SIAM J. SCI. COMPUT.
Vol. 29, No. 5, pp. 1825–1853

A GREEDY STRATEGY FOR COARSE-GRID SELECTION*

S. MACLACHLAN† AND YOUSEF SAAD†

Abstract. Efficient solution of the very large linear systems that arise in numerical modeling of real-world applications is often only possible through the use of multilevel techniques. While highly optimized algorithms may be developed using knowledge about the origins of the matrix problem to be considered, much recent interest has been in the development of purely algebraic approaches that may be applied in many situations, without problem-specific tuning. Here, we consider an algebraic approach to finding the fine/coarse partitions needed in multilevel approaches. The algorithm is motivated by recent theoretical analysis of the performance of two common multilevel algorithms, multilevel block factorization and algebraic multigrid. While no guarantee on the rate of coarsening is given, the splitting is shown to always yield an effective preconditioner in the two-level sense. Numerical performance of two-level and multilevel variants of this approach is demonstrated in combination with both algebraic multigrid and multilevel block factorizations, and the advantages of each of these two algorithmic settings are explored.

Key words. iterative methods, algebraic multigrid (AMG), algebraic preconditioners

AMS subject classifications. 65F10, 65N55, 68Q25

DOI. 10.1137/060654062

1. Introduction.

Much of modern scientific computing is driven by the need to solve increasingly large linear systems that arise from discretization of mathematical models of physical systems. The matrices of these linear systems are often not only large but also ill-conditioned. While classical methods may be efficient for smaller systems, the increasing demand for accuracy in numerical models requires a more efficient approach.

When full details of the driving application and its discretization are available, efficient multigrid solvers and multilevel preconditioners may be naturally defined. Coarse-level acceleration ideas were proposed as early as 1935 in the work of Southwell [33], while the potential efficiency of multilevel ideas was first demonstrated in the theoretical work of Fedorenko [16, 17] and Bakhvalov [2] in the 1960s. Geometric multigrid methods [10, 20], first used for computation in the 1970s, can be shown to be optimally efficient for simple problems on regular meshes. Hierarchical basis approaches based on detailed knowledge of the finite-element discretization also lead to effective preconditioners [3]. While techniques based on such complete knowledge are often the most efficient, it is not known how to construct optimal solution strategies for all problems. Further, a good discretization-based preconditioner for one problem may not work at all for a similar problem, without substantial modification.

Algebraic multigrid (AMG) methods, first proposed in [4] and later analyzed and tested in [29], offer performance comparable to geometric multigrid on many problems without requiring as much a priori knowledge. The goal of an AMG algorithm is to use algebraic means to create the multigrid hierarchy based solely on the linear system to be solved. While there are many variants of the original approach, in all cases this amounts to choosing a coarse set of points, and then defining intergrid

*Received by the editors March 10, 2006; accepted for publication (in revised form) August 14, 2006; published electronically September 28, 2007. This work was supported by NSF under grant ACI-0305120, and by the Minnesota Supercomputing Institute.
http://www.siam.org/journals/sisc/29-5/65406.html
†Department of Computer Science and Engineering, University of Minnesota, 200 Union Street S.E., Minneapolis, MN 55455 (maclach@cs.umn.edu, saad@cs.umn.edu).

transfer operators and a coarse-scale operator that provide a good correction to errors that are slow to be reduced by relaxation. AMG has been successfully applied to many problems but is known to be most effective for those that arise from standard discretizations of elliptic differential equations [12].

The algebraic recursive multilevel solver (ARMS) approach, introduced in [30], is a multilevel block factorization approach based on incomplete LU (ILU) factorizations. The multilevel block factorization approach relies on splitting the matrix into 2×2 block form, then approximately inverting one diagonal block, and then computing a Schur complement to form a preconditioner [1, Chap. 9]. In ARMS, the inversion is accomplished using a threshold-based incomplete LU factorization (ILUT) [31] approach to compute sparse LU factors, which can then be reused in forming an approximation to the needed Schur complement.

Both the AMG and ARMS approaches require a good splitting into fine and coarse grids in order to achieve the best possible performance. To be considered good, a splitting must give a coarse grid that is sufficiently smaller than the original mesh yet still allows an effective preconditioner to be constructed. In AMG, this means that the coarse grid is big enough that the coarse-grid correction can effectively address all errors that are prohibitively slow to be reduced by relaxation. Many heuristics have been proposed to do this, primarily based on independent set algorithms, as were originally used in [29]. For ARMS, the primary goal of the coarse-grid selection algorithm is to allow for an accurate sparse factorization of the fine-grid block, plus an accurate sparse approximation of the coarse-grid Schur complement.

Theoretical convergence analysis of ARMS and AMG give implicit suggestions of ways to choose the coarse grid, but there is no easy explicit connection between the selection of the coarse-grid points and the convergence of the overall scheme. The idea of compatible relaxation (CR) [6, 8, 15, 23] provides a relationship between the properties of the fine-scale block of the matrix and the overall performance of AMG. Similarly, the analysis in [27, 28] provides a relationship between the preconditioning of the fine-scale block within ARMS and the overall quality of the preconditioner. Brief reviews of the AMG and ARMS methodologies and their theoretical analyses are given in section 2.

Here, we use these theoretical results to motivate a coarse-grid selection criterion and algorithm. The optimal coarse-grid selection criterion is seen to lead to an NP-complete coarsening algorithm and, so, a greedy algorithm is proposed to approximately maximize the measure of a good splitting. The resulting algorithm is shown to always satisfy the conditions on the fine-scale block in both AMG and ARMS theory; however no guarantee of reduction in the coarse-grid size is possible. This greedy coarsening strategy and its theoretical analysis are presented in section 3.

Numerical results are given in section 4 that show the algorithm works well for many problems. Of particular interest is the comparison between the performance of AMG and ARMS. Here, it is necessary to distinguish between the efficiency and the robustness of an algorithm. Efficiency is easily measured in terms of error reduction versus computational cost. Robustness, on the other hand, is difficult to quantify precisely. Here, we consider the robustness of an algorithm to be indicated by similar performance on a wide variety of problems. Our numerical results demonstrate the efficiency of the AMG-based algorithm for finite-element discretization of elliptic PDEs. The ARMS-based algorithm is seen to be less efficient than AMG, when AMG works well, but more robust.

2. Background and motivation. The use of multilevel solution techniques has become ubiquitous in the numerical solution of PDEs, and has also begun to

play an important role in more general numerical linear algebraic contexts. While many of these techniques make use of more knowledge than is present in the linear system of equations to be solved (e.g., geometric multigrid [10], hierarchical basis transformations [3]), we concentrate here on purely algebraic approaches, such as the ARMS technique [30] and AMG methods [4, 29]. Such algebraic techniques presume a splitting of the overall set of degrees of freedom into two disjoint sets, typically called the fine- and coarse-level degrees of freedom that are used to partition the given matrix.

Consider the matrix equation, $A\mathbf{x} = \mathbf{b}$, and suppose that the degrees of freedom in $\mathbf{x}$ and $\mathbf{b}$ are partitioned into two disjoint sets, F and C. We consider the matrix, A, to be already reordered, so that

$$\begin{bmatrix} A_{ff} & -A_{fc} \\ -A_{cf} & A_{cc} \end{bmatrix} \begin{pmatrix} \mathbf{x}_f \\ \mathbf{x}_c \end{pmatrix} = \begin{pmatrix} \mathbf{b}_f \\ \mathbf{b}_c \end{pmatrix}. \tag{1}$$

2.1. Multilevel block factorization. Multilevel block factorization techniques [1, Chap. 9] (such as ARMS [30]), make use of this block form to factor A in terms of A_{ff}^{-1}. Writing

$$A = \begin{bmatrix} I & 0 \\ -A_{cf}A_{ff}^{-1} & I \end{bmatrix} \begin{bmatrix} A_{ff} & 0 \\ 0 & \hat{A}_{cc} \end{bmatrix} \begin{bmatrix} I & -A_{ff}^{-1}A_{fc} \\ 0 & I \end{bmatrix}, \tag{2}$$

where $\hat{A}_{cc} = A_{cc} - A_{cf}A_{ff}^{-1}A_{fc}$ denotes the Schur complement of A, we can solve for $\mathbf{x}_f$ and $\mathbf{x}_c$ as in Algorithm 1.

ALGORITHM 1 (block factorization solve).

1. $\mathbf{y}_f = A_{ff}^{-1}\mathbf{b}_f$
2. $\mathbf{y}_c = \mathbf{b}_c + A_{cf}\mathbf{y}_f$
3. *Solve* $\hat{A}_{cc}\mathbf{x}_c = \mathbf{y}_c$
4. $\mathbf{x}_f = \mathbf{y}_f + A_{ff}^{-1}A_{fc}\mathbf{x}_c$

Such a solution technique is, of course, only useful if we can easily solve systems with A_{ff} and $\hat{A}_{cc}$. In practice, multilevel block factorizations can lead to good preconditioners by approximating these operators by ones that can be inverted in a more computationally efficient manner. The ARMS algorithm [30] utilizes an incomplete LU factorization to approximate A_{ff}^{-1}, both in Steps 1 and 4 of Algorithm 1, and in computing an approximate Schur complement, which is approximately inverted by a recursive application of the ARMS methodology.

The action of the two-level ARMS preconditioner on a residual, $\mathbf{r} = \left(\begin{smallmatrix} \mathbf{r}_f \\ \mathbf{r}_c \end{smallmatrix}\right)$, can be easily explained in terms of a modified block factorization, rewriting (2) as

$$A = \begin{bmatrix} L_{ff} & 0 \\ -A_{cf}U_{ff}^{-1} & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & \hat{A}_{cc} \end{bmatrix} \begin{bmatrix} U_{ff} & L_{ff}^{-1}A_{fc} \\ 0 & I \end{bmatrix},$$

where $A_{ff} = L_{ff}U_{ff}$. Taking ILU factors, $L \approx L_{ff}$ and $U \approx U_{ff}$, and letting S^{-1} approximate the action of the inverse of $\hat{A}_{cc}$, we can write the ARMS preconditioner as in Algorithm 2.

ALGORITHM 2 (action of ARMS preconditioner on residual, $\mathbf{r}$).

1. $\mathbf{y}_f = L^{-1}\mathbf{r}_f$
2. $\mathbf{y}_c = \mathbf{r}_c + A_{cf}U^{-1}\mathbf{y}_f$
3. $\mathbf{x}_c = S^{-1}\mathbf{y}_c$
4. $\mathbf{z}_f = \mathbf{y}_f + L^{-1}A_{fc}\mathbf{x}_c$
5. $\mathbf{x}_f = U^{-1}\mathbf{z}_f$

Note that Algorithms 1 and 2 are, in fact, equivalent, if $A_{ff} = LU$ (i.e., if $L = L_{ff}$ and $U = U_{ff}$) and $S^{-1} = \hat{A}_{cc}^{-1}$.

It is apparent that the success of the ARMS preconditioner depends on how well the action of A_{ff}^{-1} can be approximated by the ILU solve. This, in turn, depends directly on the properties of the A_{ff} block of A. Notay relates the condition number of the preconditioned system, $B^{-\frac{1}{2}}AB^{-\frac{1}{2}}$, where B represents the action of the (approximate) multilevel block factorization, as in Algorithms 1 and 2, to the spectral equivalence of A_{ff} and $\hat{A}_{cc}$ and their approximations [27, 28]. Paraphrasing Theorem 9 of [28], we have the following result.

THEOREM 1. *Consider a symmetric and positive-definite matrix, A, partitioned as in* (1). *Let the approximate multilevel block factorization, B, be given by*

$$B = \begin{bmatrix} I & 0 \\ -A_{cf}D_{ff}^{-1} & I \end{bmatrix} \begin{bmatrix} D_{ff} & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & -D_{ff}^{-1}A_{fc} \\ 0 & I \end{bmatrix},$$

where D_{ff} and S are both symmetric and positive definite. Assume that $\lambda_{min}(D_{ff}^{-1}A_{ff}) \geq 1$ and that $\begin{bmatrix} D_{ff} & -A_{fc} \\ -A_{cf} & A_{cc} \end{bmatrix}$ is positive semidefinite.

Then,

$$\frac{\lambda_M}{\lambda_m} \leq \kappa\left(B^{-\frac{1}{2}}AB^{-\frac{1}{2}}\right) \leq \left(1 + \sqrt{1 - \lambda_M^{-1}}\right)^2 \frac{\lambda_M^2 \nu_M}{\min(\lambda_m, \nu_m)}, \tag{3}$$

where

$$\lambda_m = \lambda_{min}(D_{ff}^{-1}A_{ff}),\ \lambda_M = \lambda_{max}(D_{ff}^{-1}A_{ff}),$$
$$\nu_m = \lambda_{min}(S^{-1}\hat{A}_{cc}),\ \nu_M = \lambda_{max}(S^{-1}\hat{A}_{cc}),$$

are the extremal eigenvalues of $D_{ff}^{-1}A_{ff}$ and $S^{-1}\hat{A}_{cc}$.

While [27, 28] provide tighter bounds than those in inequality (3), we emphasize the role that good spectral equivalence to the A_{ff} block plays in these bounds: The overall preconditioner can be no better than the equivalence between D_{ff} and the A_{ff} block, and good equivalence implies the existence of a well-conditioned block factorization preconditioner for the overall system (using $S = \hat{A}_{cc}$). Thus, constructing the partition such that we know a D_{ff} with good equivalence to the A_{ff} block is an attractive approach to realizing an efficient preconditioner for A. A much harder question is achieving good spectral equivalence between the true Schur complement, $\hat{A}_{cc}$, and its approximation, S, especially as $\hat{A}_{cc}$ is, in practice, not explicitly available within the computation.

2.2. Algebraic multigrid. Originally proposed in [4] and further explained in [29], the Algebraic Multigrid (AMG) methodology has, in fact, grown to encompass a number of algorithms focused on the theme of complementing a given relaxation procedure by an algebraically determined correction from a coarser subspace. This collection of algorithms can be roughly parameterized by the choices of the multigrid components: relaxation, intergrid transfer operators, the (nodal) coarse grid, and the coarse-grid operator. Once these choices have been made, the AMG V-cycle can be described as a preconditioner or, more commonly, through its error propagation operator,

$$(I - B_{\text{AMG}}^{-1}A) = (I - M^{-T}A)^{l_2}(I - PA_C^{-1}RA)(I - M^{-1}A)^{l_1},$$

where $I - M^{-1}A$ is the error propagation operator of the chosen relaxation scheme (e.g., Jacobi or Gauss–Seidel), l_1 and l_2 are parameters giving the number of pre- and postrelaxations to be used, R is the restriction operator transferring residuals from the fine grid to the coarse grid, A_C is the coarse-grid operator, and P is the interpolation operator, transferring corrections from the coarse grid to the fine grid.

Choices of the multigrid components can be quite varied. Typical relaxation schemes are the Jacobi and Gauss–Seidel iterations, but other schemes, including symmetric Gauss–Seidel and ILU factorizations, are also possible. Choice of the coarse grid may be done based on the classical AMG definitions of strength of connection [29], using independent set algorithms (possibly modified for parallel processing environments) [14, 21], based on modified strength measures [7], or using CR [6, 8, 15, 23] (as discussed in more detail below). Once the coarse grid has been chosen, the dimensions of the intergrid transfer operators, R and P, are set, and their nonzero structure and values can be determined. Once R and P are chosen, a coarse-grid operator, A_C, may be chosen to solve for the coarse-grid correction.

If A is assumed to be symmetric and positive definite, a variational problem may be defined to simplify the problem of choosing R, P, and A_C. Note the part of the multigrid error propagation operator that is associated with the coarse-grid correction step, $I - PA_C^{-1}RA$. Whatever coarse-grid correction is calculated, error is reduced only within the range of P. Thus, a reasonable goal is to choose the coarse-grid correction to minimize the error after correction, over all corrections in the range of P—that is, to choose $\mathbf{w}_c$, such that $\|\mathbf{e} - P\mathbf{w}_c\|_A = \min$, where $\mathbf{e}$ is the error before the coarse-grid correction step, and the norm, defined as $\|\mathbf{v}\|_A^2 = \langle A\mathbf{v}, \mathbf{v}\rangle$, is chosen so that the minimization is achievable. Choosing this variational approach requires the coarse-grid correction, $\mathbf{w}_c$, to satisfy the equation $P^TAP\mathbf{w}_c = P^TA\mathbf{e}$. Noting that $A\mathbf{e} = \mathbf{r}$, the residual before coarse-grid correction, we see that this prescribes the choices of $R = P^T$ and $A_C = P^TAP$ for any given P. Note that choosing P and exactly computing $A_C = P^TAP$ is similar to Tismenetsky's strategy for computing approximate Schur complements [35], although the usual focus in multigrid is on appropriately complementing relaxation.

Thus, the problem of defining R, P, and A_C is reduced to one of choosing P such that the range of P adequately complements the space over which relaxation is effective. The classical AMG algorithm [29] does this by collapsing certain off-diagonal connections based on the assumption that the near-null space of A is accurately represented by the constant vector. If the nature of errors that are slow to be reduced by relaxation is not known, the adaptive AMG method [9] may be used to expose prototypes of such errors to be used in the definition of interpolation. Energy-minimization principles may also be used to define the multigrid interpolation operators [36, 37].

Even in the variational setting, the overall performance of multigrid depends on the complementarity obtained between the relaxation procedure and the coarse-grid correction. In particular, the key is choosing a coarse grid such that a good definition of interpolation is possible. The principle of compatible relaxation (CR), first introduced by Brandt [6], is based on the idea that the ability to define a good interpolation operator is tied to the relaxation it must complement. In [6], Brandt defines CR as "a modified relaxation scheme which keeps the coarse-level variables invariant" and states that "a general measure for the quality of the set of coarse variables is the convergence rate of the compatible relaxation." Defined in this manner, several possible versions of CR are possible [23], the most common of which is relaxation only on the unknowns associated with the fine grid (so-called F-relaxation [34]).

Falgout and Vassilevski [15] analyze the performance of AMG methods for symmetric and positive-definite matrices, A, based on a measure that accounts for the role of relaxation in complementing the variational coarse-grid correction process. For a relaxation scheme with convergent error propagation operator $I - M^{-1}A$, define the measure

$$\mu(P, \mathbf{e}) = \frac{\langle M(M + M^T - A)^{-1} M^T (\mathbf{e} - P\mathbf{e}_c), (\mathbf{e} - P\mathbf{e}_c)\rangle}{\langle A\mathbf{e}, \mathbf{e}\rangle},$$

where $M + M^T - A$ is invertible because $\|I - M^{-1}A\|_A < 1$. This measure takes into account the performance of relaxation in measuring the effectiveness of a given interpolation operator, P, beyond what is normally considered in the standard eigenvector-approximation criterion for AMG [5, 26]. If, for a given choice of P, $\mu(P, \mathbf{e}) < k$ for all $\mathbf{e} \neq \mathbf{0}$, then it can be shown that $\|I - B_{\text{AMG}}^{-1}A\|_A < 1 - \frac{1}{k}$ [15, Theorem 2.2] for $l_1 = l_2 = 1$. Thus, $\mu(P, \mathbf{e})$ can be seen as a measure of the performance of the resulting AMG scheme for a chosen relaxation, coarse-grid, and interpolation operator.

The idea of CR may be used to bound the optimal value,

$$k^\star = \min_P \max_{\mathbf{e} \neq \mathbf{0}} \mu(P, \mathbf{e}),$$

of k over all possible interpolation operators, P, from the same coarse set. Paraphrasing [15, Theorem 5.1], we have the following result.

THEOREM 2. *Let A be a symmetric and positive-definite matrix. Assume that the relaxation method with error propagation operator $I - M^{-1}A$ is convergent ($\|I - M^{-1}A\|_A < 1$). Define $S_M = \frac{1}{2}(M + M^T)$ to be the symmetric part of M, and let $\omega = \lambda_{max}(S_M^{-1}A)$. Let Δ measure the difference between M and S_M where, for all $\mathbf{v}$, $\mathbf{w}$,*

$$\langle M\mathbf{v}, \mathbf{w}\rangle \leq \Delta \langle S_M\mathbf{v}, \mathbf{v}\rangle^{\frac{1}{2}} \langle S_M\mathbf{w}, \mathbf{w}\rangle^{\frac{1}{2}}.$$

Further, take $\rho_{ff} = \|I - M_{ff}^{-1}A_{ff}\|_{A_{ff}}$, where M_{ff} is the fine-scale part of the matrix M, as in (1).

Then,

$$k^\star \leq \frac{\Delta^2}{2 - \omega} \frac{1}{1 - \rho_{ff}},$$

and $0 < \omega < 2$.

Here, the bound on $k^\star$ gives a bound on the best possible measure of a two-level AMG algorithm with a given relaxation (prescribed by M) and partition into fine and coarse grids. For symmetric smoothers ($M = M^T$), $\Delta = 1$, and this inequality gives an upper bound on the best possible convergence factor for an AMG method with a particular coarse grid, which depends on the relaxation method chosen and the performance of CR. In particular, if CR is quick to converge (and ω is bounded away from 2, meaning that the slowest to converge modes of the symmetric relaxation, $I - S_M^{-1}A$, are the lowest-energy modes of $S_M^{-1}A$), then there exists an interpolation operator, P, that gives good multigrid performance. In general, choosing such a P that also results in an easy-to-invert coarse-grid operator, P^TAP, is difficult, as we seek to retain significant sparsity in P.

The reduction-based AMG algorithm (AMGr) presumes that a partition of the degrees of freedom has been chosen so that CR is fast to converge [24]. In particular,

AMGr is based on the assumption of a close spectral equivalence between A_{ff} and a fine-scale relaxation operator, D_{ff}, as in [24, Theorem 1].

THEOREM 3. *Let A be symmetric and positive definite, partitioned as in* (1). *Suppose that*

$$\langle D_{ff}\mathbf{x}_f, \mathbf{x}_f\rangle \leq \langle A_{ff}\mathbf{x}_f, \mathbf{x}_f\rangle \leq (1+\epsilon)\langle D_{ff}\mathbf{x}_f, \mathbf{x}_f\rangle$$

for all $\mathbf{x}_f$, and that $\left[\begin{smallmatrix} D_{ff} & -A_{fc} \\ -A_{cf} & A_{cc} \end{smallmatrix}\right]$ is positive semidefinite. Consider the two-level multigrid algorithm with fine-grid only relaxation defined by its error propagation operator,

$$(I - M^{-1}A) = \left(I - \frac{2}{2+\epsilon}\begin{bmatrix} D_{ff}^{-1} & 0 \\ 0 & 0 \end{bmatrix} A\right),$$

and interpolation operator,

$$P = \begin{bmatrix} D_{ff}^{-1}A_{fc} \\ I \end{bmatrix}.$$

Then, the multigrid cycle with l prerelaxations, variational coarse-grid correction, and l postrelaxations has error propagation operator

$$MG = (I - M^{-1}A)^l(I - P(P^TAP)^{-1}P^TA)(I - M^{-1}A)^l,$$

which satisfies

$$\|MG\|_A \leq \frac{\epsilon}{1+\epsilon}\left(1 + \left(\frac{\epsilon^{2l-1}}{(2+\epsilon)^{2l}}\right)\right) < 1.$$

Now, under the additional assumptions of Theorem 3 (over those of Theorem 2), we can not only say that there exists a P that gives good multigrid convergence, but also give the form of one such P. Notice the similarities between the assumptions for Theorems 1 and 3; both require that $\lambda_{\min}(D_{ff}^{-1}A_{ff}) \geq 1$, and that the matrix, $\left[\begin{smallmatrix} D_{ff} & -A_{fc} \\ -A_{cf} & A_{cc} \end{smallmatrix}\right]$, is semidefinite. The explicit choice of a coarse-grid operator in the AMGr algorithm gives a bound on the multigrid convergence factor dependent solely on the maximum eigenvalue of $D_{ff}^{-1}A_{ff}$, which is also the dominant factor in the bound of Theorem 1.

Thus, for both multilevel block factorizations and (nodal) AMG, tight spectral equivalence between the fine-scale preconditioner or relaxation operator, D_{ff}, and A_{ff} is needed to guarantee good performance of the overall solver. Constructing the fine-grid set so that this is always possible is the subject of the remainder of this paper.

3. The greedy coarsening algorithm. In [32], the strategy of reordering the columns and rows of a nonsymmetric matrix to ensure diagonal dominance of the A_{ff} block was introduced. There, the partitioning was accomplished by ordering the columns and rows of the matrix in order of decreasing dominance, and choosing, as A_{ff}, a subblock of A with sufficiently dominant rows. The algorithm defined here is similar in style, in that it chooses rows for A_{ff} in a greedy manner, but it also takes into account the added dominance within the A_{ff} block that occurs when points are added to the coarse set.

In order to make decisions on dominance to select the A_{ff} block, we must have some sort of measure of the dominance that is sought. We consider choosing D_{ff}

as a simple, diagonal preconditioner or relaxation method, so that we seek classical rowwise diagonal dominance of A_{ff}. Writing the partition of the degrees of freedom into fine and coarse sets as $F \cup C$, we say that row i of A_{ff} has θ dominance if

$$|a_{ii}| \geq \theta \sum_{j \in F} |a_{ij}|.$$

Note that, by this definition, classical diagonal dominance is equivalent to $\theta = \frac{1}{2}$ dominance, as the diagonal, a_{ii}, is included in the sum on the right side. Then, the diagonal dominance associated with row i is given by

$$\theta_i = \frac{|a_{ii}|}{\sum_{j \in F} |a_{ij}|}.$$

Note that the measure, θ_i, depends on the selection of F- and C-points.

Ideally, we would like to choose F as the largest set of degrees of freedom such that for every $i \in F$, $\theta_i \geq \theta$. That is, we would like F to be the largest subset possible such that A_{ff} is θ dominant. This would guarantee good equivalence of a diagonal matrix, D_{ff}, to A_{ff}, as required in the theorems of section 2, but also yield as small a coarse-scale problem as possible. Formally, we would seek to maximize $|F|$ (where $|F|$ denotes the size of the set, F) under the constraint that for every $i \in F$, $\frac{|a_{ii}|}{\sum_{j \in F} |a_{ij}|} \geq \theta$ for a chosen θ. Defining the binary variables

$$f_i = \begin{cases} 1 & \text{if } i \in F, \\ 0 & \text{if } i \in C, \end{cases}$$

the problem can be posed in optimization form:

$$\begin{aligned} &\text{maximize } \sum_{i=1}^{n} f_i, \\ &\text{subject to } f_i \in \{0,1\} \ \forall i, \\ &\qquad |a_{ii}| \geq \theta \sum_{j \in F} |a_{ij}| \text{ if } f_i \neq 0. \end{aligned}$$

Note that the sum over $j \in F$ can be converted into a sum over all j, by weighting the terms by f_j, $\sum_{j \in F} |a_{ij}| = \sum_j |a_{ij}| f_j$. Then, as this is a constrained optimization problem, we convert it to an integer programming problem by adding the term $\theta \|A\|_\infty (1 - f_i)$ to the left side of the inequality constraint,

$$\begin{aligned} &\text{maximize } \sum_{i=1}^{n} f_i, \\ &\text{subject to } f_i \in \{0,1\} \ \forall i, \\ &\qquad \theta \|A\|_\infty (1 - f_i) + |a_{ii}| \geq \theta \sum_j |a_{ij}| f_j. \end{aligned}$$

Note that the modified problem is equivalent to the original, as if $i \in F$, $1 - f_i = 0$, so the original constraint is imposed for all $i \in F$, but if $i \notin F$, $1 - f_i = 1$, and the constraint is automatically satisfied, as $\sum_{j \in F} |a_{ij}| \leq \|A\|_\infty$. Thus, the problem of finding the largest set, F, with the desired dominance is a 0-1 integer programming

problem, known to be NP-complete [38]. As such, we cannot expect to easily find an algorithm that is linear in the number of nonzero entries in A that finds the largest possible set F that has the desired level of diagonal dominance. Instead, we introduce here a greedy algorithm that partitions the unknowns into F and C by iteratively building these sets, adding any sufficiently diagonally dominant points that arise to F, and moving the least diagonally dominant points into C, one at a time.

To accomplish the partitioning, we introduce a third set of points, denoted by the set, U, as those points whose partitioning is, as yet, undecided in the process. Initially, then, U includes all of the degrees of freedom, and sets F and C are both empty. We introduce the dynamic measures,

$$\hat{\theta}_i = \frac{|a_{ii}|}{\sum_{j \in F \cup U} |a_{ij}|},$$

as measures of the diagonal dominance of row i amongst those rows that either are already designated as F-points, or could potentially be so designated. If, at any point, row i is determined to be sufficiently diagonally dominant, that is, $\hat{\theta}_i \geq \theta$ for some preselected threshold θ, then i is automatically added to the set of F-points (as it may only become more dominant as points are moved from U to C or F). If no points are attractive as F-points, then a point that is not attractive as a potential F-point is chosen to be moved to C, in order to make the remaining points in U more diagonally dominant. Thus, the partitioning algorithm, given in Algorithm 3, is a greedy algorithm that, at each step, makes a choice to increase the diagonal dominance of those points in U. In this way, the points in U can be viewed as candidate F-points that are either confirmed or discarded based on the dynamic values of $\hat{\theta}_i$. A common notation, which we use here, is to call $Adj(j)$ the set of nearest neighbors of j, i.e., the set $\{k : a_{jk} \neq 0\}$.

ALGORITHM 3 (greedy partitioning algorithm).

0. *Set:* $U = \{1, 2, \ldots, n\}$; $F = \emptyset$; *Compute* $\hat{\theta}_i$, $i = 1, \ldots, n$.
1. *For* $i = 1$ *to* n *do: if* $\hat{\theta}_i \geq \theta$, *then* $\begin{cases} F := F \cup \{i\}, \\ U := U \setminus \{i\} \end{cases}$
2. *While* $U \neq \emptyset$ *do:*
   (a) *Find* $j = \operatorname{argmin}_{i \in U} \{\hat{\theta}_i\}$
   (b) *Remove* j *from* U, *as it becomes a* C*-point*
   (c) *For each* $i \in U \cap Adj(j)$ *do:*
   - *Update* $\hat{\theta}_i = \frac{|a_{ii}|}{\sum_{k \in F \cup U} |a_{ik}|}$
   - *If* $\hat{\theta}_i \geq \theta$, *then* $\begin{cases} F := F \cup \{i\}, \\ U := U \setminus \{i\} \end{cases}$

As the only way for a point to be reclassified from U to F is if $\hat{\theta}_i \geq \theta$, and $\hat{\theta}_i$ is a nondecreasing function (since points are only removed from $F \cup U$), it is immediately apparent that each row within the A_{ff} block created by Algorithm 3 has (at least) θ dominance. This, in turn, suggests the existence of a diagonal preconditioner or relaxation matrix, D_{ff}, that achieves the tight equivalence bounds suggested by Theorems 1, 2, and 3. We now verify this for the case that A is symmetric and positive definite.

THEOREM 4. *Let* A *be symmetric and positive definite, partitioned as in* (1). *Choose* $\theta \in (\frac{1}{2}, 1]$ *such that*

$$\theta \leq \theta_i = \frac{a_{ii}}{\sum_{j \in F} |a_{ij}|} \quad \forall i \in F,$$

and define $D_{ff} \equiv \text{diag}(A_{ff})$. *Then,*

$$\lambda_m = \lambda_{min}(D_{ff}^{-1}A_{ff}) \geq 2 - \frac{1}{\theta},$$
$$\lambda_M = \lambda_{max}(D_{ff}^{-1}A_{ff}) \leq \frac{1}{\theta}\ .$$

Proof. Let λ be an eigenvalue of $D_{ff}^{-1}A_{ff}$. Clearly, λ is real because $D_{ff}^{-1}A_{ff}$ is similar to the symmetric matrix $D_{ff}^{-\frac{1}{2}}A_{ff}D_{ff}^{-\frac{1}{2}}$. By Gerschgorin's theorem, there is a certain index i such that λ satisfies

$$|\lambda - 1| \leq \sum_{j\in F,\ j\neq i} \left|\frac{a_{ij}}{a_{ii}}\right| = \frac{1}{\theta_i} - 1 \leq \frac{1}{\theta} - 1$$

from which the result follows immediately. ∎

The results in Theorems 1 and 3 require that the smallest eigenvalue of $D_{ff}^{-1}A_{ff}$ be at least one. By scaling the matrix D_{ff} with the scalar $2 - 1/\theta$ the smallest eigenvalue will be scaled to 1 and the largest to $1/(2\theta - 1)$.

COROLLARY 1. *Under the same assumption as those of Theorem* 4, *but with* $D_{ff} \equiv (2 - \frac{1}{\theta})\,\text{diag}(A_{ff})$, *the following bounds are satisfied:*

$$\lambda_m = \lambda_{min}(D_{ff}^{-1}A_{ff}) \geq 1, \qquad \lambda_M = \lambda_{max}(D_{ff}^{-1}A_{ff}) \leq \frac{1}{2\theta - 1}.$$

A similar result may be obtained in the same manner as Theorem 4 using a rowwise scaling of D_{ff} based on the level of dominance in each row.

COROLLARY 2. *Under the same assumptions as those of Theorem* 4, *but with* $(D_{ff})_{ii} = \left(2 - \frac{1}{\theta_i}\right)a_{ii}$ *for all* $i \in F$, *the following bounds are satisfied:*

$$\lambda_m = \lambda_{min}(D_{ff}^{-1}A_{ff}) \geq 1, \qquad \lambda_M = \lambda_{max}(D_{ff}^{-1}A_{ff}) \leq \frac{1}{2\theta - 1}.$$

Recall that the the smallest and largest eigenvalues of $D_{ff}^{-1}A_{ff}$ are the minimum and maximum (resp.) of the Rayleigh quotients $\frac{\mathbf{x}_f^T A_{ff}\mathbf{x}_f}{\mathbf{x}_f^T D_{ff}\mathbf{x}_f}$, so these corollaries are equivalent to saying that, for all $\mathbf{x}_f$,

$$\mathbf{x}_f^T D_{ff}\mathbf{x}_f \leq \mathbf{x}_f^T A_{ff}\mathbf{x}_f \leq \frac{1}{2\theta - 1}\mathbf{x}_f^T D_{ff}\mathbf{x}_f.$$

This, in turn, implies that the parameter ϵ from Theorem 3 may be chosen as $\frac{2-2\theta}{2\theta-1}$. Knowing the extremal Rayleigh quotients also allows definition of an appropriately weighted relaxation scheme, $I - \sigma D_{ff}^{-1}A_{ff}$, with minimal spectral radius for use in the bound in Theorem 2.

Theorems 1 and 3 both require the additional assumption that $\left[\begin{smallmatrix} D_{ff} & -A_{fc} \\ -A_{cf} & A_{cc} \end{smallmatrix}\right]$ be positive semidefinite. In AMGr, this requirement is important as the coarse-level matrix depends on D_{ff} through the variational formulation. For block factorization preconditioners, this condition arises indirectly, in the bounding of the Cauchy–Bunyakowski–Schwarz constant associated with the factorization using D_{ff}. In either case, it is difficult to guarantee this condition for general classes of matrices, although it is easy to construct a matrix, A, for which it holds: simply choose $A = \left[\begin{smallmatrix} D_{ff} & -A_{fc} \\ -A_{cf} & A_{cc} \end{smallmatrix}\right] + \left[\begin{smallmatrix} E_{ff} & 0 \\ 0 & 0 \end{smallmatrix}\right]$ for any semidefinite matrices $\left[\begin{smallmatrix} D_{ff} & -A_{fc} \\ -A_{cf} & A_{cc} \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} E_{ff} & 0 \\ 0 & 0 \end{smallmatrix}\right]$.

For a given matrix, A, the existence of the decomposition can be proven in the case of diagonal dominance if D_{ff} is chosen as in Corollary 2, as in the following result.

THEOREM 5. *Let A be symmetric and positive definite, partitioned as in* (1). *Further assume that A is diagonally dominant. Define the diagonal matrix, D_{ff}, as in Corollary* 2, *by* $(D_{ff})_{ii} = \left(2 - \frac{1}{\theta_i}\right)a_{ii}$ *for* $\theta_i = \frac{a_{ii}}{\sum_{j\in F}|a_{ij}|}$. *Then,* $\left[\begin{smallmatrix} D_{ff} & -A_{fc} \\ -A_{cf} & A_{cc} \end{smallmatrix}\right]$ *is positive semidefinite.*

Proof. As A is diagonally dominant, $\sum_j |a_{ij}| \le 2a_{ii}$ for every $i \in F$. Splitting the degrees of freedom into $F \cup C$, we then have

$$\begin{aligned}\sum_{j\in C}|a_{ij}| &\le 2a_{ii} - \sum_{j\in F}|a_{ij}| \\ &= \left(2 - \frac{1}{\theta_i}\right)a_{ii} = (D_{ff})_{ii}.\end{aligned}$$

Thus, all rows in $\left[\begin{smallmatrix} D_{ff} & -A_{fc} \\ -A_{cf} & A_{cc} \end{smallmatrix}\right]$ associated with an F-point are diagonally dominant. All rows associated with a C-point are also diagonally dominant, as they already were in A. So, $\left[\begin{smallmatrix} D_{ff} & -A_{fc} \\ -A_{cf} & A_{cc} \end{smallmatrix}\right]$ is diagonally dominant. But then, by Gerschgorin's theorem, it must be positive semidefinite. ∎

Remark 1 (relationship to CR). Brandt defines compatible relaxation (CR) as "a modified relaxation scheme which keeps the coarse-level variables invariant" and states that "a general measure for the quality of the set of coarse variables is the convergence rate of the compatible relaxation" [6]. Here, we do not explicitly perform the relaxation in CR, but we achieve a similar result. The construction of the coarse/fine partition, however, is such that the existence of a quickly converging fine-level Jacobi-like relaxation is guaranteed by Theorem 4 and its corollaries. As a result, the partitioning algorithm proposed here may be considered as a new approach to CR, where a property of quickly converging CR is identified (in this case, diagonal dominance) and used to construct a partition for which CR is guaranteed to converge quickly, without actually needing to perform the iteration.

4. Numerical results. To test the approach developed in section 3, we consider a variety of problems and explore some of the different options inherent in the development of multilevel algorithms based on this approach to coarse-grid selection. The results in Theorems 1 and 3 prescribe specific two-level algorithms with bounded convergence properties based only on the fine-scale equivalence of D_{ff} and A_{ff}. Theorems 1 and 2 may also be read as giving indicators of the best possible performance of a two-level algorithm based on the partitioning prescribed. A need for more general (and more expensive) treatment of the interaction between fine and coarse scales is demonstrated and addressed by using the full power of the ARMS and AMG approaches.

4.1. AMG. The reduction-based AMG algorithm discussed in Theorem 3 prescribes a simple two-level scheme that is useful when the coarse-grid selection gives a fine-scale block that is easily relaxed. We first test the diagonal-dominance-based coarse-grid selection scheme in this setting, for PDE-based problems where AMG in general (and AMGr, in particular) is expected to perform reasonably well.

The convergence result of Theorem 3 rests on the requirement that $\left[\begin{smallmatrix} D_{ff} & -A_{fc} \\ -A_{fc}^T & A_{cc} \end{smallmatrix}\right]$ be positive semidefinite, although this condition is not explicitly enforced in the coarsening algorithm. While established only as a sufficient condition, fast convergence of the AMGr algorithm is typically not found when it is violated. For this reason, we

begin by testing problems where we know the needed definiteness occurs, when A itself is diagonally dominant (as established in Theorem 5). For these tests, we consider the matrices that arise from the bilinear finite-element discretizations of the elliptic operator, $-\nabla \cdot K(x,y)\nabla p(x,y)$, on the unit square, subject to Dirichlet boundary conditions.

There are several relevant measures of the performance of the coarsening algorithm itself and of AMGr coupled with it. We record the required setup time for the algorithm, including computing the coarse grid, partitioning the fine-scale operator, computing the coarse-grid operator, and computing any necessary parameters of the method. For AMGr, we choose a random initial guess and iterate directly on the homogeneous problem, $A\mathbf{x} = \mathbf{0}$. The number of iterations needed to reduce the ℓ^2-norm of the residual by six orders of magnitude and the time taken for these iterations are recorded. These two-level results were computed using MATLAB and, thus, the times presented are intended primarily for comparison purposes; the multilevel results that follow are computed using research C codes compiled with appropriate optimization flags. As always, the reported run times are only accurate to a few hundredths of a second, and so those presented here are rounded to the nearest tenth of a second.

As the coarse grid and D_{ff} determine the convergence bounds for these algorithms, the spectral equivalence properties for A_{ff} and D_{ff} are reported for each problem and each grid resolution. In particular, we report the number of coarse-grid points, n_c, and $\epsilon = \lambda_{\max}(D_{ff}^{-1}A_{ff}) - 1$, computed from the upper spectral equivalence bounds for the fine-scale operators (which is itself computed using a few steps of Arnoldi iteration). The asymptotic convergence factor, ρ, measured in the A-norm, is computed directly using at most 200 steps of the power method. The AMGr cycle used is one with l prerelaxations followed by a coarse-grid correction and no postrelaxation. The analysis of Theorem 3 can also be performed in this setting, yielding the bound, $\|MG\|_A \leq \left(\frac{\epsilon}{1+\epsilon}\left(1+\left(\frac{\epsilon^{2l-1}}{(2+\epsilon)^{2l}}\right)\right)\right)^{\frac{1}{2}}$, where the square-root arises as the convergence of this one-sided cycle is limited by halving the number of relaxation sweeps relative to the symmetric cycle. In the results presented here, we choose $l = 3$, which, for all examples, yielded the fastest solution time for $1 \leq l \leq 5$.

Table 1 shows the performance of the two-level AMGr scheme on matrices that come from bilinear finite element discretizations of the two-dimensional diffusion equation, $-\nabla \cdot K\nabla p$. Three different choices of the diffusion coefficient, $K(x,y)$, are chosen, corresponding to the homogeneous case ($K(x,y) = 1$, the Poisson equation), the smoothly varying case ($K(x,y) = 10^{-8} + 10(x^2 + y^2)$), and the discontinuous case, where $K(x,y)$ is 1 in 80% of the elements, chosen at random, and 10^{-8} in the remaining 20% of the elements. In all three cases, the two-level AMGr algorithm performs well, reducing the A-norm of the error by a factor of 10^6 in at most 20 iterations.

TABLE 1
Performance of two-level AMGr on test matrices from discretizations of $-\nabla \cdot K(x,y)\nabla p(x,y)$.

Coefficient	Grid	n_c	ϵ	t_{setup}	t_{solve}	# iters.	ρ
$K(x,y) = 1$	32×32	225	4.98	0.5	0.2	13	0.37
	64×64	961	4.99	7.2	0.7	13	0.37
	128×128	3969	5.00	119.3	3.1	13	0.37
$K(x,y) = 10^{-8} + 10(x^2 + y^2)$	32×32	255	4.53	0.4	0.3	17	0.51
	64×64	1012	4.69	6.4	1.2	19	0.57
	128×128	4036	4.69	106.8	5.1	18	0.55
random $K(x,y)$	32×32	277	4.61	0.5	0.3	18	0.52
	64×64	1149	4.78	6.4	1.4	19	0.59
	128×128	4715	4.86	109.2	6.7	20	0.62

Grid sizes are effectively reduced, and the resulting asymptotic convergence factors are bounded well away from unity.

In the homogeneous case, the coarsening chosen by Algorithm 3 results in uniform coarse grids. All Dirichlet boundary nodes are automatically chosen as fine-grid points (as they are already diagonally dominant). Nodes neighboring the boundary are also attractive as fine-grid points because of their inherent diagonal dominance (from the algebraic elimination of connections to the boundary nodes). Thus, the 33×33 mesh of unknowns from the 32×32 element grid is coarsened to a uniform 15×15 mesh. Similarly, the 64×64 and 128×128 element meshes are coarsened to meshes of 31×31 and 63×63 degrees of freedom. The parameter, ϵ, of the splitting stays close to 5.0 as the mesh is refined and is theoretically bounded by 9, from Corollary 2, for our choice of $\theta = 0.55$. As expected from a two-level scheme, the setup time does not scale with refinement, nor does the solve time (although this is closer). The resulting convergence factors of approximately 0.37 do not degrade with grid size, as predicted by Theorem 3, although the measured convergence factor is signficantly better than that predicted by the theory.

For the variable coefficient problems, coarsening is less uniform, as is expected given the variations in these operators. The spectral equivalence constant, ϵ, however, remains bounded below 5, leading to effective two-level solvers. Setup times again do not scale, as expected, and the lack of scaling in the iteration times is slightly more pronounced. Asymptotic convergence factors remain nicely bounded below one, and below those predicted by Theorem 3. As a result, iteration counts for reducing the A-norm of the error remain small.

4.1.1. Implementation and cost issues. For multilevel tests, we give more weight to the question of an efficient implementation of Algorithm 3. There are two potential sources of inefficiency in the partitioning algorithm—the selection of the new C-point and the updating of all neighboring F-points. In order to be able to efficiently find the absolute minimal measure, we would need to either keep a sorted list of the measures, which would need updating each time a point was put into C, or search the (unsorted) list of measures. Instead, we implement an incomplete bucket sort [13, section 9.4] based on a doubly linked-list data structure [13, section 11.2] that allows efficient identification of a point with approximately minimal measure as well as fast updates of measures.

For each potential F-point, a standard linked-list data item is created. With each linked-list item, a key for the associated node is assigned as the measure of the node. As measures of potential F-points lie in the interval $[0, \theta)$, this interval is divided into equal-sized subintervals (or buckets), and all list items whose keys lie in the given bucket are added to a single doubly linked list associated with that bucket. The nonempty bucket associated with the smallest key then contains not only the item associated with the absolute minimal measure, but also all items within some small range of the minimal measure. A single element is chosen out of this bucket as an approximately minimal measure and used as the new C-point, j, in Algorithm 3. Taking advantage of the symmetry of A, we then can look over all nonzero columns, i, in row j of A, to update $\hat{\theta}_i$ to account for the removal of j. This is easily accomplished by an indexing array that points to the linked-list item corresponding to node i. Using these structures, we no longer select the absolute minimal-measured node to remove at each iteration, but are rewarded with $O(1)$ update and find-approximate-min operations, and an $O(n)$ amount of storage for partitioning the $n \times n$ matrix, A.

As a multigrid cycle, we consider a mixture of components from the AMGr algorithm and a classical multigrid V-cycle. Full relaxation sweeps, as opposed to the

TABLE 2
Performance of multilevel AMG with AMGr-style interpolation, as in Theorem 3, on test matrices from discretizations of $-\nabla \cdot K(x,y)\nabla p(x,y)$.

Coefficient	Grid	# levels	c_G	c_A	t_{setup}	t_{solve}	# iters.	ρ
$K(x,y) = 1$	128×128	6	1.31	1.32	0.03	0.08	10	0.34
	256×256	7	1.32	1.32	0.2	0.3	10	0.36
	512×512	8	1.33	1.33	0.6	1.2	10	0.38
	1024×1024	9	1.33	1.33	2.6	4.9	10	0.40
	2048×2048	10	1.33	1.33	12.4	17.9	9	0.41
$K(x,y) = 10^{-8} + 10(x^2 + y^2)$	128×128	6	1.31	1.32	0.03	0.08	10	0.33
	256×256	7	1.32	1.32	0.2	0.3	10	0.35
	512×512	8	1.33	1.33	0.6	1.2	10	0.37
	1024×1024	9	1.33	1.33	2.6	5.0	10	0.39
	2048×2048	10	1.33	1.33	12.3	19.9	10	0.41
random $K(x,y)$	128×128	6	1.38	1.75	0.08	0.2	17	0.59
	256×256	7	1.39	1.83	0.5	0.7	17	0.62
	512×512	8	1.40	1.89	3.1	3.2	18	0.68
	1024×1024	9	1.40	1.93	33.7	14.9	21	0.77
	2048×2048	10	1.40	1.98	907.6	88.2	31	0.91

F-relaxation used in AMGr, are prevalent in AMG and, so, we use a full sweep of Gauss–Seidel relaxation for both pre- and postrelaxation within the V-cycle, ordered so that coarse points are relaxed first on the downward part of the cycle, but that F-points are relaxed first on the upward part of the cycle. Interpolation is defined as in AMGr (cf. Theorem 3), using the diagonal matrix, D_{ff}, as in Corollary 2. The problem is coarsened until the next coarsest grid would have fewer than 8 degrees of freedom, and the coarsest-grid problem is solved using a (complete) Cholesky factorization.

We first test this AMG approach on the same problems as in Table 1, again with $\theta = 0.55$. The number of subintervals in the bucket sort is taken to be 1000, regardless of the problem size. As we no longer explicitly need the parameter, ϵ, to define relaxation, we do not calculate it for these examples. In Table 2, we show the number of levels created in the multigrid hierarchy for each problem. To measure the efficiency of the multigrid coarsening process, we report both the multigrid grid complexity and operator complexity. The grid complexity, c_G, defined as the sum of the number of grid points on each level divided by the number of grid points on the finest level, is an indicator of the rate at which the problem is coarsened. The multigrid operator complexity, c_A, defined as the sum of the number of nonzero entries in the operator on each level divided by the number of nonzero entries in the finest-grid operator is a measure of both the storage needed and computational cost of a single multigrid cycle. The times required for the setup and solve phases are reported, where solution is taken to mean a reduction in the residual by a factor of 10^6 in solving the homogeneous problem, $A\mathbf{x} = \mathbf{0}$, given a random initial guess. The number of iterations needed for convergence, as well as the asymptotic convergence factor, ρ, measured after at most 200 iterations are also reported.

Performance for the two problems with smooth coefficients is typical of geometric multigrid methods applied to these problems. The number of levels grows by one for each grid refinement, and both grid and operator complexities remain relatively constant as the problem is refined. In fact, because of the approximate minimization in Step (2)(a) of Algorithm 3, a series of fully coarsened grids is chosen for both of these problems, bounding the grid complexity by a factor of $\frac{4}{3}$. That the operator complexity is also bounded by $\frac{4}{3}$ for these problems is a consequence of this regular coarsening, coupled with the fact that, in the variational definition of the coarse-grid operator, the stencil of the coarse-to-fine interpolation operator is that of A_{fc},

resulting in coarse-grid operators that are also nine-point operators on a regular mesh. Setup times roughly scale with matrix size (although there is some slowing, likely due to the indirect indexing operations needed to define interpolation), and the solve times scale nicely with problem size. The resulting convergence factors are bounded well away from unity, although are slightly worse than typical (Ruge–Stüben) AMG performance on these problems, which would lead to convergence factors in the range of 0.1 to 0.2. The simple interpolation operator defined in Theorem 3 allows for efficient computation of the AMG interpolation and coarse-grid operators, but limits its applicability. For these problems, where the AMGr interpolation is effective, setup times for the AMG algorithm can be much lower than those of classical AMG, but typically result in poorer convergence behavior, as seen here.

Performance for the randomly discontinuous coefficient problem, as seen in Table 2, degrades with refinement of the mesh. While the grid complexity does remain nicely bounded, only slightly larger than the factor of $\frac{4}{3}$ seen in the previous examples, there is a notable growth in the operator complexities. This growth is a result of the nonuniform coarsening that is naturally chosen for this problem, coupled with the use of an interpolation stencil that exactly matches that of A_{fc}. In this way, the density of nonzeros in coarse-grid operators grows as the multigrid cycle coarsens. Such growth is compounded in further coarsenings, as new nonzero entries in the stencil of A_{fc} lead to new nonzeros in the variational coarse-grid operator, as well as denser matrix-matrix-matrix products needed to compute these products. This is most noticeable in the dramatic increase in the time needed for the setup stage as the grid is refined, a result of the effectively dense matrix computations needed to coarsen the coarsest few levels of the multigrid hierarchy for the 2048×2048 element grid problem. The obvious solution to this growth is to drop some of the small (or weak) off-diagonal connections in these growing factors, either in the choice of interpolation or in the computation of the coarse-grid operators themselves. Dropping weak off-diagonal connections in the definition of interpolation is, in fact, exactly what is done in the standard AMG approach, discussed below. The slower growth in the solve times is very closely correlated to the combination of the growth in the problem size (by a factor of 4) and the number of iterations needed to reduce the residual appropriately (by a factor of $\frac{3}{2}$ between the 1024×1024 and 2048×2048 problems). This increase in the number of iterations, in turn, is closely linked to the increase in the asymptotic convergence factors as the grids are refined. It is difficult to say (without finer grids to test) if this increase is due to a lack of optimality of the overall multigrid approach (i.e., convergence factors continue to worsen as the grid is refined), or is related to the poorer treatment of the coarsest levels of the multigrid hierarchy.

4.1.2. Importance of theoretical conditions. An even poorer example of the performance of an AMGr-like algorithm can be seen when we violate the assumption that the "reduced" matrix, $\left[\begin{smallmatrix} D_{ff} & -A_{fc} \\ -A_{cf} & A_{cc} \end{smallmatrix}\right]$, be nonnegative definite. In Theorem 5, we note that such an example cannot arise when A is diagonally dominant and, so, we consider another PDE-based example, $-\nabla \cdot K(x,y)\nabla p(x,y)$, where we choose a tensor coefficient, $K(x,y) = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 0.01 \end{smallmatrix}\right]$. As with the continuum operator, the discretization has strong connections between nodes that lie on the same line of fixed y value, but a finite-element discretization of this equation has significant nonzero connections to degrees of freedom that lie off of these lines as well. The greedy coarsening algorithm effectively recognizes these properties of strength of connection and chooses the first coarse grid to be a semicoarsened version of the fine grid, known to be effective for this problem. The AMGr interpolation operator, however, attempts to define an

interpolation that connects a fine-grid point to all of its coarse-grid neighbors. In this case, this includes interpolation from points that are not strongly related to the fine-grid point in question. As a result, the multigrid complexities grow substantially (as A_{fc} is relatively dense) and the overall algorithm performs poorly. For a 32×32 element grid, a grid complexity of 4.69 and operator complexity of 48.16 were recorded, with an asymptotic convergence factor of 0.98989.

A solution to the degradation of performance seen for the randomly discontinuous problems or overall poor performance in the anisotropic case is to combine the coarsening of Algorithm 3 with classical AMG interpolation techniques. While classical AMG [29] limits interpolation to the stencil of A_{fc}, not all nonzero entries in this stencil are used. To compensate for dropping connections in A_{fc}, as well as for strong connections in A_{ff}, the nonzero entries in a row of interpolation are also not necessarily equally proportional to the original entries of A_{fc}. Interpolation is defined based on the principle that the errors remaining after relaxation yield small residuals, so that interpolation needs to be most accurate for vectors, $\mathbf{e}$, such that $A\mathbf{e} \approx \mathbf{0}$ in a pointwise sense. Considering a fine-grid point, i, this means that

$$\begin{aligned} (A\mathbf{e})_i &= 0, \\ \text{or } a_{ii}e_i &= -\sum_{k \in C_i} a_{ik}e_k - \sum_{j \in F_i} a_{ij}e_j, \end{aligned}$$

where the sets C_i and F_i are defined through the localizations of the global coarse and fine sets: $C \cap Adj(i)$, and $F \cap Adj(i)$. Interpolation from coarse neighbors of the point, i, is relatively straightforward, so the main goal is to collapse the connections in F_i onto C_i or i itself in a way to meaningfully define an interpolation operator.

4.1.3. Combination with AMG coarsening. Classical AMG [29] categorizes the off-diagonal connections in row i as strong or weak depending on their suitability for use in interpolation. When the connection between two points is said to be strong, it is treated as an important connection to account for in interpolation, while a weak connection is to be discarded. The classical definition for the set of points that i strongly depends on is $S_i = \{j : -a_{ij} \geq \beta \max_{k \neq i}\{-a_{ik}\}\}$ for a chosen strength threshold, β [10] (other possible algebraic definitions of strong connections include those in [7, 11]). The set C_i is then defined by $C_i = C \cap S_i$, while $F_i = Adj(i) \setminus C_i$. Connections in F_i are then further partitioned into strong connections, $F_i^s = F_i \cap S_i$, and weak connections, F_i^w. Weak connections are not important in interpolation, so they are collapsed directly to the diagonal. Strong connections to fine-grid points are accounted for through an indirect interpolation, based on an assumption on the errors that are slow to be reduced by relaxation (as in [29]) or, possibly, further knowledge of such errors [9]. Making choices consistent with classical AMG gives the interpolation formula [10] for $i \in F$,

$$e_i = -\sum_{k \in C_i} \left(\frac{a_{ik} + \sum_{j \in F_i^s} \left(\frac{a_{ij}a_{jk}}{\sum_{k' \in C_i} a_{jk'}} \right)}{a_{ii} + \sum_{j \in F_i^w} a_{ij}} \right) e_k. \tag{4}$$

TABLE 3
Performance of multilevel AMG with classical AMG-style interpolation and a two-stage coarsening algorithm on test matrices from discretizations of $-\nabla \cdot K(x,y)\nabla p(x,y)$.

Coefficient	Grid	# levels	c_G	c_A	t_{setup}	t_{solve}	# iters.	ρ
$K(x,y)=1$	128×128	6	1.31	1.32	0.07	0.04	5	0.11
	256×256	7	1.32	1.32	0.3	0.2	5	0.12
	512×512	8	1.33	1.33	1.3	0.6	5	0.13
	1024×1024	9	1.33	1.33	5.3	2.5	5	0.14
	2048×2048	10	1.33	1.33	21.3	10.2	5	0.14
$K(x,y)=10^{-8}+10(x^2+y^2)$	128×128	6	1.31	1.32	0.07	0.04	5	0.11
	256×256	7	1.32	1.32	0.3	0.2	5	0.12
	512×512	8	1.33	1.33	1.3	0.6	5	0.13
	1024×1024	9	1.33	1.33	5.3	2.6	5	0.14
	2048×2048	10	1.33	1.33	21.1	10.2	5	0.14
random $K(x,y)$	128×128	10	1.50	1.97	0.1	0.07	6	0.23
	256×256	12	1.51	2.02	0.6	0.3	6	0.28
	512×512	15	1.51	2.06	2.3	1.2	6	0.35
	1024×1024	19	1.51	2.08	9.7	4.7	6	0.40
	2048×2048	22	1.52	2.10	40.5	10.0	6	0.46
anisotropic $K(x,y)$	128×128	9	1.91	2.26	0.08	0.06	5	0.13
	256×256	11	1.95	2.34	0.4	0.3	5	0.13
	512×512	13	1.96	2.39	1.5	1.0	5	0.13
	1024×1024	15	1.97	2.41	6.2	4.1	5	0.20
	2048×2048	16	1.97	2.43	27.2	17.3	5	0.20

Dividing the neighbors of a node, i, into groups of strong and weak connections is necessary to ensure that AMG interpolation does not try to connect points that have no real relationship in the matrix (or governing PDE). In order to define nontrivial interpolation by (4), we now need to ensure that C_i is nonempty. To do this, we use a two-pass coarsening routine based on that used in classical AMG [29], where the coarse grid is first chosen as a maximal independent set over the strong connections of the matrix; then a second pass is made, changing F-points to C-points, to ensure that the strong connections of i are properly accounted for in interpolation. Here, we let Algorithm 3 choose an initial partition into F and C, and then use the AMG second pass algorithm that allows changing F-points to C-points to allow for better use of (4) in defining interpolation [29]. Note that removing points from F can only improve the diagonal dominance of A_{ff}, so Theorem 4 and its corollaries will still hold for the repartitioned system. Our only concern then is to ensure that we do not enlarge C by too much in seeking good approximation properties for the resulting interpolation.

Results for AMG-style interpolation (4), using Algorithm 3 as a first pass for coarsening, augmented by the second pass described above, are shown in Table 3. Here, we choose the AMG strength threshold $\beta = 0.25$ for the isotropic problems and $\beta = 0.3$ for the anisotropic problem. Operator and grid complexities are reported, as are the number of levels in the multigrid hierarchy, showing the deepening hierarchies as the coarsening becomes less uniform. This increase in the number of levels allows AMG to retain the low operator complexities seen in Table 2, yet still address the important need of all fine-grid points for adequate strong connection to the coarse grids.

For all of these problems, we see very effective solvers, with low iteration counts and convergence factors. For the smoothly varying coefficient problems, we see the same grid and operator complexities for the AMG-based approach as for the AMGr-style interpolation, as the second pass of coarsening does not add any points to the coarse grids chosen by Algorithm 3 and the nonzero pattern of interpolation is the same. The setup times for these problems are larger, due both to the second pass of coarsening (which still checks that all points have suitable connection to the coarse grid) and the more complicated definition of interpolation. The AMG-style interpo-

lation, however, produces solvers with lower convergence factors, so the extra setup time is roughly equally offset by the lower solve time. For the randomly discontinuous-coefficient problem, we see slightly higher grid and operator complexities, but these are more evenly distributed throughout the multigrid hierarchy, so there is no longer a significant slowing of the setup stage on finer grids. The resulting solvers are slightly less efficient than those of the smooth coefficient cases but still much better than those shown in Table 2. The anisotropic problem is also well handled by this algorithm, with larger grid and operator complexities consistent with geometric multigrid approaches to this problem. As with all of these examples, setup and solve times scale nicely with the increase in problem size, and the convergence factors grow slightly, but remain bounded well below unity.

4.2. ARMS. Effective two-level block factorizations, as discussed in Theorem 1, rely on good spectrally equivalent preconditioners for both the fine-scale block and Schur complement, $\hat{A}_{cc}$. To test the use of the diagonal-dominance-based coarsening scheme with approximate block-factorization preconditioners, we first consider two-level preconditioners, making use of the equivalence from Corollary 2 to handle the fine-scale preconditioning. Preconditioning of the Schur complement block is considered both exactly and using the variational coarse-grid operator as an approximation to $\hat{A}_{cc}$.

The requirement that $\left[\begin{smallmatrix} D_{ff} & -A_{fc} \\ -A_{fc}^T & A_{cc} \end{smallmatrix}\right]$ be positive semidefinite also arises in the theory for ARMS (cf. Theorem 1), so we begin by testing the two-level ARMS algorithm on the same PDE-based test problems as we did with AMGr. Some of the measures discussed above are again relevant, including the overall setup time (choosing the coarse grid, partitioning the fine-scale operator, computing the fine-scale preconditioner, and computing the coarse-scale operator). We test the two-level ARMS algorithm as a preconditioner for conjugate gradient (CG) by choosing a right-hand side, $\mathbf{b}$, to be the result of the matrix applied to the vector of all ones, $\mathbf{b} = A\mathbf{1}$, and a random initial guess. The number of iterations and time needed to reduce the A-norm of this error by a relative factor of 10^6 are also reported. Once again, these two-level results are computed using MATLAB with the reported times useful only for comparison between themselves and those in Table 1; the multilevel results that follow are computed using research C codes compiled with the appropriate optimization flags. The reported CPU times are accurate only to a few hundredths of a second and, so, are rounded to the nearest tenth of a second.

The coarse grids are selected using the same code and parameters as are used for the runs in Table 1, so the coarse-grid sizes presented here are the same as those (and are presented only for convenient reference). As in Theorem 1, the important quantities in bounding the condition number of the overall preconditioner are $\lambda_M = \lambda_{\max}(D_{ff}^{-1}A_{ff})$ and $\lambda_m = \lambda_{\min}(D_{ff}^{-1}A_{ff})$, the upper and lower spectral equivalence bounds for the fine-scale operators. We also report the condition number of the preconditioned system, $\kappa(B^{-\frac{1}{2}}AB^{-\frac{1}{2}}) = \frac{\lambda_{\max}(B^{-1}A)}{\lambda_{\min}(B^{-1}A)}$, as an indicator of the overall success of the preconditioner.

Table 4 shows details of the performance of two-level ARMS-preconditioned CG, using the exact Schur complement, on these model PDE problems. As observed with AMGr, Algorithm 3 effectively reduces the dimension of the fine-scale problem and produces a diagonal matrix, D_{ff}, that has good spectral equivalence to the fine-scale block, A_{ff}. Setup and solve times scale poorly, as expected, due to the time required to first compute the exact Schur complement and then to invert it for each

TABLE 4
Performance of two-level ARMS using exact Schur complements on test matrices from discretizations of $-\nabla \cdot K(x,y)\nabla p(x,y)$.

Coefficient	Grid	n_c	λ_m	λ_M	t_{setup}	t_{solve}	# iters.	$\kappa(B^{-\frac{1}{2}}AB^{-\frac{1}{2}})$
$K(x,y)=1$	32×32	225	1.00	5.98	0.6	0.3	19	7.45
	64×64	961	1.00	5.99	10.7	6.5	18	7.48
	128×128	3969	1.00	6.00	199.1	366.0	18	7.49
$K(x,y)=10^{-8}+10(x^2+y^2)$	32×32	255	1.00	5.53	0.7	0.4	20	8.64
	64×64	1012	1.00	5.69	13.1	8.5	21	9.33
	128×128	4036	1.00	5.69	241.7	419.8	20	9.14
random $K(x,y)$	32×32	277	1.00	5.61	1.3	0.4	19	8.16
	64×64	1149	1.00	5.78	24.2	11.5	20	9.25
	128×128	4715	1.00	5.86	253.8	696.2	21	10.16

TABLE 5
Performance of two-level ARMS using variational approximations to the Schur complements on test matrices from discretizations of $-\nabla \cdot K(x,y)\nabla p(x,y)$.

Coefficient	Grid	n_c	ν_m	ν_M	t_{setup}	t_{solve}	# iters.	$\kappa(B^{-\frac{1}{2}}AB^{-\frac{1}{2}})$
$K(x,y)=1$	32×32	225	0.63	1.00	0.3	0.2	22	10.53
	64×64	961	0.63	1.00	4.4	0.7	22	10.62
	128×128	3969	0.63	1.00	71.8	4.0	22	10.64
$K(x,y)=10^{-8}+10(x^2+y^2)$	32×32	255	0.47	1.00	0.4	0.3	25	14.13
	64×64	1012	0.43	1.00	6.3	1.3	27	16.47
	128×128	4036	0.44	1.00	109.2	6.4	26	16.02
random $K(x,y)$	32×32	277	0.48	1.00	0.4	0.5	25	13.97
	64×64	1149	0.41	1.00	6.3	1.4	27	16.83
	128×128	4715	0.37	1.00	109.6	9.8	28	18.63

iteration. Iteration counts are stable as the problem size grows, and there is only slight growth in the condition numbers as the meshes are refined. The resulting condition numbers satisfy the bounds of Theorem 1, but these bounds are clearly not sharp. In particular, the upper bounds for these condition numbers are at least 100, while the largest observed condition number is just larger than 10. The lower bound, dependant only on the spectral equivalence of D_{ff} and A_{ff}, is a much better predictor of the true performance for these problems.

Table 5 shows details of the performance of two-level ARMS preconditioned CG, using variational approximations to the Schur complement, on these model PDE problems. The variational approximation is formed, as in AMGr, by taking the interpolation operator, $P = \begin{bmatrix} D_{ff}^{-1}A_{fc} \\ I \end{bmatrix}$, and then computing the approximation, $S = P^T AP$. As in Theorem 1, the spectral equivalence between S and $\hat{A}_{cc}$ plays an important role in the upper bound on the condition number of the preconditioned system. Here, we report the upper and lower spectral-equivalence bounds, ν_M and ν_m, along with the usual measures of performance. The fine-scale spectral-equivalence bounds, λ_M and λ_m, for these problems are the same as those appearing in Tables 1 and 4. The coarse-scale spectral-equivalence bounds for these problems are nicely controlled, with the upper bounds close to 1, and the lower bounds away from 0. Setup time again does not scale, as is expected from the increasing cost of computing the partitioning (without an efficient sorted data structure) and coarse-scale operator. Iteration times are very similar to those of two-level AMGr, as in Table 1, with a slightly larger factor of increase over that of the problem size. Iteration counts are steady, as are the condition numbers of the preconditioned operators. Once again, the bounds of Theorem 1 are loose bounds on the actual performance of the preconditioner; the lower bound does not take into account the role of the approximation to the Schur complement and, so, is unchanged from the case of an exact Schur complement, whereas the upper

bound increases slightly (by a factor of $\frac{1}{\nu_m}$) but remains significantly larger than the condition numbers that are observed.

4.2.1. PDE problems. To test the effectiveness of coarsening within ARMS using Algorithm 3 in a multilevel setting, we again consider PDE-based problems where the system matrix is symmetric, positive definite, and diagonally dominant (so that we are certain to satisfy all of the conditions of Theorem 1). As exact computation of the Schur complement on each level is prohibitively expensive, we first test the variational coarsening scheme used in Table 5. Here, GMRES is preconditioned by ARMS, and the number of iterations and time to solution needed to reduce the ℓ^2-norm of the residual by a relative factor of 10^6 are recorded. The time needed to set up the ARMS preconditioner is also reported, including time needed for choosing the coarse grids, computing the variational coarse-grid operators, and computing the exact Schur complement for the coarsest-grid operator, which is taken to be any operator which has dimension less than 10, or which is itself θ dominant. The parameter, θ, is again chosen to be 0.55. The ARMS preconditioner complexity (fill factor), c_B, defined to be the storage required for the ARMS preconditioner on all levels, divided by the number of nonzeros in the matrix A (and, thus, excluding the storage needed for A itself) is also recorded, as is the number of levels in the ARMS hierarchy.

Here, we use GMRES as the Krylov accelerator instead of CG for reasons of consistency. The ARMS algorithm has been developed with a focus on general matrices and not only symmetric problems [32, 30]. As such, GMRES has been the appropriate choice of accelerator. For these symmetric problems, using variational coarse-grid operators, we have chosen to use GMRES to remain consistent with this previous work. Additionally, while effort is made to retain some degree of symmetry in the ILU-based preconditioners to follow, symmetry may be sacrificed in favor of accuracy in the treatment of the coarsest levels (which may be quite large). In these cases, as well, GMRES is the only appropriate choice of Krylov subspace method. Thus, we have chosen to use it here, as well, for ease of comparison between these results. Nonsymmetric extensions of the approaches proposed here are considered (and compared to symmetric approaches) in [25].

Results for the multilevel-ARMS-preconditioned GMRES using variational coarse-grid operators and the greedy coarsening algorithm are shown in Table 6. Coarsening for the constant and smoothly varying cases are again the same, as they were in the multilevel AMGr results of Table 2. Counting the storage needed for the fine-scale operator itself, the operator complexities for all three problems are slightly higher than those of AMGr. Coupled with the added memory requirements of GMRES (storing one fine-scale vector per iteration), the 2048×2048 element mesh problems that were possible with AMGr are no longer tractable on a workstation with 2 Gigabytes of memory. Setup times are, in general, notably larger than AMGr for the same problem, with the only exception being the 1024×1024 mesh for the randomly discontinuous permeability where AMGr performed quite poorly. In all cases, iteration times for ARMS-preconditioned GMRES are larger than those for AMGr.

4.2.2. Combination with ILU-based coarsening. Theorem 1 also requires that $\left[\begin{smallmatrix} D_{ff} & -A_{fc} \\ -A_{cf} & A_{cc} \end{smallmatrix}\right]$ be nonnegative definite for the bounds on the condition number of the preconditioned system to hold. While this is true for all of the PDE problems tested in Table 6, it is not true for an anisotropic problem, such as when $K(x,y) = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 0.01 \end{smallmatrix}\right]$. Testing the variational-coarsening-based ARMS as a preconditioner for GMRES on a 32×32 element mesh for this problem requires 911 iterations

TABLE 6
Performance of multilevel ARMS with variational coarsening on test matrices from discretizations of $-\nabla \cdot K(x,y)\nabla p(x,y)$.

Coefficient	Grid	# levels	c_B	t_{setup}	t_{solve}	# iters.
$K(x,y) = 1$	128×128	6	0.69	0.1	0.3	42
	256×256	7	0.70	0.4	1.5	46
	512×512	8	0.70	1.7	8.0	50
	1024×1024	9	0.70	6.9	35.4	53
$K(x,y) = 10^{-8} + 10(x^2 + y^2)$	128×128	6	0.69	0.1	0.3	42
	256×256	7	0.70	0.4	1.5	46
	512×512	8	0.70	1.7	7.8	49
	1024×1024	9	0.70	6.9	34.2	52
random $K(x,y)$	128×128	6	0.89	0.1	0.3	46
	256×256	7	0.92	0.6	2.0	53
	512×512	8	0.95	2.7	12.2	64
	1024×1024	9	0.96	11.9	52.8	66

to reduce the residual by a relative factor of 10^6. To achieve the robustness we would like from our preconditioner, we turn to the full power of the ARMS approach [30], using an ILU factorization in place of the diagonal D_{ff}. Because we have partitioned A specifically so that the A_{ff}-block is diagonally dominant, we expect to be able to find a suitably sparse ILU factorization that provides both better spectral equivalence to A_{ff} than is possible with a diagonal matrix and a better approximate Schur complement. This combination should then result in a more robust preconditioner than is possible with a diagonal D_{ff}.

Since the fine-scale operators from these PDE problems are symmetric, certain savings are natural in the coarse-grid selection algorithm. In particular, when a coarse-grid node, j, is selected from the set, U, we need only look at the nonzero elements in row j of the matrix to find the nodes whose measures need updating. This is particularly easy when using a compressed-sparse-row storage format, as these indices and the appropriate updates are easily available. In the general (nonsymmetric) case, the update is not as convenient (when using compressed sparse row storage), as a search for nonzero entries in column j must be done, requiring either extra storage or extra computation. To ensure this advantage is maintained, a modification of the ILUT procedure is used in order to ensure that a symmetric dropping strategy is employed. Here, the ILU factorization of A_{ff} is computed using the ILUT algorithm (dropping based both on size and rowwise fill factors [31]), but the U factor in $A_{ff} \approx LDU$ is replaced by L^T. The matrix product $L^{-1}A_{fc}$ is easily computed at the same time as the factorization of A_{ff}, by applying the elimination to the extended matrix, $[A_{ff},\ -A_{fc}]$. The Schur complement can then be computed rowwise from the matrix equation, $S = A_{fc} - (L^{-1}A_{fc})^T D^{-1}(L^{-1}A_{fc})$. In order to control sparsity directly within S, we precompute the diagonal entries of S, allowing dropping based on size, using the symmetric drop criterion, $s_{ij} \leq \alpha\sqrt{s_{ii}s_{jj}}$. Dropping based on level of fill is also possible [22] but not used in the examples presented here.

Table 7 shows the results for the ARMS-preconditioned GMRES algorithm based on symmetrized ILUs. The drop tolerance, α, is chosen to be either 0.01 or 0.1 for both the ILU factorization of A_{ff} and the approximate Schur complement calculation. Choosing α appropriately requires a balance between accuracy of the ILU approximation to A_{ff} and the fill of the resulting preconditioner. Typically, choosing α to be relatively small results in a more effective preconditioner, but a larger value of α results in a smaller preconditioner complexity, c_B, which is useful for large problems

TABLE 7

*Performance of multilevel ARMS with symmetric ILU-based coarsening on test matrices from discretizations of $-\nabla \cdot K(x,y)\nabla p(x,y)$. Results marked with a * indicate that the residual reduction criterion was a relative factor of 10^4 instead of 10^6.*

Coefficient	Grid	α	# levels	c_B	t_{setup}	t_{solve}	# iters.
$K(x,y) = 1$	128×128	0.01	2	2.59	0.3	0.3	28
	256×256	0.01	2	2.65	1.5	2.5	44
	512×512	0.01	2	2.68	12.7	24.5	82
	1024×1024	0.1	2	1.03	159.1	34.2*	46*
$K(x,y) = 10^{-8} + 10(x^2 + y^2)$	128×128	0.01	2	2.60	0.3	0.4	31
	256×256	0.01	2	2.65	1.5	3.4	56
	512×512	0.01	2	2.68	12.7	31.7	97
	1024×1024	0.1	2	1.03	159.6	40.6*	52*
random $K(x,y)$	128×128	0.01	3	1.40	0.2	0.4	32
	256×256	0.01	3	1.41	0.7	2.5	45
	512×512	0.01	3	1.42	3.1	25.1	83
	1024×1024	0.01	3	1.42	13.5	12.4*	17*
anisotropic $K(x,y)$	128×128	0.01	5	1.61	0.2	0.3	26
	256×256	0.01	5	1.62	0.8	2.3	42
	512×512	0.01	5	1.63	3.3	17.3	65
	1024×1024	0.01	5	1.63	14.9	6.9*	10*

where memory limitations are restrictive. A further dropping rule is applied to each row of L, allowing no more than twice the average number of nonzeros in a row of A. Again, $\theta = 0.55$, and the coarsening is continued until either the coarse-grid operator has fewer than 10 degrees of freedom or is itself θ dominant.

The robustness of the ILU-based approach comes at a cost in increased memory consumption, reflected in the higher operator complexities for the results in Table 7, as compared to those in Table 6. For the smooth isotropic coefficients, only one coarsening was performed, because of the resulting θ-dominance of the coarse-grid operator. This behavior is largely independent of the choices of α and the row-based fill threshold, only changing if θ was measurably changed. As a result, the operator complexities for these problems are somewhat large, over 2 when $\alpha = 0.01$. In order to perform any meaningful number of iterations on the 1024×1024 element mesh problems, it is necessary to choose $\alpha = 0.1$ to reduce the fill. Even then, memory limitations prevented reducing the residual by a relative factor of 10^6, and a smaller reduction, by a relative factor of 10^4 is used instead. GMRES performance (residual reduction per iteration) for these problems does tend to get worse as the iterations proceed and, so, the number of iterations recorded for these large problems is significantly fewer than the number that would be needed for a reduction by a relative factor of 10^6.

Comparing performance between these four problems, however, shows the robustness of the combination of the greedy coarsening algorithm with ILU-based ARMS, which will be further demonstrated in section 4.2.3. Preconditioner complexities, a primary factor in memory limitations, are controlled by the parameter, α. Setup times are generally comparable across the different problems (for the same α), as are solution times and iteration counts. In fact, the problems which caused most difficulty for the variational coarse-grid operator approach are now more efficient than the smoothly varying coefficient problems that were most efficient. The structure of these problems appears to naturally lead to deeper ARMS hierarchies, yielding lower preconditioner complexities and, thus, lower costs per iteration.

Comparing the results for AMG (Table 3) to those for ARMS (Table 7), we see that both generally perform well for these PDE-based problems, but that the AMG approach appears to be more efficient. ARMS requires storage of part of the

TABLE 8
Performance of multilevel ARMS with symmetric ILU-based coarsening on test matrices with $\theta = 0.55$, drop tolerance of 0.01, average fill factor of 5, and a maximum of 10 levels of coarsening.

Name	n	nnz	c_B	t_{setup}	t_{solve}	# iters.
bodyy4	17546	121938	1.08	0.1	0.05	6
bodyy5	18589	129281	1.06	0.1	0.04	4
bodyy6	19366	134748	1.04	0.1	0.03	3
bcsstk18	11948	149090	1.17	0.1	2.7	189
t2dah_a	11445	176117	1.23	0.2	0.9	88
obstclae	40000	197608	2.41	0.3	0.09	5
jnlbrng1	40000	199200	1.85	0.3	0.1	7
minsurfo	40806	203622	2.02	0.3	0.1	6
bcsstk25	15439	252241	1.26	0.3	3.7	190
cvxbqp1	50000	349968	6.51	7.2	1111.2	2670
bcsstk17	10974	428650	2.30	1.2	24.7	660
t3dl_a	20360	509866	1.48	0.5	20.8	465
gridgena	48962	512084	1.32	0.5	2.7	76
finan512	74752	596992	1.32	0.6	0.2	4
gyro_k	17361	1021159	1.13	1.1	19.9	397
msc10848	10848	1229778	1.54	2.9	22.0	431
cfd1	70656	1828364	3.33	10.5	391.3	1003
vanbody	47072	2336898	7.55	244.7	1131.8	2415
oilpan	73752	3597188	1.48	6.2	862.7	1582

operator on each level, plus the ILU factors of each fine-grid block, while AMG requires storage of only the operators, as relaxation on each level is determined entirely by these operators. The ability of AMG to run as an iteration by itself, instead of as a preconditioner for a Krylov subspace method, also reduces its memory overhead, although the ARMS preconditioners used here could be modified for use with CG instead of GMRES, ameliorating this effect. Overall solve times and iteration counts are also much lower for the AMG approach. This is not surprising, as multigrid techniques are largely focused on solving PDE-based problems such as these. The ARMS approach, on the other hand, is motivated by purely algebraic consideration, and, so, we expect it to not perform as well as AMG on PDE-based problems, but to exhibit robustness that the AMG approach may not have.

4.2.3. General sparse symmetric systems. To further test the robustness of the ARMS approach, we consider a subset of the matrices from the test set of [18, 19]. As we are interested primarily in matrices that are not naturally diagonally dominant, we consider only those symmetric and positive-definite matrices for which full data is available (i.e., we do not consider matrices for which only a nonzero pattern is given). In order to run all tests on a single machine (dual-processor Intel Xeon 3.0 GHz with 2 GBytes of RAM), we test all problems with fewer than 3 million nonzero entries. By using swap space, convergence results for one problem with approximately 3.5 million nonzero entries, `oilpan`, are also obtained.

Results in Table 8 show the performance of ARMS-preconditioned GMRES for these matrices. As a first test, we take the parameters for ARMS to allow significant fill in the preconditioner, B, thus looking at the case where accuracy in computing B is not a significant factor in ARMS performance. Diagonal-dominance parameter θ was chosen to be 0.55, as in the PDE-based problems above. The ILU drop tolerance, α, was taken to be 0.01, and further dropping is done so that no row of the approximate L factor has more than 5 times the average number of nonzeros per row of A_{ff} in it. A maximum of 10 levels is allowed, with the coarsest-grid operator computed using

ILUTP with $\alpha = 0.0001$ and fill per row restricted to 20 times the average number of nonzeros per row of the coarsest-grid operator. ILUTP is a pivoting variant of the ILUT algorithm discussed above [31]; elements in the elimination are again filtered by size and fill per row, but a column pivoting step is added for each row to improve the stability of the factorization. With these parameters, ARMS failed to reduce the norm of the residual by a relative factor of 10^6 before memory limitations were reached for only three of the twenty-two problems.

Table 8 reports the complexities (fill factors), c_B, of the resulting preconditioners, setup and iteration times (in seconds), and number of iterations needed to reduce the norm of the residual by a relative factor of 10^6 for the nineteen problems that did converge. As these problems arise from many different applications, it is difficult to make comparisons across different problems, in particular, between smaller and larger problems. We do see quick convergence with low preconditioner complexities for a number of problems, however. For the problems with slower convergence, those with low preconditioner complexities allow many iterations within a short period of CPU time, making the overall time to solution acceptable. For a few of the problems with preconditioner complexities greater than 3, solution was still possible before running into memory limitations. The three problems for which convergence was not obtained, `bcsstk36` ($nnz = 1143140$, $c_B = 10.63$), `msc23052` ($nnz = 1154814$, $c_B = 9.83$), and `nasasrb` ($nnz = 2677324$, $c_B = 4.80$), the large preconditioner complexities coupled with the large initial problem sizes required termination of the iterations before convergence. Slow but steady convergence was seen for `msc23052` and `nasasrb` before iterations were stopped. Only iteration on `bcsstk36` was ineffective in reducing the residual substantially in 6100 iterations before memory limitations were reached.

Experiments with larger drop tolerances (e.g., $\alpha = 0.05$ or $\alpha = 0.1$) show good decreases in preconditioner complexity, but poor performance of the resulting preconditioners. Thus, in Table 9, we look to reduce the preconditioner complexities by allowing each row of L to have only twice as many nonzeros as an average row of A_{ff}, by allowing more levels (up to 50), and by using a smaller $\theta = 0.51$ in the resulting preconditioners. For some problems, these results are equivalent to those in Table 8, up to measurement errors in the times reported. For the problems with large preconditioner complexities in Table 8, the preconditioner complexities are now lower (although still possibly large). This reduction of complexity, however, is (as expected) correlated with an increase in iteration counts and overall time-to-solution (although this is not always the case; cf. `vanbody` and `cvxbqp1`). Setup and solution times are comparable for the problems with fewer than one million nonzeros, although, in some cases, notable growth in c_B and setup time are also seen. For the larger problems, results are slightly mixed. A few more iterations are needed for `cfd1` and `oilpan`, resulting in slightly higher solution times, whereas a much lower preconditioner complexity results in lower setup and solution times for `vanbody`. The lower preconditioner complexities for `bcsstk36` and `msc23052` result in sufficient residual reduction to meet the convergence criteria before available memory (including swap space) is filled. A slightly lower preconditioner complexity for `nasasrb`, 4.36, was obtained for these parameters, but memory considerations still forced early termination of the run, with a residual reduction by a relative factor of approximately 3.35×10^4 in 2500 iterations.

4.2.4. A comparison with AMG and ILUTP. As discussed in section 4.2.2, AMG is expected to perform better than ARMS in some cases (particularly for ma-

TABLE 9
Performance of multilevel ARMS with symmetric ILU-based coarsening on test matrices with $\theta = 0.51$, drop tolerance of 0.01, average fill factor of 2, and a maximum of 50 levels of coarsening.

Name	n	nnz	c_B	t_{setup}	t_{solve}	# iters.
bodyy4	17546	121938	1.01	0.06	0.07	6
bodyy5	18589	129281	0.99	0.09	0.05	4
bodyy6	19366	134748	0.98	0.1	0.04	3
bcsstk18	11948	149090	1.16	0.1	2.1	165
t2dah_a	11445	176117	1.19	0.2	0.9	81
obstclae	40000	197608	8.40	3.9	0.1	3
jnlbrng1	40000	199200	5.10	1.3	0.1	4
minsurfo	40806	203622	3.29	0.4	0.1	6
bcsstk25	15439	252241	1.24	0.3	3.1	162
cvxbqp1	50000	349968	4.02	1.8	642.0	2018
bcsstk17	10974	428650	1.90	0.9	27.4	705
t3dl_a	20360	509866	1.33	0.5	25.5	527
gridgena	48962	512084	1.32	0.5	2.9	81
finan512	74752	596992	1.33	0.5	0.2	5
gyro_k	17361	1021159	1.12	1.0	19.9	398
bcsstk36	23052	1143140	5.79	21.6	2301.5	5648
msc23052	23052	1154814	5.72	22.4	2403.4	5754
msc10848	10848	1229778	1.37	2.3	21.9	433
cfd1	70656	1828364	2.99	8.2	515.6	1198
vanbody	47072	2336898	3.51	19.6	460.8	1413
oilpan	73752	3597188	1.33	4.6	979.1	1694

trices arising from discretization of elliptic PDEs), but ARMS is generally expected to be more robust. That is, we expect that the loss in efficiency of ARMS for certain problems will be offset by more consistent performance on a general collection of problems, such as those in Tables 8 and 9. To test this expectation, Table 10 gives the performance of classical AMG on the general problems of the previous section. Here, the two-stage coarsening algorithm using greedy coarsening as a first pass with $\theta = 0.55$ and strength threshold $\beta = 0.3$ is used to select the coarse grid and interpolation stencil, while the classical (Ruge–Stüben) algorithm is used to define the interpolation weights. A single full-grid sweep of Gauss–Seidel relaxation is used on each level descending and ascending the V-cycle. To ensure the multigrid hierarchy fits in available memory, 10 sweeps of relaxation are used to "solve" the coarsest-grid problem, instead of computing the (sometimes large, dense) LU factorization of the coarsest-grid operator.

Comparing the AMG and ARMS results for these problems, we see that AMG results in a faster overall time to solution for only five problems. For problems `obstclae`, `jnlbrng1`, and `minsurfo`, the setup time for AMG is one-third faster and solve times the same as those in Table 8. For these three problems, the AMG operator complexities are also close to the ARMS preconditioner complexities, reflecting similar overall memory costs, especially as the AMG operator complexity includes the storage for the original fine-scale operator, not counted in the ARMS fill factors. AMG outperforms ARMS on matrix `cvxbqp1`, although this really reflects better, but still bad, performance. The low AMG operator complexity for this problem allows AMG to converge in less time than ARMS but, with over ten thousand iterations required to reduce the residual by a relative factor of 10^6, AMG performance can hardly be called acceptable. Finally, the reduced storage costs of AMG compared to ARMS-preconditioned GMRES allows AMG to converge on the `nasasrb` matrix, although with a large operator complexity and many iterations.

TABLE 10

Performance of multilevel AMG V(1,1) cycles on test matrices with full-grid Gauss–Seidel relaxation, two-stage greedy coarsening using $\theta = 0.55$ and $\beta = 0.3$, a maximum of 20 levels of coarsening, with the coarsest-grid problem solved by 10 sweeps of relaxation.

Name	n	nnz	# levels	c_G	c_A	t_{setup}	t_{solve}	# iters.
bodyy4	17546	121938	8	1.50	1.44	0.03	2.8	363
bodyy5	18589	129281	8	1.57	1.51	0.04	15.8	1851
bodyy6	19366	134748	7	1.61	1.55	0.04	74.7	8385
bcsstk18	11948	149090	20	2.49	5.51	0.2	11.0	497
t2dah_a	11445	176117	20	5.11	6.89	0.4	324.3	10348
obstclae	40000	197608	11	1.73	2.47	0.2	0.08	3
jnlbrng1	40000	199200	13	2.10	3.45	0.2	0.1	4
minsurfo	40806	203622	13	1.90	2.93	0.2	0.1	4
bcsstk25	15439	252241	20	3.42	9.07	0.6	9.4	168
cvxbqp1	50000	349968	4	1.81	1.38	0.2	287.7	10694
bcsstk17	10974	428650	20	4.57	13.46	1.4	525.9	4290
gridgena	48962	512084	16	1.80	3.13	0.6	13.9	274
t3dl_a	20360	509866	20	3.69	6.31	0.8	587.9	7333
finan512	74752	596992	12	1.74	2.50	0.4	0.4	6
gyro_k	17361	1021159	20	4.08	8.17	2.4	172.3	946
bcsstk36	23052	1143140	20	6.87	26.03	7.1	2828.0	4486
msc23052	23052	1154814	20	6.95	28.00	8.5	2312.4	3357
msc10848	10848	1229778	20	8.07	20.05	6.0	10375.7	19829
cfd1	70656	1828364	20	4.52	80.16	52.1	4318.7	1426
vanbody	47072	2336898	20	6.15	21.87	13.5	2846.3	2526
nasasrb	54870	2677324	20	4.27	13.02	9.6	3586.4	4711
oilpan	73752	3597188	20	2.80	3.66	3.3	1314.2	3938

For the remaining problems, AMG is generally worse in terms of both storage factors (or cost per iteration) and iteration count. The operator complexities for AMG are often larger than ten times the storage of the fine-grid operator, and as much as eighty times this storage is needed. Iteration counts are over ten thousand for three problems and are often many times those used for ARMS. Put simply, there is no reason to expect that the assumptions made in classical AMG on the slow-to-converge modes of Gauss–Seidel relaxation are valid for these general problems and, as a result, the AMG interpolation scheme may provide an arbitrarily bad complement to this relaxation. While the parameters of the AMG methodology may be tuned to improve performance on any one of these problems, the results in Table 10 clearly show the lack of robustness of AMG when used as a black-box solver, in comparison to the relatively good results seen in Tables 8 and 9.

For comparison with a single-level preconditioning strategy, we also look at the performance of (nonsymmetrized) ILUTP on these same test problems in Table 11. In order to make fair comparisons between the multilevel ILU strategy in ARMS and that in ILUTP, we choose the parameters of ILUTP, notably α, the drop tolerance, such that the overall preconditioner complexity is roughly the same as those in Table 9. We limit the number of nonzero entries in each row of L and U^T to be no more than 20 times the average number of nonzeros per row of A. This is done so that the choice of α is used as the primary control over the preconditioner complexities; α is varied in increments of 0.0025 to give the operator complexity closest to those of Table 9. We also report the preconditioner complexities, setup and solve times, and number of iterations to reduce the residual by a relative factor of 10^6. For each problem, we allow at most twice the number of iterations that ARMS-preconditioned GMRES took to converge in Table 9, unless that would require too much memory (i.e., for the problems where swapping was required), in which case the limit is the number of iterations that

TABLE 11
Performance of ILUTP-preconditioner GMRES on test problems.

Name	n	nnz	α	c_B	t_{setup}	t_{solve}	# iters.	Reduction
bodyy4	17546	121938	0.03	1.00	0.5	0.03	6	-
bodyy5	18589	129281	0.0175	1.00	0.6	0.04	7	-
bodyy6	19366	134748	0.0125	0.97	0.7	0.03	7	-
bcsstk18	11948	149090	0.03	1.17	0.1	4.5	330	1.79×10^4
t2dah_a	11445	176117	0.1975	1.18	0.2	1.5	162	1.24×10^1
obstclae	40000	197608	0.0025	4.55	3.4	0.09	4	-
jnlbrng1	40000	199200	0.0025	4.39	3.5	0.1	6	-
minsurfo	40806	203622	0.0025	3.54	3.4	0.09	5	-
bcsstk25	15439	252241	0.04	1.21	0.2	3.2	216	-
cvxbqp1	50000	349968	0.0225	4.14	4.4	594.5	2018	2.29×10^2
bcsstk17	10974	428650	0.125	1.91	0.8	70.4	1410	5.32×10^2
t3dl_a	20360	509866	0.03	1.36	0.9	7.4	275	-
gridgena	48962	512084	0.025	1.32	1.5	4.0	118	-
finan512	74752	596992	0.0075	1.34	2.4	0.1	4	-
gyro_k	17361	1021159	0.0725	1.16	0.9	44.4	796	1.04×10^5
bcsstk36	23052	1143140	0.21	5.70	54.0	2111.8	5648	3.67
msc23052	23052	1154814	0.18	5.70	82.3	2172.8	5754	7.54
msc10848	10848	1229778	0.05	1.38	2.4	9.0	244	-
cfd1	70656	1828364	0.0175	2.97	12.0	44.9	297	-
vanbody	47072	2336898	0.3025	3.60	12.9	1266.4	2826	9.09×10^1
oilpan	73752	3597188	0.2	1.25	11.0	908.4	1694	3.58

ARMS-preconditioned GMRES took to converge. The residual reduction factors for the problems that did not converge are reported in the final column of Table 11.

Comparing the results from Tables 9 and 11, we see that the ARMS-preconditioning approach often, but not always, pays off relative to ILUTP-preconditioned GMRES. In terms of total time to solution, the two strategies tie for a single problem (`bcsstk25`), the ARMS-preconditioners are faster for sixteen problems, while ILUTP yields faster solvers for four problems. In terms of number of iterations (closely related to iteration time as the preconditioner complexities are approximately equal for each problem), ILUTP-preconditioned GMRES required fewer iterations for five problems, with one tie and the ARMS preconditioner requiring fewer iterations for fifteen problems. Of these fifteen problems, ILUTP converged before iteration and memory limits were reached for only six problems. For the remaining nine problems, ARMS was significantly better than ILUTP. On three of the test matrices, not only did ILUTP not converge in double the number of iterations that ARMS required, but it did not succeed in reducing the norm of the residual by a factor of even half that required for convergence. On the two most difficult problems for which ARMS was successful, `bcsstk36` and `msc23052`, ILUTP was unable to reduce the residual by a factor of 10 in the same number of iterations that ARMS required for a reduction by a relative factor of 10^6. For the `nasasrb` matrix that caused ARMS the most difficulty, ILUTP reduced the residual by a factor of 2.02 in the same number of iterations it took ARMS to reduce the residual by a factor of 3.4×10^4.

5. Conclusions. Motivated by a theoretical analysis, a new coarsening algorithm for algebraic multigrid and multilevel solvers is proposed. This greedy algorithm partitions the degrees of freedom so that the fine-grid block satisfies some diagonal dominance properties, and, thus, can be easily treated by relaxation or an ILU factorization. Numerical results show that two-level solvers based on this technique exhibit optimal convergence properties. Multilevel solvers based on simple grid-transfer oper-

ators perform well for simple model problems but break down when certain theoretical assumptions are violated. When used in combination with coarsening ideas from the usual AMG and ARMS methodologies, the resulting algorithms achieve improved robustness, demonstrated on both PDE-based problems and test problems from a variety of sources.

Acknowledgment. The authors would like to thank the anonymous referees for their comments that helped to improve the quality of this paper.

REFERENCES

[1] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, Cambridge, UK, 1994.
[2] N. S. Bakhvalov, *On the convergence of a relaxation method under natural constraints on an elliptic operator*, Z. Vycisl. Mat. i. Mat. Fiz., 6 (1966), pp. 861–883.
[3] R. E. Bank, *Hierarchical bases and the finite element method*, in Acta Numerica, Acta Numer. 5, Cambridge University Press, Cambridge, UK, 1996, pp. 1–43.
[4] A. Brandt, S. F. McCormick, and J. W. Ruge, *Algebraic multigrid (AMG) for sparse matrix equations*, in Sparsity and Its Applications, D. J. Evans, ed., Cambridge University Press, Cambridge, UK, 1984.
[5] A. Brandt, *Algebraic multigrid theory: The symmetric case*, Appl. Math. Comput., 19 (1986), pp. 23–56.
[6] A. Brandt, *General highly accurate algebraic coarsening*, Electron. Trans. Numer. Anal., 10 (2000), pp. 1–20.
[7] J. Brannick, M. Brezina, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *An energy-based AMG coarsening strategy*, Numer. Linear Algebra Appl., 13 (2006), pp. 133–148.
[8] J. Brannick and R. Falgout, *Compatible relaxation and coarsening in algebraic multigrid*, in preparation.
[9] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *Adaptive algebraic multigrid*, SIAM J. Sci. Comp., 27 (2006), pp. 1261–1286.
[10] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed., SIAM, Philadelphia, 2000.
[11] O. Bröker, *Parallel Multigrid Methods using Sparse Approximate Inverses*, Ph.D. thesis, Swiss Federal Institute of Technology Zurich, Zurich, Switzerland, 2003.
[12] 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 J. Sci. Comput., 21 (2000), pp. 1886–1908.
[13] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, The MIT Electrical Engineering and Computer Science Series, MIT Press, Cambridge, MA, 1990.
[14] H. De Sterck, U. M. Yang, and J. J. Heys, *Reducing complexity in parallel algebraic multigrid preconditioners*, SIAM J. Matrix Anal. Appl., 27 (2006), pp. 1019–1039. Also available as LLNL technical report UCRL-JRNL-206780.
[15] R. D. Falgout and P. S. Vassilevski, *On generalizing the alebraic multigrid framework*, SIAM J. Numer. Anal., 42 (2004), pp. 1669–1693. Also available as LLNL technical report UCRL-JC-150807.
[16] R. P. Fedorenko, *On the speed of convergence of an iteration process*, Z. Vycisl. Mat. i. Mat. Fiz., 4 (1964), pp. 559–564 (in Russian).
[17] R. P. Fedorenko, *On the speed of convergence of an iteration process*, U.S.S.R. Comput. Math. and Math. Phys., 4 (1964), pp. 227–235.
[18] N. I. M. Gould and J. A. Scott, *Complete results for a numerical evaluation of HSL packages for the direct solution of large sparse, symmetric linear systems of equations*, Numerical Analysis Internal Report 2003-2, Rutherford Appleton Laboratory, Chilton, Didcot Oxfordshire, UK, 2003.
[19] N. I. M. Gould and J. A. Scott, *A numerical evaluation of HSL packages for the direct solution of large sparse, symmetric linear systems of equations*, ACM Trans. Math. Software, 30 (2004), pp. 300–325.
[20] W. Hackbusch, *Multigrid Methods and Applications*, Springer Ser. Comput. Math. 4, Springer-Verlag, Berlin, 1985.
[21] V. E. Henson and U. M. Yang, *BoomerAMG: A parallel algebraic multigrid solver and preconditioner*, Appl. Numer. Math., 41 (2002), pp. 155–177.

[22] L. Little, Y. Saad, and L. Smoch, *Block LU preconditioners for symmetric and nonsymmetric saddle point problems*, SIAM J. Sci. Comput., 25 (2003), pp. 729–748.
[23] O. Livne, *Coarsening by compatible relaxation*, Numer. Linear Algebra Appl., 11 (2004), pp. 205–227.
[24] S. MacLachlan, T. Manteuffel, and S. McCormick, *Adaptive reduction-based AMG*, Numer. Linear Algebra Appl., 13 (2006), pp. 599–620.
[25] S. MacLachlan and Y. Saad, *Greedy coarsening strategies for non-symmetric problems*, Tech. rep. umsi-2006-58, Minnesota Supercomputing Institute, University of Minnesota, Minneapolis, MN, 2006.
[26] S. F. McCormick and J. W. Ruge, *Multigrid methods for variational problems*, SIAM J. Numer. Anal., 19 (1982), pp. 924–929.
[27] Y. Notay, *Using approximate inverses in algebraic multilevel methods*, Numer. Math., 80 (1998), pp. 397–417.
[28] Y. Notay, *Algebraic multigrid and algebraic multilevel methods: A theoretical comparison*, Numer. Linear Algebra Appl., 12 (2005), pp. 419–451.
[29] J. W. Ruge and K. Stüben, *Algebraic multigrid (AMG)*, in Multigrid Methods, S. F. McCormick, ed., Frontiers Appl. Math. 3, SIAM, Philadelphia, 1987, pp. 73–130.
[30] Y. Saad and B. Suchomel, *ARMS: An algebraic recursive multilevel solver for general sparse linear systems*, Numer. Linear Algebra Appl., 9 (2002), pp. 359–378.
[31] Y. Saad, *ILUT: A dual threshold incomplete ILU factorization*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.
[32] Y. Saad, *Multilevel ILU with reorderings for diagonal dominance*, SIAM J. Sci. Comput., 27 (2005), pp. 1032–1057.
[33] R. Southwell, *Stress calculation in frameworks by a method of systematic relaxation of constraints.* I, II, Proc. Roy. Soc. London. Ser. A., 151 (1935), pp. 56–95.
[34] K. Stüben, *An introduction to algebraic multigrid*, in Multigrid, U. Trottenberg, C. Oosterlee, and A. Schüller, Academic Press, San Diego, CA, 2001, pp. 413–528.
[35] M. Tismenetsky, *A new preconditioning technique for solving large sparse linear systems*, Linear Algebra Appl., 154–156 (1991), pp. 331–353.
[36] W. L. Wan, T. F. Chan, and B. Smith, *An energy-minimizing basis interpolation for robust multigrid methods*, SIAM J. Sci. Comput., 21 (2000), pp. 1632–1649.
[37] J. Xu and L. T. Zikatanov, *On an energy minimizing basis in algebraic multigrid methods*, Computing and Visualization in Science, 7 (2004), pp. 121–127.
[38] D. Zuckerman, *On unapproximable versions of NP-complete problems*, SIAM J. Comput., 25 (1996), pp. 1293–1304.