets of coarse points (C-points), which are taken to the next level, and fine points (F-points), which are interpolated from C-points. Only matrix coefficients that are sufficiently large are considered in the coarsening process: only *strong connections* are considered. We say that a point i *strongly depends on* j or j *strongly influences* i if

$$|a_{ij}| \geq \alpha \max_{k \neq i} |a_{ik}|, \tag{3.1}$$

where the *strength threshold* α is a positive constant smaller than one.

In the RS, Falgout and CLJP coarsenings [11, 4, 6], the following two heuristics are imposed:

H1: For each point j that strongly influences an F-point i, j is either

a C-point, or strongly depends on a C-point k that also strongly influences i.

H2: The set of C-points needs to form a maximal independent set in the reduced graph of the matrix (with only strong connections retained in the graph).

H1 implies that two strongly connected F-points are interpolated by a common C-point. In RS, Falgout and CLJP coarsening, H1 is strongly enforced, while H2 is imposed where possible. Heuristic H1 assures that all strong connections can be taken into account adequately when interpolation formulas are constructed, generally resulting in a solver with constant AMG convergence factors bounded away from one. However, H1 is imposed by choosing additional C points, and may thus potentially lead to higher complexities. Heuristic H2 intends to provide enough C-points (maximal set), but not too many (independent set) for efficient and accurate interpolation.

The RS coarsening [11] is an essentially serial realization of the above described coarsening principles. Each point i is assigned a measure λ_i, which equals the number of points that are strongly influenced by i. Then a point with a maximal λ_i (there are usual several) is selected to be the first C-point. Now all points that strongly depend on i become F-points. For each point j that strongly influences one of these new F-points, λ_j is increased to improve j's chances to become a new C-point. This highly sequential process is repeated until all points are either F- or C-points. Heuristic H1 is enforced in a second pass over the grid by adding C-points. The resulting AMG algorithm works optimally for many highly structured problems, because the structure is preserved on coarse grids due to the systematic marching, and tends to perform nearly optimally for more unstructured problems: scalable complexities and convergence factors are obtained for a wide range of problems [5, 1, 11, 12].

The same heuristics are used as a guideline for the essentially parallel CLJP coarsening [4, 6]. In CLJP coarsening, the sequential marching strategy from RS for choosing candidate C points, is avoided by adding a random number between 0 and 1 to each measure, thus making each point distinct. Now the points with local maximal measure can be selected to be C-points leading to a completely parallel process. This algorithm retains much of the algorithmic scalability, while adding good parallel scalability for a large class of problems [6]. Due to the random nature of the coarsening, fine-grid structure is not preserved on coarse grids, and as a result, for highly structured problems, complexity and convergence may degrade compared to RS coarsening. However, the CLJP coarsening has the important advantage of being parallel, and tests show that it may actually perform somewhat better for unstructured problems than parallel RS implementations, which use the RS coarsening in the interior of each processor, but have different ways of handling boundary points. Examples of parallel RS coarsenings are the RS3 coarsening, which deals with processor boundaries via a third pass applied to the processor boundaries, or the Falgout coarsening, which is a hybrid form of the RS and the CLJP coarsening [6]. It turns out that the Falgout coarsening is often the most efficient in a parallel environment, for problems with a regular fine-grid structure. In general, RS, CLJP, and Falgout coarsening all provide good results in terms of convergence factors, but severe complexity growth may arise, especially for 3D problems. As will be shown in numerical experiments below, the growth in C_{op} and stencil size hampers scalability for large problems on machines with large processor numbers. This loss of scalability may already become an issue on serial machines, but for larger problem sizes on parallel machines the effects are

more severe, with additional degradation due to increased communication.

The discussion above has illustrated that, in general, a larger set of C-points tends to lead to more scalable convergence factors because more accurate interpolation formulas can be constructed, but, at the same time, adding C-points can significantly increase the complexity of the method.

In 2D, imposing heuristic H1 is important for obtaining an efficient method with scalable convergence factors and good complexity. However, it appears that in 3D, the balance may be different: imposing heuristic H1 may be too strong a requirement, resulting in prohibitive complexity growth. In this paper, we look for an improved balance between AMG complexity and convergence factors for 3D problems, by considering more sparse coarse grids while trying to retain acceptable convergence factors. We present new, simplified coarsening algorithms that are based on enforcing heuristic H2, while mostly ignoring heuristic H1. In fact, we replace heuristic H1 by the following much less stringent requirement [9] for our new coarsening algorithms:

H1$'$: Each F-point needs to strongly depend on at least one C-point.

4. Two New Parallel AMG Variants. In this section we give a detailed description of the newly proposed PMIS and hybrid HMIS coarsening algorithms, followed by a brief description of the interpolation formula used in our parallel AMG preconditioners.

4.1. The PMIS Coarsening Algorithm. Given a matrix problem $Au = f$, define V as the set of unknowns (or nodes) of A. Define the auxiliary strength matrix S as follows:

$$S_{ij} = \left\{ \begin{array}{ll} 1 & \text{if } i \neq j \text{ and } |a_{i,j}| \geq \alpha \max\limits_{k \neq i} |a_{ik}|, \\ 0 & \text{otherwise.} \end{array} \right\}, \tag{4.1}$$

i.e., $S_{ij} = 1$ only if i strongly depends on j. Here α is the strength threshold which is usually chosen between 0.25 and 0.50 [11]. The ith row of S defines S_i, the set of nodes that influence i, while the ith column of S defines S_i^T, the set of nodes that are influenced by i. That is, $S_i = \{j \in V \mid S_{ij} = 1\}$ and $S_i^T = \{j \in V \mid S_{ji} = 1\}$. Define $G = (V, E) = g(S)$ as the undirected graph of S, where V is now the vertex set of the graph, and E the edge set: $E = \{\{i, j\} \in V \times V \mid S_{ij} = 1 \text{ or } S_{ji} = 1\}$. Given an undirected graph $G = (V, E)$, $I \subset V$ is an *independent set* of V in G *iff* $\forall i, j \in I : \{i, j\} \notin E$. $I \subset V$ is a *maximal independent set* of V in G *iff* I is an independent set of V in G, and $\forall j \in V \setminus I, I \cup \{j\}$ is not an independent set of V in G. Given an undirected graph $G = (V, E) = g(S)$ of a strength matrix S, $G' = (V', E') = g(V', S)$ is the subgraph of G induced by vertex set $V' \subset V$, *iff* $E' = \{\{i, j\} \in V' \times V' \mid \{i, j\} \in E\}$.

A formal description of the PMIS algorithm can be given as follows.

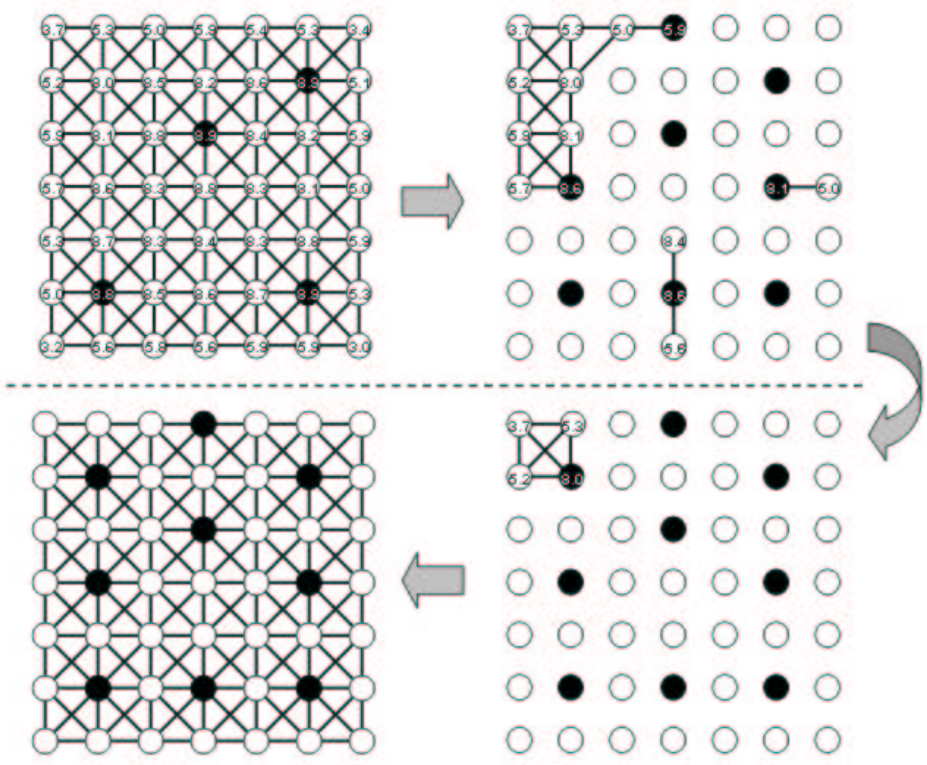

FIG. 4.1. *PMIS coarsening performed for a 2D 9-point Laplace operator on a 7×7-grid.*

The PMIS algorithm:

Given $G = (V, E) = g(S)$, define weights $w(i)$ $\forall i \in V$: $w(i) = \#S_i^T + \text{Rand}([0, 1])$.
The initial set of F-points $F = \{i \in V | \#S_i^T = 0\}$.
The initial set of C-points $C = \emptyset$.
Take the F-points out of the remaining vertex set: $V' = V \setminus F$.
The subgraph induced by the remaining vertex set is then $G' = g(V', S)$.
While $V' \neq \emptyset$ do:
  Choose an independent set I of V' in G':
    $i \in I$ *iff* $w(i) > w(j)$ $\forall\ j$ with $\{i, j\} \in E'$.
  Make all elements of I C-points: $C = C \cup I$.
  Make all elements of $V' \setminus I$ that are strongly influenced by a new C-point,
    F-points: $F = F \cup F_{new}$, with $F_{new} = \{j \in V' \setminus I \mid \exists\ i \in I : i \in S_j\}$.
  Remove all new C- and F-points from V': $V' = V' \setminus \{I \cup F_{new}\}$.
  The remaining subgraph is then $G' = g(V', S)$.
Enddo.

The PMIS algorithm as described above is basically Luby's parallel maximal independent set algorithm [10], with three small modifications. An illustration of the algorithm applied to a 2D 9-point Laplace operator on a 7×7 grid is given in Figure 4.1. The different steps in the algorithm can briefly be described as follows. First,

every point i in V is given a measure, $w(i)$, which is the sum of the number of points that i influences, $\#S_i^T$ (8 for all interior points in the example, 5 on the boundaries and 3 on the corners), and a random number between 0 and 1. The measure $w(i)$ is a measure of point i's eligibility to be a C-point. Points that influence many other points are likely to be good candidates for becoming C-points, since many strongly connected points would be able to interpolate from them, and they thus receive a large measure $w(i)$. The random number is added to break ties when the numbers of influenced points of neighboring points are equal. In Luby's original algorithm, only random numbers are used to select the independent set, so adding $\#S_i^T$ is the first modification to Luby's algorithm. In every step of the ensuing procedure, all points in V that have not been assigned yet as C- or F-points, become C-points (black points in Figure 4.1) if their measure is larger than those of their unassigned strongly connected neighbors. In Luby's original algorithm, all unassigned points that are strongly connected to new C-points, become F-points, and the process repeats with the selection of new C-points as above. After a finite, typically small number of iterations, the grid is fully partitioned into C- and F-points. The resulting set of C-points forms a maximal independent set in the undirected graph of the auxiliary strength matrix S. The set is independent, because for every new C-point, all its neighbors are F-points by construction, and the set is maximal, because turning any F-point into a C-point would violate the independence, as every new F-point is the neighbor of a C-point by construction.

In order to make the Luby algorithm useful for AMG coarsening, two additional small modifications have to be made. First, initially all points i of V that do not influence any other point of V, are made F-points, because they would not be useful for interpolation as C-points. This also precludes that isolated points stay C-points on all levels. Because of this modification, the set of C-points determined by the PMIS algorithm may not strictly be maximal (because isolated points can be made C-points without violating the independence of the set). Second, directionality of strong connections is taken into account (heuristic H1$'$): when a new set of C-points is determined, only unassigned neighbors that are strongly influenced by a new C-point are made F-points. This assures that all F-points (except for, possibly, the initially assigned F-points) are influenced by at least one C-point, from which they will be able to interpolate. Because of this modification, the set of C-points determined by the PMIS algorithm may not be strictly independent (because points in C may be strongly connected).

This algorithm is inherently parallel, due to the random nature of C-point selection. Parallelization is straightforward, and only requires communication of boundary information between neighboring processors. Note also that if we guarantee that each point has a fixed random number independent of the number of processors and the distribution of points across processors, the algorithm will generate the same coarsening independent of the number of processors and the distribution of points across processors. This algorithm can also be seen as a simplified version of the CLJP coarsening algorithm [4, 6]. The CLJP algorithm is also derived from Luby's parallel independent set algorithm, with additional operations that guarantee heuristic H1 of Sec. 3.

4.2. The HMIS Coarsening Algorithm. Inspired by the hybridization concept of Falgout coarsening [6], we decided to use the same idea for a new hybrid coarsening algorithm that does not enforce heuristic H1. Falgout coarsening proceeds by first using the classical RS coarsening independently on each processor. Then all

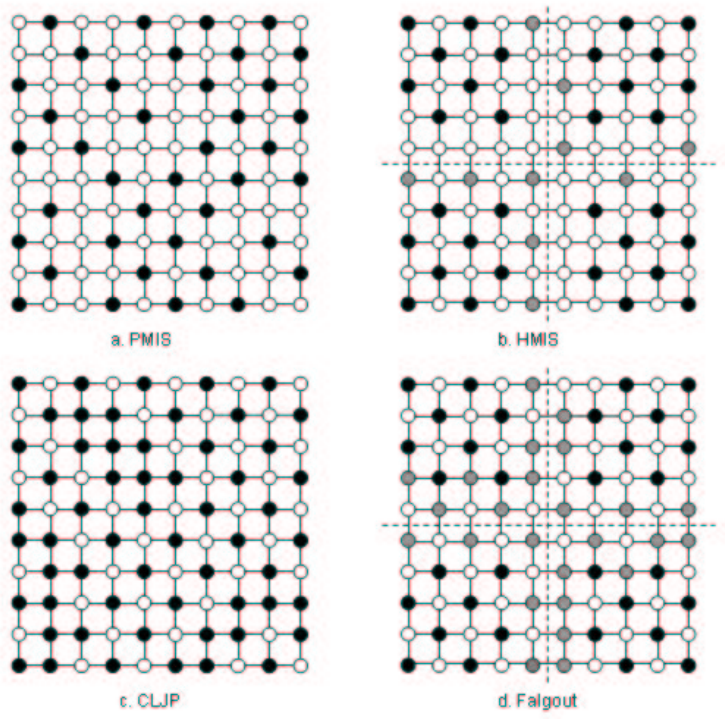

FIG. 4.2. *Various coarsenings applied to a 2D 5-point Laplace operator on a 10×10-grid across 4 processors*

the C-points that have been generated in the RS coarsening process and that are located in the interior of each processor are combined in a set, which is used as the first set of C-points (note that this might not be an independent set) for the CLJP-coarsening strategy. The treatment of the boundary points is completely handled by the CLJP coarsening scheme.

Now, since we are pursuing a coarsening strategy that does not enforce heuristic H1, we decided to use only the first pass of the RS coarsening scheme on each processor independently as our first part of the hybrid algorithm. When this step has been completed, we use all the interior C-points, i.e. those C-points that are not located on the processor boundaries, as the first independent set of C-points to be fed into the PMIS algorithm. We call the resulting new coarsening scheme the Hybrid Modified Independent Set (HMIS) coarsening scheme, since it is a hybrid algorithm, which combines one-pass RS coarsening inside the processor interiors with PMIS coarsening near the processor boundaries (each of which ignore heuristic H1, but impose H1$'$). An example of the HMIS coarsening applied to the 2D 5-point Laplacian on a 10×10-grid distributed across 4 processors is illustrated in Figure 4.2b, where the black points denote C-points that were generated in the first phase of the HMIS coarsening, when one-pass RS coarsening is applied independently on each processor, and the grey points are C-points that were added in the second phase, when applying PMIS coarsening.

The HMIS coarsening scheme will be compared in the following sections with the CLJP, Falgout, and PMIS coarsenings. Due to previous experiences with CLJP and Falgout coarsenings [6], we anticipate that for problems with regular fine-grid structure, the HMIS algorithm may perform better than the PMIS algorithm, due to the preservation of structure within the processor interiors. Figure 4.2 illustrates the four coarsenings that we will compare applied to a 2D 5-point Laplace operator distributed across 4 processors. Processor boundaries are not indicated for PMIS and CLJP, since they generate coarsenings that are independent of the number of processors. White points denote F-points, whereas black and grey points denote C-points. Grey C-points are those that have been generated in the second phase of the hybrid algorithms. This small example already shows that the new coarsenings will lead to much coarser grids, due to relaxing condition H1.

4.3. Interpolation. In classical AMG, the interpolation of the error at the F-point i takes the form

$$e_i = \sum_{j \in C_i} w_{i,j} e_j, \tag{4.2}$$

where $w_{i,j}$ is an interpolation weight determining the contribution of the value e_j in the final value e_i, and C_i is the subset of C-points whose values will be used to interpolate a value at i. In most classical approaches to AMG interpolation, C_i is a subset of the nearest neighbors of grid point i, and longer-range interpolation is normally not considered.

The points, to which i is connected, comprise three sets: the set C_i, the set D_i^s of points that strongly influence i but are not coarse interpolatory points, and the set D_i^w of points connected to, but not strongly influencing, i. Based on assumptions on small residuals for smooth error [1, 11, 12, 2], the following formula can be derived for the interpolation weights:

$$w_{i,j} = -\frac{1}{a_{i,i} + \sum_{k \in D_i^w} a_{i,k}} \left(a_{i,j} + \sum_{k \in D_i^s} \frac{a_{i,k} a_{k,j}}{\sum_{m \in C_i} a_{k,m}} \right). \tag{4.3}$$

However, because on our coarse grids heuristic H1 is not strongly imposed and two strongly connected F-points i and k may not have a common C-point m, it may happen that some of the terms $\sum_{m \in C_i} a_{k,m}$ in Eq. 4.3 vanish. We have modified interpolation formula Eq. 4.3 such that in case that $\sum_{m \in C_i} a_{k,m} = 0$, $a_{i,k}$ is added to the so-called diagonal term (the term $a_{i,i} + \sum_{k \in D_i^w} a_{i,k}$ in Eq. 4.3), i.e. a strongly influencing neighbor point k of i that is an F-point is treated like a weak connection of i if it does not share a common neighbor that is a C-point with i. We denote the set of strongly connected neighbors of i that are F-points and do not share a common C-point, by F_i. Although this does not guarantee that the interpolation formula will be sufficiently accurate (see the numerical convergence results below and the ensuing discussion for an assessment of this question), it does guarantee that the interpolation formula remains well-defined, and that constant error is interpolated accurately. Our interpolation is further modified as proposed in [6] to avoid extremely large interpolation weights that can lead to divergence. This leads to the following

interpolation formula:

$$w_{i,j} = -\frac{1}{a_{i,i} + \sum\limits_{k \in D_i^w \cup F_i} a_{i,k}} \left(a_{i,j} + \sum_{k \in D_i^s \setminus F_i} \frac{a_{i,k}\hat{a}_{k,j}}{\sum\limits_{m \in C_i} \hat{a}_{k,m}} \right), \tag{4.4}$$

where

$$\hat{a}_{i,j} = \begin{cases} 0 & \text{if } \operatorname{sign}(a_{i,j}) = \operatorname{sign}(a_{i,i}) \\ a_{i,j} & \text{otherwise.} \end{cases}$$

5. Numerical Results: Scalar Elliptic PDEs. This section contains an extensive numerical scalability study of the PMIS and HMIS coarsenings as compared with the existing CLJP and Falgout coarsenings. These results were obtained on the LLNL Linux cluster MCR. Strength threshold $\alpha = 0.25$ was used for all PMIS and HMIS runs. For CLJP and Falgout, the strength thresholds used were the optimal values determined for the various test cases in earlier work [6], which were in general $\alpha = 0.5$ for 3D test runs and $\alpha = 0.25$ for 2D test runs, unless noted otherwise. A large class of scalar elliptic PDE problems on structured and unstructured grids is considered. For all tests, the AMG solver is accelerated with a GMRES(10) Krylov solver. Most test cases were discussed in detail in [6] and will only be described briefly here.

The legend for the tables in all subsections is as follows:

- p: number of processors
- C_{op}: operator complexity
- $\#lev$: total number of levels in the AMG V-cycle
- s_{avg}: maximal average stencil size (average number of nonzeros per matrix row, largest value that occurs on any of the V-cyle levels)
- l_{max}: on which V-cycle level this maximum s_{avg} occurs
- t_{setup}: AMG setup time
- t_{solve}: AMG solve time
- its: number of iterations to convergence
- t_{tot}: total time (AMG setup + solve time)
- dof: total number of degrees of freedom

5.1. 3D Laplacian Problems. In Table 5.1, we show the results of parallel scaling tests for a 7-point standard finite difference discretization of Laplace's equation

$$-\Delta u = f \tag{5.1}$$

on a cubic regular domain, with problem size per processor $40 \times 40 \times 40$ points. These results immediately illustrate the complexity growth problems that affect the CLJP, and, to a somewhat lesser extent, the Falgout coarsening. Operator complexities, total numbers of levels, and stencil sizes are excessively high, and setup times are much larger than the solution phase execution times. Interestingly enough, while CLJP has extremely large operator complexities, its average stencil sizes are significantly smaller than those of the Falgout coarsening. A closer examination of the two coarsenings reveals that Falgout coarsening initially coarsens much faster than CLJP. Stencil sizes grow faster than for CLJP (caused by somewhat larger interpolation operator stencil sizes), peak in the intermediate levels, at greatly reduced grid sizes, and decrease fairly fast in the lowest levels. CLJP coarsens fairly slow at the beginning. Its stencil sizes

Method	p	C_{op}	$\#lev$	s_{avg}	l_{max}	t_{setup}	t_{solve}	its	t_{tot}
CLJP	1	14.39	15	157.9	6	1.88	1.47	6	3.35
	8	15.75	17	206.3	9	6.03	4.23	8	10.26
	64	16.59	20	260.1	11	13.30	7.69	9	20.99
	512	17.02	22	307.0	12	22.33	13.50	10	35.83
	1000	17.15	23	316.3	12	27.16	15.03	10	42.19
	1331	17.19	23	321.4	12	29.57	16.68	10	46.25
Falgout	1	3.61	11	103.6	5	0.77	0.42	5	1.19
	8	4.43	13	282.4	8	3.24	1.33	6	5.57
	64	5.07	15	451.5	9	11.77	3.15	6	14.93
	512	5.45	18	575.4	9	23.19	7.26	7	30.45
	1000	5.51	19	598.4	9	26.64	8.59	7	35.23
	1331	5.55	19	608.3	9	30.57	8.99	7	39.56
PMIS	1	2.32	7	55.4	4	0.41	0.87	13	1.28
	8	2.35	8	63.2	4	1.15	2.56	17	3.71
	64	2.36	9	68.1	4	2.81	4.86	21	7.67
	512	2.37	10	70.7	4	5.04	7.73	25	12.77
	1000	2.37	10	71.1	4	6.63	8.39	26	15.02
	1331	2.37	10	71.3	4	8.28	9.71	28	17.99
HMIS	1	2.79	7	57.1	5	0.73	0.38	5	1.11
	8	2.80	8	76.3	5	1.45	1.74	10	3.19
	64	2.80	9	79.3	5	2.62	3.23	12	5.85
	512	2.80	10	82.2	5	4.89	5.03	15	9.92
	1000	2.80	10	83.6	5	6.18	5.10	15	11.28
	1331	2.80	10	83.4	5	7.56	5.62	16	13.18

TABLE 5.1

AMG-GMRES(10) with different coarsenings applied to the 7-point Laplacian, 40^3 *dof per processor.*

decrease gradually, stay close to the maximal size over various levels and decrease slower than for Falgout in the lowest levels.

PMIS and HMIS, however, are two to three times as fast as CLJP and Falgout, use less than half the amount of memory, and have much smaller stencil sizes, leading to significantly smaller setup times. They require more iterations, but the execution time per iteration is much lower due to a smaller operator complexity, resulting in overall time savings also for the solution phase. HMIS is slightly faster and uses only a little more memory than PMIS for this highly structured problem.

In Table 5.2 we show scaling results for the 7-point Laplacian problem with a large problem size per processor. CLJP results were not obtained because the MCR machine quickly ran out of memory for this coarsening scheme. This is a structured problem, and therefore the Falgout method, which is capable of keeping the structure on coarser grids away from subdomain problems, does not suffer as much from complexity problems at the total scale of the problem considered. The PMIS and HMIS methods do not have much room to improve the results, and as a result the PMIS/HMIS timings are very similar to those of Falgout for this problem with a large ratio of processor interior dof to processor boundary dof. However, the memory used by PMIS and HMIS is still only about half of the memory used with Falgout coarsening. Note that the 1331-processor runs feature 1.331 billion variables, which

Method	p	C_{op}	$\#lev$	s_{avg}	l_{max}	t_{setup}	t_{solve}	its	t_{tot}
Falgout	8	4.23	17	501.6	10	49.56	19.15	6	68.71
	512	4.69	21	750	10	133.03	33.73	8	166.76
	1331	4.74	23	773.9	10	139.48	32.64	7	172.12
PMIS	8	2.59	9	73.2	5	21.33	55.20	22	76.53
	512	2.61	12	79.4	5	36.47	123.00	44	159.47
	1331	2.61	12	79.7	5	41.26	131.21	46	172.47
HMIS	8	2.89	9	141	5	44.48	32.17	11	76.65
	512	2.88	11	131.3	5	103.37	58.01	19	161.38
	1331	2.88	12	130.4	5	114.58	50.26	16	164.84

TABLE 5.2

AMG-GMRES(10) with various coarsenings applied to the 7-point Laplacian, 100^3 *dof per processor.*

pushes the scalability of our AMG solver over the one-billion dof border.

Even though PMIS and HMIS may not always be faster than Falgout coarsening for large problem sizes per processor, the results presented earlier show that they can be significantly faster and more scalable for smaller problem sizes per processor, which is of benefit on architectures with smaller memory capacities, or when the rest of the application requires a large amount of the memory, which is often the case.

We also investigate AMG scalability for a 3D unstructured grid problem, since previous experiments with CLJP and Falgout have shown that CLJP performs better on unstructured grids [6]. A finite element discretization of the Laplace equation is applied to a grid with approximately 20,000 dof per processor. Table 5.3 shows that PMIS and HMIS perform better than CLJP and Falgout. PMIS/HMIS are more than double as fast as Falgout, and use less than half the memory of Falgout. We find that CLJP is slightly faster than Falgout, and PMIS is slightly faster than HMIS. This is an unstructured problem, for which the RS marching approach is not especially advantageous because there is no fine-grid structure to be kept. Interestingly, the more random nature of coarse point selection of CLJP and PMIS seems to be slightly better for unstructured problems.

Method	p	C_{op}	s_{avg}	t_{setup}	t_{solve}	its	t_{tot}
CLJP	4	3.84	151.6	1.32	0.89	9	2.21
	32	4.47	207.0	5.07	2.76	10	7.83
	288	4.68	271.0	12.34	9.35	12	21.69
Falgout	4	4.45	165.6	1.64	1.01	9	2.65
	32	5.30	232.4	6.32	2.97	10	9.29
	288	5.53	288.4	14.01	9.55	12	23.56
PMIS	4	1.41	39.9	0.39	0.73	15	1.12
	32	1.45	49.3	0.92	2.11	18	3.03
	288	1.46	53.0	2.41	4.86	24	7.27
HMIS	4	1.48	45.4	0.47	0.75	15	1.22
	32	1.59	56.7	1.31	2.09	18	3.40
	288	1.60	60.8	2.91	4.94	23	7.85

TABLE 5.3

AMG-GMRES(10) with various coarsenings applied to a 3D Laplacian on an unstructured grid.

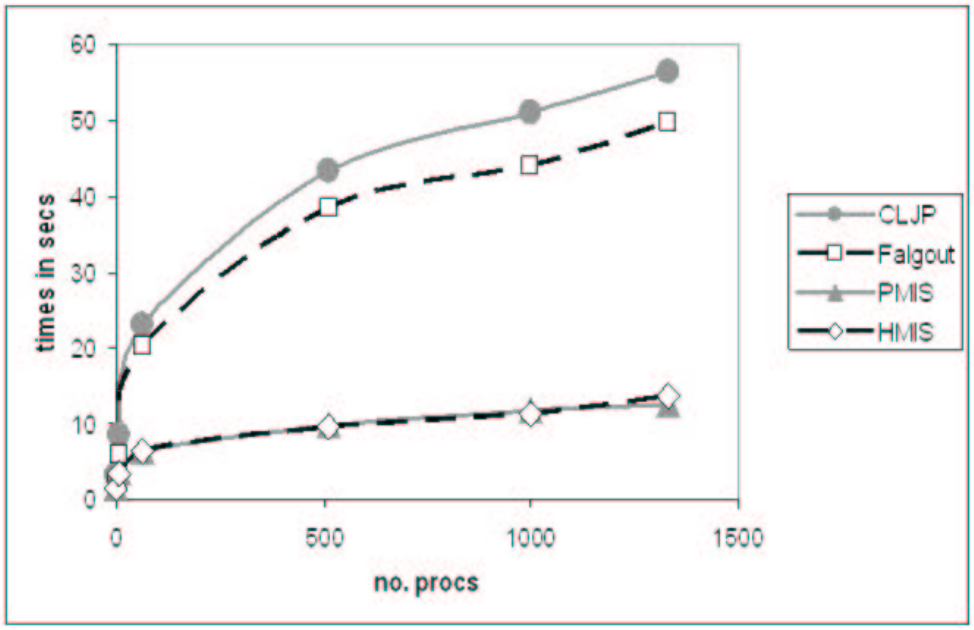

FIG. 5.1. *Total times for AMG-GMRES(10) with various coarsenings applied to a 27-point operator,* 40^3 *degrees of freedom per processor.*

In Figure 5.1 we present the results of parallel scaling tests for CLJP, Falgout, PMIS, and HMIS coarsening applied to a 27-point finite element discretization of Laplace's equation on a cubic regular domain. Problem size per processor is 40^3 degrees of freedom. For this problem CLJP has an operator complexity that is growing from 2.08 on one processor to 2.99 on 1331 processors, Falgout's complexities range from 1.62 to 2.48, whereas PMIS and HMIS have constant complexities of 1.11 and 1.20, respectively. Stencil sizes for CLJP range from 238.4 to 1015, and for Falgout from 123.8 to 849.0, whereas they only slightly vary for PMIS from 36.4 to 51.4 and for HMIS from 85.4 to 106.3. The stencil size, which is about 27 for this problem on the finest grid level, thus grows to an astounding order of a thousand for CLJP and Falgout coarsening. In spite of the fact that Falgout and CLJP require only about half of the number of iterations as PMIS and HMIS, overall timings for HMIS and PMIS are up to four times as fast. Just as in the 7-point case, PMIS and HMIS are more than double as fast and use less than half the memory of CLJP and Falgout.

In Table 5.4 scaling results for the 27-point operator are presented using $90 \times 90 \times 90$ points per processor. Again, CLJP results were not obtained because the MCR machine quickly ran out of memory for this problem. Here, the Falgout coarsening also runs out of memory due to growing complexities when using 64 or more processors. Obviously, PMIS and HMIS are here preferable over the other coarsenings.

Method	p	C_{op}	$\#lev$	s_{avg}	l_{max}	t_{setup}	t_{solve}	its	t_{tot}
Falgout	8	2.29	17	769.8	10	35.68	25.54	8	61.22
	512	Out of Memory							
	1000	Out of Memory							
PMIS	8	1.11	8	42.9	4	12.29	32.43	17	44.72
	512	1.11	10	51.7	5	17.36	55.35	25	72.71
	1000	1.11	11	53.1	6	20.02	59.41	27	79.43
HMIS	8	1.23	9	138.9	5	17.96	24.04	11	42.00
	512	1.23	10	151.3	5	24.24	47.17	20	71.41
	1000	1.23	11	152.6	5	26.58	53.82	22	80.40

TABLE 5.4
AMG-GMRES(10) with various coarsenings applied to a 27-point operator, 90^3 dof per processor.

5.2. Anisotropic Laplacian and Convection-Diffusion Problems. In order to test the robustness and generality of the trends observed in the previous subsection, we have also tested performance and scalability for an anisotropic problem

$$-cu_{xx} - u_{yy} - u_{zz} = f \tag{5.2}$$

with c =0.001 (see Table 5.5), and a convection-diffusion problem

$$-\Delta u + c(u_x + u_y + u_z) = f, \tag{5.3}$$

with $c = 10q$ and the number of processors $p = q^3$, which leads to a nonsymmetric matrix (see Table 5.6). For both problems 40^3 points per processor were used. For the anisotropic problem we chose as a strength threshold $\alpha = 0.25$ for the CLJP and Falgout coarsenings, and $\alpha = 0.75$ for the nonsymmetric problem.

Method	C_{op}	$\#lev$	s_{avg}	l_{max}	t_{setup}	t_{solve}	its	t_{tot}
CLJP	7.07	20	241.3	12	24.32	9.59	8	33.91
Falgout	3.95	19	297.2	10	31.41	7.23	6	38.64
PMIS	2.29	13	31.8	6	7.70	11.59	29	19.29
HMIS	3.60	13	35.3	7	8.90	6.99	16	15.89

TABLE 5.5
Anisotropic Laplacian, 40^3 dof per processor, 1331 processors.

For both test cases PMIS and HMIS are faster, require less memory, and are more scalable than CLJP and Falgout very much in the same way as above. Interestingly enough, in Table 5.5, we find that the operator complexity of Falgout is only slightly larger than that of HMIS, but the setup time for HMIS is much lower due to a significantly smaller stencil size.

5.3. 2D Laplacian Problems. It is interesting to see how PMIS and HMIS perform for 2D problems. As mentioned before, in 2D CLJP and Falgout coarsenings typically do not show complexity growth, so we do not expect that PMIS and HMIS can offer much improvement. Table 5.7 shows that for a 9-point 2D Laplacian finite element discretization with 250^2 dof per processor, CLJP and Falgout coarsening indeed provide scalable AMG methods, and, not contrary to expectation, PMIS and HMIS are somewhat slower while not saving much memory. Note the very small

Method	C_{op}	$\#lev$	s_{avg}	l_{max}	t_{setup}	t_{solve}	its	t_{tot}
CLJP	9.62	25	91.2	10	20.57	16.14	16	36.71
Falgout	5.32	22	107.7	8	17.95	9.35	12	27.30
PMIS	2.34	11	68.2	5	8.43	10.59	29	19.02
HMIS	2.89	12	77.0	5	8.75	7.16	17	15.91

TABLE 5.6
Convection-diffusion problem, 40^3 dof per processor, 1331 processors.

Method	p	C_{op}	$\#lev$	s_{avg}	l_{max}	t_{setup}	t_{solve}	its	t_{tot}
CLJP	1	1.76	10	28.6	6	0.46	0.54	8	1.00
	64	1.83	14	35.2	7	1.07	1.93	10	3.00
	256	1.83	15	35.6	7	2.02	3.50	10	5.52
	1024	1.83	16	36.2	8	6.00	2.76	10	8.76
Falgout	1	1.33	7	9.0	1	0.26	0.35	7	0.61
	64	1.35	12	25.5	9	0.75	1.17	7	1.92
	256	1.35	14	28.9	9	1.56	1.62	8	3.18
	1024	1.35	15	30.9	10	5.74	2.01	8	7.75
PMIS	1	1.24	7	9.3	4	0.21	1.11	21	1.32
	64	1.24	10	10.5	5	0.46	4.30	37	4.76
	256	1.24	11	10.8	6	1.03	4.97	41	6.00
	1024	1.24	12	10.9	6	4.27	6.06	48	10.33
HMIS	1	1.33	7	9.0	1	0.24	0.35	7	0.59
	64	1.33	10	11.0	7	0.53	1.91	16	2.44
	256	1.33	11	11.4	7	1.08	2.09	17	3.17
	1024	1.33	12	11.6	7	4.34	2.60	20	6.94

TABLE 5.7
AMG-GMRES(10) with various coarsenings applied to a 9-point 2D Laplacian, 250^2 dof per processor.

maximal stencil sizes for PMIS and HMIS coarsening. Scaling tests for the 2D 5-point Laplacian show very similar results, and are not shown here.

We also considered the 2-dimensional rotated anisotropic problem

$$-(c^2 + \epsilon s^2)u_{xx} + 2(1-\epsilon)scu_{xy} - (s^2 + \epsilon c^2)u_{yy} = 1 \tag{5.4}$$

with $s = \sin\gamma$, $c = \cos\gamma$, and ϵ =0.001. We again used 250^2 dof per processor, and ran tests for the rotation angles $\gamma = 45^o$ and $\gamma = 60^o$. Results using 1024 processors are shown in Tables 5.8 and 5.9. For these test problems, the PMIS and HMIS coarsenings perform much worse than CLJP and Falgout. Particularly PMIS convergence degrades. Also, note the small stencil sizes when using PMIS and HMIS coarsening. These results show that the interpolation we use is not good enough for these problems.

5.4. 3-Dimensional PDEs with Jumps. A similar observation as in the case of rotated anisotropies can be made for the difficult case of a 3D elliptic PDE with jumps in the coefficients. We solve the partial differential equation

$$(au_x)_x + (au_y)_y + (au_z)_z = 1 \tag{5.5}$$

with Dirichlet boundary conditions on a unit cube, where $a(x,y,z) = 1000$ on $0.1 < x,y,z < 0.9$, $a(x,y,z) = 0.01$ on $0 < x,y,z < 0.1$ and the other cubes of size

Method	C_{op}	$\#lev$	s_{avg}	l_{max}	t_{setup}	t_{solve}	its	t_{tot}
CLJP	2.96	21	40.8	11	8.02	3.29	9	11.31
Falgout	2.45	21	46.5	10	9.12	3.80	9	12.92
PMIS	1.91	15	11.1	7	5.26	20.47	96	25.73
HMIS	2.26	15	15.7	7	5.40	11.89	55	17.29

TABLE 5.8
2D rotated anisotropic Laplacian, $\gamma = 45^o$, 250^2 dof per processor, 1024 processors.

Method	C_{op}	$\#lev$	s_{avg}	l_{max}	t_{setup}	t_{solve}	its	t_{tot}
CLJP	4.93	20	44.9	9	8.43	11.28	29	19.71
Falgout	3.39	20	38.7	10	7.55	6.09	17	13.64
PMIS	1.75	12	12.8	3	4.21	77.62	550	81.83
HMIS	3.27	14	27.0	3	5.22	10.63	52	15.85

TABLE 5.9
2D rotated anisotropic Laplacian, $\gamma = 60^o$, 250^2 dof per processor, 1024 processors.

$0.01 \times 0.01 \times 0.01$ that are located at the corners of the domain, and $a(x, y, z) = 1$ elsewhere. The results in Table 5.10 show that for problems with large jumps in coefficients scalability is lost.

Method	C_{op}	$\#lev$	s_{avg}	l_{max}	t_{setup}	t_{solve}	its	t_{tot}
CLJP	17.00	23	306.3	12	27.47	25.01	17	52.48
Falgout	5.77	19	591.4	9	26.28	14.41	13	40.69
PMIS	2.40	10	69.2	5	6.79	205.00	686	211.79
HMIS	2.82	10	87.3	5	6.27	65.99	202	72.26

TABLE 5.10
3D elliptic PDE with varying coefficients, 40^3 dof per processor, 1000 processors.

The results in Tables 5.8, 5.9 and 5.10 suggest that, for these difficult problems, the stencil sizes resulting from PMIS and HMIS coarsening may be too small for nearest-neighbor interpolation to be sufficiently accurate, and Krylov acceleration is not effective in this case. Additional C-points or long-range interpolation may be required here. The number of iterations needed for convergence can be reduced by increasing the number of relaxation sweeps on each level of a V-cycle, but, for the 3D example, this does not improve total execution times. Good convergence can be obtained by using W-cycles (PMIS converges within 40 iterations and HMIS within 16 iterations for the example in Table 5.10 using this approach), but W-cycles are very expensive and not scalable. Adding C-points in order to better adhere to heuristic H1 would improve interpolation accuracy and convergence, but the resulting complexity growth may be prohibitive for many large applications. Another possible approach would be to consider, for this type of difficult problems, interpolation formulas with an extended range not limited to the nearest neighbors, for instance techniques along the lines of the multi-pass interpolation proposed by Stueben [12]. This has to be done carefully, because increasing the range of interpolation may result in stencil growth due to the RAP Galerkin condition. Preliminary results for a sequential run are given in Table 5.11. Here the 3D problem is solved on a $60 \times 60 \times 60$ cube and CLJP, RS, PMIS and PMIS/MP (PMIS with multi-pass interpolation) coarsenings are compared. The results show that PMIS with multi-pass interpolation converges

much faster, and interestingly enough, for this example the complexities of PMIS and PMIS/MP are comparable. Note that, since this is a sequential run, we cannot make any comparison with Falgout or HMIS coarsening, and that we can choose here the "classical" RS-coarsening, which is inherently sequential and in general leads to better complexities than Falgout, since there is no processor boundary treatment. Improved interpolation formulas and their use on grids coarsened with the PMIS and HMIS techniques, will be considered in more detail in future research.

Method	C_{op}	$\#lev$	s_{avg}	l_{max}	t_{setup}	t_{solve}	its	t_{tot}
CLJP	15.20	15	170.2	9	9.02	10.73	11	19.75
RS	5.00	13	185.3	8	3.94	2.77	7	6.71
PMIS	2.55	7	52.9	4	2.26	18.54	70	20.80
PMIS/MP	2.57	7	55.0	4	1.84	6.87	25	8.71

TABLE 5.11
3D elliptic PDE with varying coefficients, 60^3 *on one processor*

6. Numerical Results: 3D FOSLS Stokes System. In this section, we present numerical results for 3D Stokes fluid flow simulations using a First-Order System Least-Squares (FOSLS) finite element discretization. These results were obtained on the linux cluster Hemisphere at CU Boulder, using Jeff Heys' parallel FOSLS code *Parafos*, coupled to *hypre*'s AMG-code *BoomerAMG*.

The Stokes equations for modeling low Reynolds number ($Re << 1$) flow can be written as:

$$-\nabla q + \Delta \mathbf{v} = 0 \quad \text{in } \Omega, \tag{6.1}$$

$$\nabla \cdot \mathbf{v} = 0 \quad \text{in } \Omega, \tag{6.2}$$

where q is pressure and $\mathbf{v} = (v_1, v_2, v_3)$ is velocity. The standard Galerkin finite element method for discretizing equations 6.1 and 6.2 does not produce an H^1-elliptic form and multigrid schemes typically perform poorly. An alternative formulation is based on reformulating the Stokes equation as a first-order system and minimizing the least-squares norm of residual equations in the resulting system – the so-called, First-Order System Least-Squares (FOSLS) approach [3]. Defining a 3 by 3 matrix of new variables, U, the first-order system for the Stokes equation is

$$U - \nabla \mathbf{v}^T = 0 \text{ in } \Omega, \tag{6.3}$$

$$\nabla \cdot U - (\nabla q)^T = 0 \text{ in } \Omega, \tag{6.4}$$

$$\nabla \cdot \mathbf{v} = 0 \text{ in } \Omega. \tag{6.5}$$

To achieve a fully H^1 elliptic functional and to help expose divergence free error [3], the previous first-order system is augmented with the following consistent equations:

$$\nabla \times U = 0 \text{ in } \Omega, \tag{6.6}$$

$$\nabla tr(U) = 0 \text{ in } \Omega, \tag{6.7}$$

where $tr(U) = U_{11} + U_{22} + U_{33}$. For the first-order system, bold letters indicate a vector, capital letters indicate a second-order tensor, and the shape of zero is implied by the left side.

The least-squares functional resulting from the system of first-order Eqs. (6.3-6.7) is given by

$$G(U,\mathbf{v},q) := \begin{array}{l} \|U-\nabla\mathbf{v}^T\|_{0,\Omega}^2+\|\nabla\cdot U-(\nabla q)^T\|_{0,\Omega}^2+ \\ \|\nabla\cdot\mathbf{v}\|_{0,\Omega}^2+\|\nabla\times U\|_{0,\Omega}^2+\|\nabla tr(U)\|_{0,\Omega}^2 \end{array} . \tag{6.8}$$

The boundary conditions have been omitted from G because they can be imposed directly on the finite element (approximation) space.

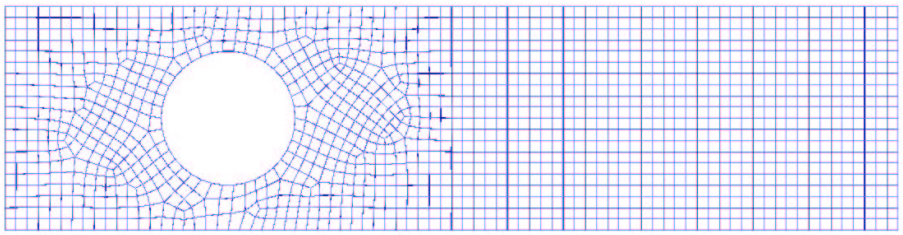

FIG. 6.1. *Unstructured grid section for the 3D Stokes flow test problem.*

The functional is minimized by setting the Gateaux derivative to zero in the weak sense. A finite element basis is then chosen so that the weak form generates a matrix problem. All of the simulations presented in this Section utilized a trilinear finite element basis for all of the variables. The FOSLS formulation allows the solution spaces for the variables to be chosen independently, and there is no restrictive stability condition to satisfy. As a result, both the pressure and velocity in the Stokes equations can be approximated with a trilinear basis. The functional, G, measures the first derivative of the error in the primary variables (i.e., velocity and pressure), unlike the error in the L_2 sense. Therefore, error characterized by 'wiggles' in the solution, which may be small and hidden in the L^2 norm, are fully exposed and thus controllable in the functional norm.

Scaling results are presented below for a 3D Stokes simulation of flow between two plates with a cylindrical obstacle. Fig. 6.1 shows the grid and the domain for the simulation. The grid is unstructured. Fig. 6.2 shows the resulting Stokes flow as a simulation result obtained with our codes.

Table 6.1 compares the algorithmic scalability of the Falgout and PMIS coarsenings for this problem. Conjugate Gradient (CG) acceleration could be used as a Krylov accelerator, because the FOSLS-matrices have the advantageous property of being symmetric. This results in some memory savings as compared to GMRES(10) acceleration because only two previous iteration vectors need to be stored. Because this PDE system is highly coupled with 13 degrees of freedom per node, approximately

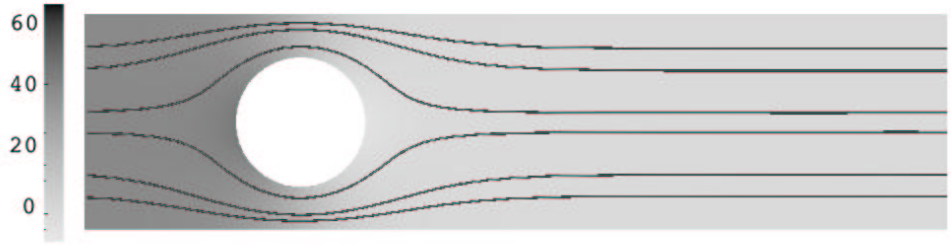

FIG. 6.2. *3D Stokes flow test problem. Gray scales represent the pressure, and the streamlines of the fluid flow are shown.*

Method	p	dof	C_{op}	$\#lev$	s_{avg}	t_{setup}	t_{solve}	its	t_{tot}
Falgout	1	55359	3.06	11	770.4	7.94	50.35	28	58.28
	8	422463	2.73	13	942.7	168.32	132.93	48	301.25
	64	3224991	3.10	16	1079.0	1280.42	689.74	91	1970.16
PMIS	1	55359	1.34	6	485.9	7.11	18.35	21	25.46
	8	422463	1.37	7	598.6	53.95	43.38	34	97.33
	64	3224991	1.40	8	733.1	333.16	125.86	52	459.02

TABLE 6.1
AMG-CG with different coarsenings.

300 nonzeros are present in every matrix row on the finest grid level, on average. Parallel scaling tests were performed with approximately 55,000 degrees of freedom per processor. Strength treshold $\alpha = 0.9$ was used for Falgout coarsening in order to keep complexity at an acceptable level. Lower α would allow for better convergence, but at the cost of prohibitively high complexity.

The table confirms, for this PDE system, the results that were shown above for scalar PDEs. The PMIS coarsening uses less than half the memory of Falgout, the stencil size grows substantially less, and both setup times and solve times are about four times faster for PMIS than for Falgout coarsening. The PMIS results are also more scalable than the Falgout results. One can note that the setup and solution times, in absolute terms, scale, in fact, poorly for both approaches. This is mostly due to a software problem beyond our control on the machine we had available for these tests, which forced us to use slow plain ethernet communication between cluster nodes. In any case, in this paper it is our main goal to compare PMIS and RS-based coarsenings, and the results clearly show that PMIS coarsening performs much better than Falgout coarsening for this problem. It is interesting that, for this complex PDE system problem, PMIS leads to much less iterations than Falgout coarsening. This behavior was not seen in the previously discussed scalar problems, and, even though we do not have an immediate explanation for this, it may point to an additional advantage of the PMIS approach for PDE systems, to be confirmed by further tests and analysis.

7. Conclusions. In this paper, simple coarsening schemes for parallel AMG were proposed, that are based solely on enforcing a maximum independent set prop-

erty, resulting in coarse grids that are sparser than the grids obtained by adhering to the original Ruge-Stueben coarsening heuristics. The new PMIS and HMIS coarsening techniques remedy memory and execution time complexity growth for large 3D problems, and, combined with Krylov acceleration, the resulting AMG-Krylov methods also tend to perform well in terms of the number of iterations required for convergence. Numerical results were presented showing that for an extensive class of large 3D problems on parallel machines in the several 1,000-processor class, the newly proposed methods use less than half the memory, and require less than half the computer time of AMG methods that use a Ruge-Stueben type coarsening. Also, for the problem sizes and machines tested, the scalability in memory and execution time (especially AMG setup times) is generally much better than for existing approaches. Efficient and scalable results were obtained for a large class of scalar and system problems on both structured and unstructured grids. However, for some difficult problems, including rotated anisotropic problems and problems with large jumps in coefficients, standard AMG interpolation that only relies on nearest neighbors for interpolation, is not sufficiently accurate on the coarse grids that result from the new coarsenings, and Krylov acceleration turns out not to be effective in this case. For these difficult cases interpolation formulas with an extended range may need to be considered, for instance techniques along the lines of the multi-pass interpolation proposed by Stueben [12]. Such improved interpolation formulas and their use on grids coarsened with the PMIS and HMIS techniques, are the subject of further research.

Acknowledgments.
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. It was also 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 Laboratory under contract number 15268, and the National Science Foundation under VIGRE grant number DMS-9810751 and grant number DMS-0410318.

REFERENCES

[1] A. Brandt, S. F. McCormick, and J. W. Ruge, Algebraic multigrid (AMG) for automatic multigrid solutions with application to geodetic computations. Report, Inst. for Computational Studies, Fort Collins, Colo., October 1982.

[2] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial* (SIAM, Philadelphia, PA, second ed., 2000).

[3] Z. Cai, Thomas A. Manteuffel, and Stephen F. McCormick, First-order system least squares for second-order partial differential equations: Part II, *SIAM J. Numer. Anal.*, 34 (1997), 425–454.

[4] A. J. Cleary, R. D. Falgout, V. E. Henson, and J. E. Jones, Coarse grid selection for parallel algebraic multigrid, in *Proceedings of the fifth international symposium on solving irregularly structured problems in parallel* (Springer-Verlag, New York, 1998).

[5] A. J. Cleary, R. D. Falgout, V. E. Henson, J. E. Jones, T. A. Manteuffel, S. F. McCormick, G. N. Miranda, and J. W. Ruge, Robustness and scalability of algebraic multigrid, *SIAM Journal on Scientific Computing*, 21 (2000), 1886–1908.

[6] V. E. Henson and U. M. Yang, BoomerAMG: a parallel algebraic multigrid solver and preconditioner, *Applied Numerical Mathematics* 41 (2002) 155–177.

[7] M. T. Jones and P. E. Plassman, A parallel graph coloring heuristic, *SIAM Journal on Scientific Computing* 14 (1993) 654–669.

[8] W. Joubert and J. Cullum, Scalable algebraic multigrid on 3500 processors, Los Alamos National Laboratory Technical Report No. LAUR03-568. Submitted to *Electronic Transactions on Numerical Analysis* (2003).

[9] A. Krechel and K. Stüben, Parallel algebraic multigrid based on subdomain blocking, *GMD*

Report 71, GMD, Sankt Augustin, Germany, submitted to Parallel Computing.
[10] M. Luby, A simple parallel algorithm for the maximal independent set problem, *SIAM Journal on Computing* 15 (1986) 1036–1053.
[11] J. W. Ruge and K. Stüben, Algebraic multigrid (AMG), in : S. F. McCormick, ed., *Multigrid Methods, vol. 3 of Frontiers in Applied Mathematics* (SIAM, Philadelphia, 1987) 73–130.
[12] K. Stüben, Algebraic multigrid (AMG): an introduction with applications, in : U. Trottenberg, C. Oosterlee and A. Schüller, eds., *Multigrid* (Academic Press, 2000).
[13] U. Trottenberg, C. Oosterlee and A. Schüller, *Multigrid* (Academic Press, 2000).