ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/220411616

Adaptive Algebraic Multigrid

Article *in* SIAM Journal on Scientific Computing · January 2006

DOI: 10.1137/040614402 · Source: DBLP

CITATIONS
130

READS
200

6 authors, including:

Marian Brezina
University of Colorado Boulder
42 PUBLICATIONS **3,060** CITATIONS

SEE PROFILE

Steve F McCormick
University of Colorado Boulder
239 PUBLICATIONS **7,758** CITATIONS

SEE PROFILE

John Ruge
University of Colorado Boulder
85 PUBLICATIONS **3,344** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Approximate Ideal Restriction in AMG View project

Nonsymmetric AMG View project

All content following this page was uploaded by Marian Brezina on 07 July 2014.

The user has requested enhancement of the downloaded file.

ADAPTIVE ALGEBRAIC MULTIGRID*

M. BREZINA† R. FALGOUT‡ S. MACLACHLAN†
T. MANTEUFFEL† S. MCCORMICK† AND J. RUGE†

Abstract. Efficient numerical simulation of physical processes is constrained by our ability to solve the resulting linear systems, prompting substantial research into the development of multiscale iterative methods capable of solving these linear systems with an optimal amount of effort. Overcoming the limitations of geometric multigrid methods to simple geometries and differential equations, algebraic multigrid methods construct the multigrid hierarchy based only on the given matrix. While this allows for efficient black-box solution of the linear systems associated with discretizations of many elliptic differential equations, it also results in a lack of robustness due to unsatisfied assumptions made on the near null spaces of these matrices. This paper introduces an extension to algebraic multigrid methods that removes the need to make such assumptions by utilizing an adaptive process. Emphasis is on the principles that guide the adaptivity and their application to algebraic multigrid solution of certain symmetric positive-definite linear systems.

1. Introduction. While the original development of algebraic multigrid (AMG) began over twenty years ago (cf. [8, 24]), the current level of interest and research activity is fairly recent. This dramatic increase is largely due to the potential of these methods for solving very large problems arising from partial differential equations with irregular grids and varying coefficients. See, for example, [1, 3, 13, 28, 33].

By the term algebraic multigrid, we mean the class of solvers based on multigrid principles that depend little or not at all on geometric information about the problem, but instead attempt to use basic concepts of "algebraic smoothness" to determine effective coarsening and/or relaxation processes. Solvers of this type typically assume some defining characteristic of algebraic smoothness, specifying error components that are not quickly eliminated by the relaxation that is being used. For example, all such components are assumed, in standard AMG (cf. [27] and §2), to vary slowly along so-called strong connections in the matrix, or, in smoothed aggregation (SA; cf. [30]), to be represented locally by a few prototype vectors supplied by the user. While appropriate use of the characteristic of algebraic smoothness seems essential for obtaining effective solvers, these additional assumptions limit the scope of applicability of such methods. In many important cases, errors missed by standard relaxation processes can vary substantially along strong matrix connections, and, in many cases, even the concept of strength of connection is not well understood. Moreover, supplying a fully representative set of prototypical smooth components is not always easy nor possible in practice.

The principal aim of the adaptive approach developed here is to eliminate, or substantially reduce, this reliance on the additional assumptions usually present in these methods. The basic idea is to test the initial version of the given solver on the homogeneous problem, $A\mathbf{x} = \mathbf{0}$, to determine its performance and expose whatever

*Submitted to the SIAM Journal on Scientific Computing, September 7, 2004.

†Department of Applied Mathematics, Campus Box 526, University of Colorado at Boulder, Boulder, CO, 80309–0526. *email*: mbrezina@math.cudenver.edu, maclachl@colorado.edu, tmanteuf@colorado.edu, stevem@colorado.edu, and jruge@colorado.edu. This work was sponsored by the Department of Energy under grant numbers DE-FC02-01ER25479 and DE-FG02-03ER25574, Lawrence Livermore National Laboratory under contract number B533502, Sandia National Laboratory under contract number 15268, and the National Science Foundation under VIGRE grant number DMS-9810751.

‡Center for Applied Scientific Computation, Lawrence Livermore National Lab, Post Office Box 808, Livermore, CA, 94551. *email*: rfalgout@llnl.gov.

types of errors it does not effectively reduce. The resulting prototypical errors that these tests produce are then used to improve the algebraic multigrid process.

The concept of using a multigrid algorithm to improve itself is not new. Using prototypical algebraically smooth vectors in the coarsening process was first developed in the early stages of the AMG project of Brandt, McCormick, and Ruge (documented in [24]), where interpolation was defined to fit vectors obtained by relaxation on the homogeneous problem. In [8], a variation of this idea was used for recovering typical AMG convergence rates for a badly scaled scalar elliptic problem. While this initial approach was very basic and used only one prototype of algebraically smooth error, it contained many of the ingredients of the adaptive process described here. These concepts were developed further in [23, 25, 26, 27]. The concept of fitting eigenvectors corresponding to the smallest eigenvalues was advocated in [23] and [27], where an AMG algorithm determining these eigenvectors through Rayleigh quotient minimization was outlined. These vectors were, in turn, used to update the AMG interpolation and coarse-level operators. Adaptivity of linear solvers has a long history, including adaptive choice of the overrelaxation parameter in SOR [18] or the parameters in a Chebyshev iteration [21], and has seen renewed interest of late as a way to improve robustness of existing linear solvers, as in the work of Brandt [7] and Wagner and Wittum [31, 32].

In many ways, using algebraically smooth vectors in the definition of interpolation represents the next logical step in improving the interpolation operators used in robust geometric and algebraic multigrid methods. Alcouffe et al. introduced the concept of operator-induced interpolation in [2]. This improvement on the previously used geometric interpolation approach opened up a much larger class of problems to black-box multigrid solution.

In the present paper, we use an operator-induced interpolation approach as well, but also rely on an automatic process that supplies representative algebraically smooth components to ensure optimal performance. By integrating information regarding algebraically smooth vectors into the definition of interpolation, we develop multigrid schemes that are hopefully optimal for elliptic problems where the discrete system is not necessarily in the form of an M-matrix. These operator- and relaxation-induced interpolation approaches can, if properly implemented, greatly enlarge the class of problems that admits optimal performance by a black-box multigrid technique.

We also introduce this form of adaptivity into AMG. While classical multigrid methods can be viewed as stationary iterative methods [4], the methods presented here are dynamic in their setup phase. In fact, we propose using the method itself to drive its own iterative improvement. A "bootstrap" AMG method that is similar to the approach developed here was recently proposed for the classical AMG setting by Brandt [7, 9].

Several other attempts have been made to allow for the solver itself to determine from the discrete problem the information required to solve it successfully, without a priori assumptions on the form of the smooth error, including the methods of [10, 12, 14, 17]. All of these methods, however, have in common their requirement that the local finite element matrices of the problem be available, and they construct the multigrid transfer operators based on the algebraically smooth eigenvectors corresponding to local stiffness matrices assembled over element agglomerates. Although they can achieve encouraging convergence rates, their need to construct, store, and manipulate the coarse-level element information typically leads to increased storage requirements compared to those of classical AMG or standard SA. The methods de-

scribed below attempt to achieve the good convergence properties of the element-based methods without requiring access to or manipulation of element information that may not even be available.

We refer to the approach developed here as adaptive because it involves self-testing to expose slowly converging error components and adaptation of the schemes' components to improve themselves. The acronym αMG is used to refer to the general class of multigrid methods of this type, to suggest their primal algebraic nature: we have in mind methods that use only the defining characteristic of algebraic smoothness and must use an automatic, algebraic process to determine additional characteristics that enable effective determination of the full MG algorithm. The additional abbreviations αAMG and αSA, respectively, are used to refer to the specific AMG and SA versions of αMG.

In the next section, we give a brief description of standard AMG to set the stage for the development of αAMG in the following sections. Section 3 provides an introduction into the adaptive framework and develops some fundamental principles that guide our construction of the adaptive process. In Section 4, we discuss details of the interpolation scheme used, as well as some of its theoretical properties. Some important details of implementation are discussed in Section 5, and numerical performance is illustrated in Section 6.

2. Standard AMG. The key to the efficiency of any multigrid process lies in the complementarity of the relaxation and coarse-grid correction processes. For this reason, any discussion of the classical AMG algorithm begins with the defining property of algebraic smoothness that the residual is, on average, small after a few sweeps of relaxation: $(A\mathbf{e})_i \approx 0$ for each point i. Considering pointwise relaxation, such as Gauss-Seidel, and a symmetric positive-definite operator, such errors are typically associated with the small eigenvalues of the operator, and this is often what is meant when discussing algebraically smooth errors. Also central is the assumed property that such errors vary slowly along strong connections in the matrix. This additional assumption is generally applicable for discretizations of scalar elliptic PDEs, for which AMG was originally designed. Other problems, such as systems of PDEs, may require other assumptions on the nature of algebraically smooth errors. In any case, awareness of this assumption is important in analyzing the performance of AMG, particularly when it is poor.

To construct interpolation to approximate a general algebraically smooth error, $\mathbf{e}$, we use the premise of a small residual (that, for example, $\mathbf{e}^T A^2 \mathbf{e} \ll \rho(A)\mathbf{e}^T A\mathbf{e}$, where $\rho(A)$ is the spectral radius of A) to conclude that, for a given row, i, $(A\mathbf{e})_i \approx 0$ or

$$a_{ii}e_i \approx -\sum_{j\neq i} a_{ij}e_j. \tag{2.1}$$

Now, suppose that a coarse set of points that forms a subset of the fine degrees of freedom (DOFs) has been chosen (strategies for which are discussed below). Then, the fine-level DOFs can be represented as $\{1, 2, \ldots, N\} = C \cup F$, where C is the set of coarse-level points and F is the set of remaining fine-level points (so $C \cap F = \emptyset$). Since A is sparse, we introduce "neighborhood" notation: $N_i = \{j \neq i : a_{ij} \neq 0\}$, $C_i = C \cap N_i$, and $F_i = F \cap N_i$. Equation (2.1) can then be rewritten as

$$a_{ii}e_i \approx -\sum_{j\in C_i} a_{ij}e_j - \sum_{k\in F_i} a_{ik}e_k. \tag{2.2}$$

Were the last sum not present in Equation (2.2), this expression could be used to define interpolation because it would give the F-point value, e_i, approximately as a sum of the C_i-point values. The aim is therefore to "collapse" the connections from point i to points $k \in F_i$ onto the points $\{j \in C_i\} \cup \{i\}$. That is, we want to set a_{ik}, $k \in F_i$, to 0 while adjusting a_{ij}, for $j \in C_i$, and a_{ii} in some way to compensate for the inaccuracies this elimination introduces. The main assumption needed to collapse the stencil is that the values of algebraically smooth $\mathbf{e}$ at F_i points can be written in terms of its values at points in $C_i \cup \{i\}$:

$$e_k \approx \sum_{j \in C_i} \omega_{kj}^i e_j + \omega_{ki}^i e_i, \text{ for } k \in F_i. \tag{2.3}$$

We could then substitute this expression into the latter sum in Equation (2.2) to obtain an expression for e_i in terms of $e_j, j \in C_i$, which is exactly the aim. Note that Equation (2.3) is a special form of interpolation from $C_i \cup \{i\}$ to F_i. This special interpolation formula is used in reducing the stencil connections to determine the final interpolation formula, so this overall process is sometimes called "twice-removed" or "iterated" interpolation.

Now, in classical AMG algorithms, the $\{\omega_{kj}^i\}$ in Equation (2.3) are determined from $\{a_{kj}\}$ based on the additional assumption that $\mathbf{e}$ is constant along strong connections. So, first we must ask the question as to whether the connection between i and k is important. To do this, we define the concept of strength of connection between two nodes. Considering the connections in the matrix to represent weights on the edges of the matrix graph (if $a_{ij} = 0$, there is no edge between nodes i and j), node i is said to strongly depend on node j if $-a_{ij} \geq \theta \max_{k \neq i}\{-a_{ik}\}$. Likewise, node i strongly influences node j if $-a_{ji} \geq \theta \max_{k \neq j}\{-a_{jk}\}$. Here, θ is a predetermined threshold value, $0 \leq \theta \leq 1$, often chosen to be 0.25.

Now, if k does not strongly influence i, then its value is not helpful in interpolation as it is not a strong interpolation point. The set of such k is denoted F_i^w and referred to as the set of weak connections. These connections are not simply discarded; rather, their values are collapsed onto the diagonal. That is, $\omega_{kj}^i = 0$ for $j \neq i$, and $\omega_{ki}^i = 1$. Notice that this approach is, in fact, quite stable. If the classification of $k \in F_i$ misidentifies a strong influence as a weak one, it replaces the value e_k with e_i, which (because of the assumption about smooth error and strong connections) are, in fact, approximately equal.

The remaining $k \in F_i \setminus F_i^w \equiv F_i^s$ all strongly influence i. Adding the requirement that a node $k \in F_i^s$ is also strongly dependent on some $j \in C_i$, then we can hope to determine the value at point k based on the coarse-grid neighbors it has in common with i. Since the strength of k's connection to a point $j \in C_i$ is proportional to the matrix coefficient a_{kj}, the value e_k can be approximated by the average of these e_j weighted by their coefficients. That is,

$$e_k = \frac{\sum_{j \in C_i} a_{kj} e_j}{\sum_{l \in C_i} a_{kl}}.$$

An important property of this intermediate interpolation formula is that it is an average; if $\mathbf{e}$ is constant for all $j \in C_i$, then it takes the same value at k. For

problems such as the Laplacian, where the smoothest mode is the constant vector (up to boundary conditions), it is quite important to ensure the accuracy of interpolation for this mode.

Combining these approximations yields an overall interpolation formula for $i \in F$: $e_i = \sum_{j \in C_i} w_{ij} e_j$, with

$$w_{ij} = -\frac{a_{ij} + \sum_{k \in F_i^s} \left(\frac{a_{ik} a_{kj}}{\sum_{l \in C_i} a_{kl}} \right)}{a_{ii} + \sum_{m \in F_i^w} a_{im}}.$$

The effectiveness of this interpolation operator is now very dependent on the quality of the coarse-grid points selected. This is apparent from the assumption that each point $k \in F_i^s$ has a strong dependence on some $j \in C_i$. In addition, we need the coarse grid to satisfy an often contradictory condition that it must be small enough that there is a real benefit to coarsening. There are many options for choosing the coarse-grid points, such as the approach of [27] where the number of strongly connected neighbors of each point is considered. Also of interest is the recent work on compatible relaxation; cf. [16, 19]. We do not discuss the selection process further, except to say that it is of primary importance to the efficiency and success of an algebraic multigrid method: the coarse level must support accurate approximation of algebraically smooth error using only a small fraction of the fine-level points.

Once a set of coarse points, C, and an interpolation operator, P, have been chosen, we must still choose a restriction operator, R (for transferring the residuals to the coarse level), and a coarse-level operator, A_c (for defining the coarse-level correction equation). Assuming that A is a symmetric and positive-definite matrix, it is natural to define these operators by the so-called Galerkin conditions (cf. [27]):

$$R = P^T \qquad \text{and} \qquad A_c = RAP.$$

These definitions arise from minimization principles and are a result of choosing R and A_c so that the coarse-level correction minimizes the fine-level A-norm of the error over all such corrections.

AMG is a generalization of classical geometric multigrid. It is an efficient solver for many problems, including those involving discretizations on stretched or irregular grids, or discretizations of many problems with anisotropic or variable coefficients. There are, however, still many problems for which classical AMG is not effective, including problems with highly anisotropic or highly variable coefficients and those coming from the discretization of certain systems of PDEs such as linear elasticity. Simply put, the further the algebraically smooth components of a problem are from being locally constant, the more the performance of classical AMG suffers.

3. The Adaptive AMG Framework. The details of the αAMG and αSA[11] algorithms are quite intricate. We arrived upon them by careful consideration of the basic principles and methodologies of an adaptive algorithm. For this reason, our discussion focuses on these basic principles before the particular details. We restrict our attention for the remainder of the paper to the case that the $N \times N$ matrix, $A = (a_{ij})$, is symmetric and positive definite, although most of what is developed

applies to more general cases. Our aim is to develop an algebraic multigrid process to solve the matrix equation,

$$A\mathbf{x} = \mathbf{b},$$

without a priori knowledge of the character of algebraically smooth error.

3.1. The Adaptive MG Algorithm. An efficient multigrid process for solving $A\mathbf{x} = \mathbf{b}$ relies on the appropriate complementarity of relaxation and coarse-grid correction. Thus, we view the goal of the adaptive process as the development of a representative collection of vectors for which the chosen relaxation process is inefficient. In its most basic form, the adaptive process would be quite simple: relax on a significant number of vectors to expose slow-to-converge components (which act as prototypes of algebraically smooth error) and then choose interpolation to fit these vectors. Such an approach may, however, be quite inefficient and a multiscale viewpoint often proves more useful.

Suppose we know the matrix, A, but nothing more. Relaxation is then the only feasible way to expose algebraically smooth error components. However, with no further knowledge, there is no way of predicting, a priori, how many distinct components are needed to achieve good results. Experience (and, in the case of SA, theory [29]) has shown that, for discretizations of second-order scalar elliptic PDEs, a single component is sufficient, whereas six components may be needed for a problem such as 3D linear elasticity. Thus, to arrive at an optimal solver with a minimal amount of work, it seems necessary to start with a single vector and introduce new prototypes as the evolving method proves inadequate.

The situation is then that we have a given matrix and seek to find a single algebraically smooth vector upon which to base interpolation. Relaxation alone can achieve this, simply by iteration on the homogeneous problem. However, it typically requires a huge number of relaxations to expose a global algebraically smooth vector, and so we seek to expose such components through multiscale development instead. We first relax only a few times, which should be sufficient to expose errors that are smooth enough to be handled on the first coarse grid. Smoother errors may be exposed by relaxation on the homogeneous problem on this coarse grid, particularly since the injected fine-grid prototype is a good initial guess for the algebraically smooth error we seek. The coarse-grid algebraically smooth error may then be used to create a still coarser grid in the usual multigrid fashion. Just a few steps of relaxation on the homogeneous problem on the finest grid quickly reduces a significant portion of a random initial guess, leaving error that can then be said to be locally algebraically smooth. If this prototype is used locally to define interpolation from some preselected coarse grid, then a coarse-grid problem that adequately represents the algebraically smooth error on the fine grid can be created. We can now iterate to an appropriate coarsest grid and interpolate a prototype of the smooth error to all grids. Proceeding recursively, this resembles a full approximation scheme (FAS; cf. [5]) multigrid process for the algebraically smooth component, rather than the usual correction scheme method.

In this manner, a good prototype of a single algebraically smooth component can be determined and the resulting solver tested by applying it to the homogeneous problem, $A\mathbf{x} = \mathbf{0}$, with a random initial guess. If it proves sufficient, then the adaptive stage is complete. Inefficiency in the resulting solver indicates that relaxation and coarse-grid correction are not yet appropriately complementary, and that there

are distinct algebraically smooth components that are not being well enough approximated on the coarser levels. Since these components are being reduced neither by relaxation nor coarse-grid correction, they can be exposed by an iteration as above with the current solver taking the place of relaxation. This may be repeated until acceptable convergence rates are attained.

Thus, we sketch the adaptive procedure as

ALGORITHM 1 (Abstract Adaptive Process).

1. *Let $k = 1$ and $\mathbf{x}^{(1)}$ be a random vector. Define, for all grids l, the methods $CYCLE_l(\mathbf{x}^{(l)},\mathbf{b}^{(l)})$ to be ν relaxation sweeps on $A^{(l)}\mathbf{x}^{(l)} = \mathbf{b}^{(l)}$.*
2. *$CYCLE_k(\mathbf{x}^{(k)},\mathbf{0})$.*
3. *If not sufficiently coarsened, form interpolation, I_{k+1}^k, and its (Galerkin) coarse-grid operator. Let $\mathbf{x}^{(k+1)} = (\mathbf{x}^{(k)})_c$ (that is, $\mathbf{x}^{(k)}$ evaluated at the grid $k+1$ points), $k = k+1$, and goto Step 2.*
   Otherwise, continue.
4. *While $k > 1$, let $k = k-1$, interpolate the coarse-grid approximation, $\mathbf{x}^{(k)} = I_{k+1}^k\mathbf{x}^{(k+1)}$, and perform $CYCLE_k(\mathbf{x}^{(k)},\mathbf{0})$.*
5. *Let $k = 1$ and $\mathbf{x}^{(1)}$ be a random vector. For all grids l, redefine the cycle, $CYCLE_l(\mathbf{x}^{(l)},\mathbf{b}^{(l)})$, to be ν current V-cycles on $A^{(l)}\mathbf{x}^{(l)} = \mathbf{b}^{(l)}$. If the performance of $CYCLE_1(\mathbf{x}^{(1)},\mathbf{0})$ is not acceptable, go to Step 2.*

3.2. Principles. Perhaps the easiest way to understand the adaptive methodology is to begin with the principles upon which it is based. Here, we list the core ideas that motivate and provide a foundation for αMG, with the primary focus on the αAMG scheme. The pragmatic reader may prefer to defer reading this discussion until after Section 4 or [11].

Algebraic Smoothness. The concept of algebraic smoothness is of utmost importance in achieving an optimally efficient algebraic multigrid method. Since we only allow reduction of the error through the processes of relaxation and coarse-grid correction, the algebraically smooth error (which, by definition, is slow to be resolved by relaxation) must be accurately corrected from the coarse grid. That is, interpolation must be very accurate for algebraically smooth components. In fact, a stronger requirement is imposed by the eigenvector approximation criterion that, for a given eigenvector, $\mathbf{u}$, of A, interpolation must reconstruct $\mathbf{u}$ to an accuracy proportional to its eigenvalue [6, 22].

The algebraic multigrid methods considered here are based on some defining characteristic of what algebraic smoothness means. This definition generally amounts to articulating an algebraic property of the errors that the given (fixed) relaxation process cannot reduce effectively. For example, classical AMG is developed based on properties of a polynomial iterative method such as the Richardson iteration:

$$\tilde{\mathbf{x}} \leftarrow \tilde{\mathbf{x}} - \frac{1}{\|A\|}(A\tilde{\mathbf{x}} - \mathbf{b}).$$

For this iteration, the error, $\mathbf{e} = A^{-1}\mathbf{b} - \tilde{\mathbf{x}}$, converges slowly in the A-norm,

$$\|\mathbf{e}\|_A = \sqrt{< A\mathbf{e}, \mathbf{e} >},$$

if and only if $\mathbf{e}$ yields a small generalized Rayleigh quotient:

$$RQ_A(\mathbf{e}) = \frac{< A\mathbf{e}, A\mathbf{e} >}{\|A\| < A\mathbf{e}, \mathbf{e} >}.$$

For this reason, the set of algebraically smooth vectors is often also referred to as the near null space of A; we use these two terms interchangeably, with the prevailing meaning to be that of vectors that the chosen relaxation scheme is slow to reduce. Proper use of this defining property of algebraic smoothness gives AMG its potential for optimal performance over a wide range of problems. It enables coarsening processes that rightfully depend on the matrix and hopefully capture the errors that relaxation cannot eliminate.

Most algebraic multigrid methods, however, require additional assumptions regarding algebraically smooth error that allows them to capitalize on the special nature of the assumed algebraic smoothness. For example, classical AMG rests on two related main assumptions: that the constant vector, $\mathbf{1}$, must be interpolated exactly; and that algebraically smooth errors vary slowly along strong connections. While this enables effective treatment of many problems, it also restricts the class to which these algorithms apply. Many discrete systems exhibit algebraically smooth errors that vary dramatically across strong connections and many others offer no clear understanding of what strength of connection even means. Also, as we discuss further in the next section, $\mathbf{1}$ is not necessarily a good representative of algebraically smooth error. A major goal of the adaptive process is to capitalize on the definition of algebraically smooth error without making additional specific assumptions about its character.

Prototypes. A central idea in the development of αAMG methods is the use of prototypes that serve as representatives of algebraically smooth error. In fact, prototypes are used in the development of nearly all multigrid methods, often implicitly. As mentioned above, for example, classical AMG uses $\mathbf{1}$ to build its matrix-based interpolation coefficients (see Equation 2.1 and the discussion in Section 2). αAMG differs in that it attempts to generate these prototypes automatically.

Given a set of these prototypes, it is important to recognize that they should only be used locally as representatives of algebraically smooth error. Otherwise, it would not be possible to achieve optimality. As an illustration of this point, consider the fact that the preconditioned conjugate gradient method can be interpreted as a two-level αMG method that uses the generated prototypes globally as representatives of slow-to-converge error components (here, the smoother plays the role of the preconditioner and the coarse-grid corrections are the Krylov subspace projections; see [15] for details). In general, errors left by relaxation consist of a large fraction of the spectrum of the matrix, so that coarsening must effectively approximate $O(N)$ components with varying degrees of accuracy. The goal in designing a multigrid interpolation operator is to achieve this approximation property by using only a small, $O(1)$, number of computed prototypes.

This use of a small number of prototypes to achieve approximation of a much larger space is a cornerstone of multigrid methods. It is essential, then, that each prototype be used effectively as a representative of many components with similar local character. Remember that the constant vector, $\mathbf{1}$, is used in classical AMG to define an interpolation whose range not only includes $\mathbf{1}$, but also approximates all smooth errors. This is analogous to how local basis functions are used in finite elements: piecewise polynomial basis functions are used locally not only because they can reconstruct their global counterparts, but also because they can approximate all smooth components of the solution of the PDE. The global prototype is a representative of many algebraically smooth components and, thus, is used locally to determine an interpolation operator that has many such smooth components in its range.

Self-Testing. Computation of a rich supply of prototypes can be done by carefully testing the algorithm as it evolves. These self-tests should be done on a problem with known solution. The homogeneous problem, $A\mathbf{x} = \mathbf{0}$, is especially appropriate because it avoids trouble with machine representation when the approximation is very close to that solution. For our αMG schemes, we can test the current version of the algorithm on $A\mathbf{x} = \mathbf{0}$ by measuring the A-norm of the error of successive iterates. This test serves a dual role: it signals when the algorithm is performing well enough and it produces a good prototype when it is not. Assuming that enough iterations are used, the prototype must be appropriate because it is algebraically smooth (relaxation is not eliminating it), yet poorly represented by whatever current coarsening process is being used (if any). This prototype can then be used in the underlying algorithm precisely where the additional smoothness assumptions were used. For classical AMG, this means that the prototype would provide information on the correct coefficients to use in eliminating the matrix connections to points that are only on the fine level.

While the homogeneous problem is important as a measure of performance because it has a known solution, other measures can be useful in monitoring the evolving behavior and improving the prototypes. This issue is most clearly exposed when a direct solver is used on the coarsest level, where solving the homogeneous problem seems paradoxical: why solve $A\mathbf{x} = \mathbf{0}$ when all you presumably get is $\mathbf{x} = \mathbf{0}$? At this point, a simple approach is to just accept the prototype computed on the next finer level so that the coarsest level is never really used in the adaptive process. This means that the prototype is never really improved there either. It may be better to enhance the coarsest-level prototype by using a more precise measure of smoothness. For our αMG schemes, we could choose to improve the prototype, $\mathbf{x}$, on the coarsest level by minimizing the generalized Rayleigh quotient, $RQ_A(\mathbf{x})$. The situation is more complicated in the case of multiple prototypes, however; in neither the scalar PDE case considered in §4 nor in the αSA method in [11] have we found any need for improving the coarsest-level prototype in this manner, so we choose not to address this issue further. The Rayleigh quotient can, however, be useful as a diagnostic tool in assessing how the adaptive process is behaving, as in [20].

Range of Interpolation. The primary aim of coarsening in any multigrid process is to appropriately complement relaxation, allowing fast solution of the given system of equations. To accomplish this, all algebraically smooth errors must be approximated well by vectors in the range of interpolation. Within the adaptive process, such errors are represented only by the prototypes and, therefore, the primary aim of this process is to develop an interpolation operator that accurately approximates the prototypes in such a way that good multigrid convergence is obtained by the resulting cycle.

Achieving the desired multigrid performance requires more than simply fitting the set of given prototypes exactly. Error in the direction of a single prototype can easily be corrected from a single coarse degree of freedom, but this generally does not provide enough of an acceleration to the relaxation process to yield a scalable algorithm. Indeed, such an algorithm resembles a preconditioned conjugate gradient iteration [15] that would require many steps to converge for the matrices that typically arise from discretized PDEs. To develop an effective interpolation operator, it is necessary to consider each prototype as a *representative* of a class of errors that relaxation is slow to reduce. Since pointwise relaxation is a local process, any error that has a similar character to the prototype, measured locally, is also expected to be slow to converge with relaxation. We look for all such errors to be approximated by the range of

interpolation and, thus, employ a localization (partition of unity) strategy in defining our multigrid interpolation operator. This strategy, unfortunately, is also generally not sufficient to achieve good multigrid performance: piecewise constant interpolation fits the prototypical constant vector exactly, and many locally similar errors well, yet leads to poor V-cycle performance for solving Laplace's equation [5]. Interpolation must also be chosen so that a variational coarse-grid correction is not distracted by non-algebraically smooth errors that are also in the range of interpolation. Classical AMG interpolation (as discussed in §2) is known to attain all of these properties for typical scalar elliptic equations based on the single prototype of the constant vector, and we aim to mimic these properties here in the more general adaptive AMG setting.

Constructing interpolation in this way (described in detail in Section 4) has important consequences. Not only is each prototype fit closely by the range of interpolation, but so are all locally similar vectors are as well. Thus, the range of interpolation includes many algebraically smooth vectors, including, if possible, many vectors that are significantly smoother (measured, for example, by their Rayleigh quotients) than the prototype used to construct interpolation in the first place. Given the lower cost of relaxation on the coarser grid, we look to improve the prototype through iteration on the homogeneous coarse-grid problem. Thus, we coarsen in the adaptive process, to quickly expose better prototypes of the fine-level algebraically smooth error. As the fine-level prototype is, itself, well-represented on the coarse grid, the adaptive process attempts to improve the prototype directly on this level, instead of through a more cumbersome correction scheme. Success of this process rests on our ability to find components in the range of the constructed interpolation operator that are significantly smoother than the current prototype. Again, the keys to this success are in AMG's local operator-induced interpolation approach, which ensures that the range of P is rich in components that are locally similar to the prototypes, and the variational construction of the coarse-grid operators through the Galerkin condition [27], which allows iteration on the coarse grid to find a significantly smoother prototype than the one upon which interpolation was based (if such a vector exists).

Optimality. Multigrid methods are useful solution techniques because they exhibit optimal traits, such as $O(N)$ or $O(N \log N)$ scaling in both number of operations and storage. As such, any adaptive multigrid process should also retain this optimality. In particular, the adaptive process must not make requirements of the solver that compromise the optimality of the overall process and must itself scale optimally in operation count and storage.

Classical AMG controls complexity by its intricate way of determining the coarse points and its careful use of the matrix entries. The adaptive AMG approach assumes that a suitable coarsening process is available (such as compatible relaxation [7, 16, 19]), with the attendant assumption that there is sufficient reduction in grid sizes from fine to coarse levels. When fitting multiple prototypes, however, it is tempting to abandon the tight control on the stencil of interpolation (such as to the nonzero pattern of A_{fc}, the submatrix of A linking fine- to coarse-grid nodes, as is used in classical AMG) to allow for exact fitting of more prototypes. This must be done with utmost care, as each new nonzero entry in the interpolation operator can lead to new nonzero connections in the coarse-grid matrix. Care must also be taken to ensure that the stencil of interpolation is not too small: early experiments limited the size of the set of coarse-grid points from which any fine-grid point i is interpolated, C_i, to the number of prototypes being fit, which led to an extremely inefficient algorithm because a single prototype could only be used to define one-sided interpolation, and

so multiple prototypes were needed for good convergence even for second-order, scalar PDEs.

Constructing a prototype set of minimal size is important to practical success and control over the complexity of the algorithm. While the same near null space may be well-represented by a small number of very precise vectors or a larger number of less-resolved prototypes, the costs of the adaptive process and, possibly, the resulting method increase with the number of prototypes. For this reason, it is often more efficient to consider improvement of the existing prototype(s) than to add a new one. As each prototype emerges, we can consider improvement of its representation by running the setup cycle again, with the prototype as an initial guess for the algebraically smooth error. This either improves the prototype as a representative of the algebraically smooth components that are not well-represented by the rest of the prototype set, or signals that the prototype is a strong representative of such components. In either case, valuable information about the adaptive process is gained, either through an improved prototype, or the knowledge that further improvement in the resulting algorithm must be gained through additional prototypes.

To ensure that the adaptive process is also optimal, adaptations are made whenever sufficient new information becomes known, but also only when the change is expected to improve the overall algorithm. For example, we develop the algebraically smooth prototype in a manner similar to the full approximation scheme, representing the prototype on all levels and looking for the algebraically smoothest component on each level. The prototype on a given level is discarded when it can be improved from a coarser grid, as we expect relaxation on coarser levels to better expose the algebraically smooth components that we seek. We do not, however, update interpolation or coarse-grid operators on the upward traverse of the setup V-cycle. Such an adaptation would be wasted because operators at higher levels also change as the cycle moves toward the finest grid. For this reason, while we allow for multiple V-cycles to be performed in the setup phase, the last cycle always terminates at the coarsest grid.

Termination of the adaptive process must also be properly implemented in order to maintain optimality. Experience has shown that improvement in the resulting multigrid process becomes less cost-effective with the number of setup phases and the total amount of relaxation in the adaptive step. A method with an acceptable convergence factor may be attained after even a single adaptive step, and a second adaptive step improves this factor by only a fraction of a percent. This may be addressed by reducing the amount of relaxation per adaptive step to a single sweep on each level, and monitoring the convergence of the prototype vector between sweeps (for example, measuring its Rayleigh quotient). Unfortunately, the majority of the cost of an adaptive step is in the computation of interpolation and coarse-grid operators and not relaxation, so performing many adaptive steps is undesirable. For this reason, we choose a strategy attempting to minimize the number of setup cycles, motivated in [20], with enough relaxations in these cycles to quickly achieve convergence factors within a few percent of the apparent optimal performance.

4. Interpolation for Adaptive AMG. Our goal in developing a new type of algebraic multigrid method is to extend the applicability of classical AMG schemes. Thus, a primary concern is the generalization of the definition of interpolation in AMG. The guiding principles for this generalization come from basic properties of all multigrid algorithms:

- simple relaxation is inefficient for solving $A\mathbf{x} = \mathbf{b}$ on error components, $\mathbf{e}$,

whose residuals, $A\mathbf{e}$, are small relative to $\mathbf{e}$ in some sense; and
- efficient multigrid performance depends on effective complementarity of relaxation and coarsening so that they efficiently cooperate to eliminate all error components.

In this section, we propose a method for choosing interpolation that is appropriate in the case of M-matrices for which the algebraically smoothest mode is not close to the constant vector. As such, this interpolation may be viewed as an extension of the classical AMG interpolation [27]. In developing this new interpolation procedure, we consider the case of pure algebraic coarsening; however, for practical reasons, we chose to first implement the algorithm in the case of regular geometric coarsening. We do not address the important question of coarse-grid selection here, but expect that, in a practical situation, the coarsening techniques of compatible relaxation [7, 16, 19] would be necessary to choose an adequate coarse grid. The numerical results presented in Section 6 are for this interpolation on geometrically chosen grids in the case of a scalar PDE.

4.1. Definition of Interpolation. Since the success of our methods depends on the complementarity of relaxation and coarse-grid correction, a good starting point for defining interpolation is to consider a vector, $\mathbf{e}$, that is not quickly reduced by relaxation. Using a simple (pointwise) relaxation scheme, such as Gauss-Seidel, this also means that $A\mathbf{e} \approx \mathbf{0}$, or

$$a_{ii}e_i \approx -\sum_{j\in C_i} a_{ij}e_j - \sum_{k\in F_i} a_{ik}e_k, \tag{4.1}$$

where, as in Equation 2.2, we assume a splitting of N_i into C_i and F_i. Again writing error e_k, $k \in F_i$, as

$$e_k \approx \sum_{j\in C_i} \omega_{kj}^i e_j + \omega_{ki}^i e_i$$

and using this expression in the second sum of Equation 4.1 yields a general interpolation formula for point $i \in F$,

$$e_i = -\sum_{j\in C_i} \left(\frac{a_{ij} + \sum_{k\in F_i} a_{ik}\omega_{kj}^i}{a_{ii} + \sum_{k\in F_i} a_{ik}\omega_{ki}^i} \right) e_j. \tag{4.2}$$

The αAMG interpolation is different from that used in classical AMG [27] in that $\{\omega_{kj}^i\}$ are chosen to depend on both the entries in A and a (computed) prototype, $\mathbf{x}^{(1)}$, that represents many algebraically smooth components. How this prototype is computed is the subject of Section 5.

To be specific about the choice of $\{\omega_{kj}^i\}$, consider the idea of twice-removed interpolation [8]. Suppose we have a point, i, whose neighbors have been partitioned into the two sets, C_i and F_i. The problem of collapsing the $F-F$ connections is equivalent to that of determining a way to interpolate to point $k \in F_i$ from points $j \in C_i$ (or, more generally, $j \in C_i \cup \{i\}$). That is, we seek to write (as before)

$$e_k = \sum_{j\in C_i} \omega_{kj}^i e_j, \tag{4.3}$$

dropping the term $\omega^i_{ki}e_i$, under the assumption that k is as strongly connected to some point (or points) in C_i as it is to i. If there is a particular vector, $\mathbf{x}^{(1)}$, that we want to be in the range of interpolation, then we ask that Equation 4.3 hold when $\mathbf{e}$ is replaced by $\mathbf{x}^{(1)}$. This constraint with one vector, $\mathbf{x}^{(1)}$, fixes one DOF of the possibly many for set $\{\omega^i_{kj}\}$, but leads to a unique F_i-interpolation formula if it is restricted to be of the form $D^{-1}A_{fc}$, where D is a diagonal matrix. (This choice is motivated by the discussion in [10].) The matrix, D, is determined by

$$d^i_{kk}x^{(1)}_k = -\sum_{j\in C_i} a_{kj}x^{(1)}_j$$

or

$$d^i_{kk} = \frac{-\sum_{j\in C_i} a_{kj}x^{(1)}_j}{x^{(1)}_k}. \tag{4.4}$$

Thus, choosing $\omega^i_{kj} = (d^i_{kk})^{-1}a_{kj}$ in Equation 4.3, the F_i-interpolation formula is

$$e_k = -\sum_{j\in C_i} \frac{a_{kj}}{d^i_{kk}} e_j.$$

Interpolation to $i \in F$, given by Equation 4.2, then has the particular form

$$e_i = -\sum_{j\in C_i} \frac{1}{a_{ii}} \left(a_{ij} + \sum_{k\in F_i} a_{ik} \frac{a_{kj}x^{(1)}_k}{\sum_{j'\in C_i} a_{kj'}x^{(1)}_{j'}} \right) e_j. \tag{4.5}$$

Note that the interpolation operator, P, as a mapping from C to $F\cup C$, then has the form

$$P = \begin{bmatrix} W \\ I \end{bmatrix},$$

where W is the matrix of coefficients determined by Equation 4.5.

This αAMG interpolation formula is a simple generalization of the classical AMG formula that allows for a sense of smoothness that may differ from what AMG conventionally uses. The primary assumption used in standard AMG to collapse $F-F$ connections is that the smoothest error component is constant [27]. Thus, classical AMG interpolation is recovered from the formula in Equation 4.5 by choosing $\mathbf{x}^{(1)} \equiv \mathbf{1}$.

The iterated interpolation of Equations 4.3 and 4.4 was chosen to exactly match the near null space approximation, $\mathbf{x}^{(1)}$. The final interpolation in Equation 4.5, however, does not necessarily match this vector exactly. For a fine-grid point, i, the misfit is easily calculated as

$$x^{(1)}_i - (P\mathbf{x}^{(1)}_c)_i = \frac{1}{a_{ii}}(A\mathbf{x}^{(1)})_i. \tag{4.6}$$

This is in accord with the classical AMG point of view that interpolation must be more accurate for errors that yield smaller residuals. In fact, it may be directly compared

with the eigenvector approximation criterion, as described by Brandt [6, Theorem 4.1], which relies on the existence of a constant, C_0, bounded away from zero, such that

$$C_0 \sum_i a_{ii}(e_i - (P\mathbf{e}_c)_i)^2 \leq \mathbf{e}^T A\mathbf{e},$$

for all $\mathbf{e} \in \mathbb{R}^N$. Squaring Equation 4.6, multiplying through by a_{ii}, and summing over both fine- and coarse-grid points, we have

$$\begin{aligned}\sum_i a_{ii}(x_i^{(1)} - (P\mathbf{x}_c^{(1)})_i)^2 &\leq \sum_i \frac{1}{a_{ii}}(A\mathbf{x}^{(1)})_i^2 = (\mathbf{x}^{(1)})^T A \left(\text{diag}(\frac{1}{a_{ii}})\right) A\mathbf{x}^{(1)} \\ &\leq \rho\left(A^{\frac{1}{2}}\left(\text{diag}\left(\frac{1}{a_{ii}}\right)\right)A^{\frac{1}{2}}\right)(\mathbf{x}^{(1)})^T A\mathbf{x}^{(1)}.\end{aligned}$$

For a diagonally dominant operator with constant diagonal, such as the finite-element Laplacian, $\rho\left(A^{\frac{1}{2}}\left(\text{diag}\left(\frac{1}{a_{ii}}\right)\right)A^{\frac{1}{2}}\right)$ is easily bounded by 2 and so the constant, C_0, in Brandt's bound is not made unduly small due to the misfit in the interpolation of $\mathbf{x}^{(1)}$. While such a bound is only for the prototype, $\mathbf{x}^{(1)}$, and not for any arbitrary fine-grid vector (as required by the eigenvector approximation criteria), we consider $\mathbf{x}^{(1)}$ to be an appropriate prototype of the algebraically smooth error for which this bound is most difficult to achieve and, thus, indicative of a good interpolation scheme.

Remark 1. While this paper considers methods involving just one prototype vector, $\mathbf{x}^{(1)}$, appropriate for scalar PDEs, these concepts can also be generalized to systems. Consider discretizing a system so that its DOFs are located on the same grid, i.e., there are d DOFs co-located at each node. Since we seek to generalize the ideas from the scalar case, we start by generalizing the notation: A_{kj} becomes the $d \times d$ matrix of connections between the DOFs located at nodes k and those located at node j, the diagonal entries of D (D_{kk}^i) become $d \times d$ matrices, and $\mathbf{x}^{(1)}$ becomes the matrix, $X^{(1)}$, composed of d columns of distinct prototypes. Its restriction to the d DOFs at node k is denoted by $X_k^{(1)}$. The analogue of Equation 4.4 is then

$$D_{kk}^i = -\left(\sum_{j \in C_i} A_{kj} X_j^{(1)}\right)\left(X_k^{(1)}\right)^{-1}.$$

The F_i-interpolation formula for systems thus becomes

$$\mathbf{e}_k = -\sum_{j \in C_i} (D_{kk}^i)^{-1} A_{kj}\mathbf{e}_j,$$

which yields the final nodal interpolation formula

$$\mathbf{e}_i = -A_{ii}^{-1}\left(\sum_{j \in C_i}\left(A_{ij} + \sum_{k \in F_i} A_{ik}(D_{kk}^i)^{-1}A_{kj}\right)\right)\mathbf{e}_j.$$

4.2. Theoretical Properties. One situation that can cause difficulty for classical AMG is when the matrix is rescaled. For example, if A is the discretization of a Poisson-like problem, then it is generally true that A applied to $\mathbf{1}$ yields a relatively small residual: $A\mathbf{1} \approx \mathbf{0}$. This means that constant vectors are indeed algebraically

smooth, as classical AMG presumes. Rescaling A by multiplying it on both left and right (to preserve symmetry) by a positive diagonal matrix can dramatically change this property. Thus, if A is replaced by $\hat{A} = SAS$ for some positive diagonal matrix S, then $\hat{A}(S^{-1}\mathbf{1}) \approx 0$, and the new near null space component is actually $S^{-1}\mathbf{1}$. If the diagonal entries of S have significant variation in them, then $S^{-1}\mathbf{1}$ has a significantly different character than does $\mathbf{1}$. For classical AMG, this can cause a dramatic deterioration in convergence rates, although it can be prevented if the scaling is supplied to AMG so that the original matrix can essentially be recovered (as in [8]), but this is not always possible in practice. Fortunately, as the following result shows, such scaling causes no problem for αAMG, provided the scaled prototype can be accurately computed.

THEOREM 4.1. *Given a positive diagonal matrix, S, vectors $\mathbf{x}^{(1)}$, $\hat{\mathbf{x}}^{(1)} = S^{-1}\mathbf{x}^{(1)}$, $\mathbf{b}$, and $\hat{\mathbf{b}} = S\mathbf{b}$, then the convergence of αAMG on $\hat{A}\hat{\mathbf{x}} = \hat{\mathbf{b}}$ with prototype $\hat{\mathbf{x}}^{(1)}$ (measured in the $\hat{A}$-norm) is equivalent to that of αAMG on $A\mathbf{x} = \mathbf{b}$ with prototype $\mathbf{x}^{(1)}$ (measured in the A-norm).*

Proof. Given a coarse-grid set, C, and the complementary fine-grid set, F, partition A and S to have the forms

$$A = \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix} \text{ and } S = \begin{bmatrix} S_f & 0 \\ 0 & S_c \end{bmatrix},$$

so that

$$\hat{A} = \begin{bmatrix} S_f A_{ff} S_f & S_f A_{fc} S_c \\ S_c A_{cf} S_f & S_c A_{cc} S_c \end{bmatrix}.$$

The weights, $\hat{\omega}^i_{kj}$, for matrix $\hat{A}$ are given by

$$\begin{aligned} \hat{\omega}^i_{kj} &= \frac{\hat{a}_{kj}\hat{x}^{(1)}_k}{\sum_{j' \in C_i} \hat{a}_{kj'}\hat{x}^{(1)}_{j'}} \\ &= \frac{s_k a_{kj} s_j s_k^{-1} x^{(1)}_k}{\sum_{j' \in C_i} s_k a_{kj'} s_{j'} s_{j'}^{-1} x^{(1)}_{j'}} = s_k^{-1}\omega^i_{kj} s_j, \end{aligned}$$

where the weights, ω^i_{kj}, are chosen as for matrix A. Equation 4.5 then gives (with some algebra)

$$e_i = -\sum_{j \in C_i} s_i^{-1} \left(\frac{a_{ij} + \sum_{k \in F_i} a_{ik}\omega^i_{kj}}{a_{ii}} \right) s_j e_j, \quad i \in F.$$

For $i \in C$, again simply take the value from the coarse-grid and assign it as the value on the fine-grid. Thus, the interpolation operator, $\hat{P}$, is of the form

$$\hat{P} = \begin{bmatrix} \hat{W} \\ I \end{bmatrix} = \begin{bmatrix} S_f^{-1} W S_c \\ I \end{bmatrix} = S^{-1} P S_c,$$

where P is the interpolation operator from the unscaled case. Further, considering the coarse-grid operator, $\hat{A}_c$, note that $\hat{A}_c = S_c P^T A P S_c = S_c A_c S_c$, where A_c is the coarse-grid operator from αAMG on A.

So, the coarse-grid operator for the scaled problem is simply the scaled version of the coarse-grid operator for the unscaled problem. Since standard relaxation techniques such as Gauss-Seidel or Jacobi (both pointwise and block forms) are scaling invariant (that is, if A is scaled to SAS as above, initial guess $\mathbf{x}^{(0)}$ to $S^{-1}\mathbf{x}^{(0)}$ and initial right side $\mathbf{b}$ to $S\mathbf{b}$, then the approximation generated changes from $\mathbf{x}^{(1)}$ to $S^{-1}\mathbf{x}^{(1)}$), we see that the entire process is independent of any diagonal scaling. □

THEOREM 4.2. *Theorem 4.1 extends to the systems algorithm, which is invariant to diagonal scaling with pointwise relaxation and nodal scaling (any invertible transformation of the DOFs located at a node) with nodal relaxation.*

Proof. The proof is identical in form to the scalar case, and is, thus, omitted. □

5. Determining $\mathbf{x}^{(1)}$. Successful implementation of this interpolation scheme relies on having an appropriate prototype vector, $\mathbf{x}^{(1)}$ (or set of vectors, $X^{(1)}$). Since we rely on the complementarity of relaxation and coarsening, the best choice for $\mathbf{x}^{(1)}$ would be a representative of the vectors for which relaxation is inefficient. Thus, a straightforward method for generating this prototype would be to start with a vector that is hopefully rich in all components (i.e., all eigenvectors of symmetric A), relax on $A\mathbf{x} = \mathbf{b}$ for some $\mathbf{b}$, and then determine the error in the approximate solution after a sufficient number of relaxations.

We typically make use of relaxation schemes whose error-propagation matrices have the form $I - BA$. While, for general B, it is possible that the slow-to-converge modes of the relaxation iteration, $I - BA$, are not modes for which $A\mathbf{e} \approx \mathbf{0}$, in many practical situations they are. In particular, for the pointwise relaxation schemes considered here, the two descriptions of algebraically smooth error are equivalent. In fact, for many choices of B, the true near null space of A is accurately reflected in the vectors for which relaxation is inefficient. Knowledge of this space could be used as it is with standard AMG to determine an effective coarsening process. Our focus, however, is on the case where this knowledge is misleading, inadequate, or even unavailable. We, thus, concentrate on the case that a good prototype of errors that are slow to converge under the given relaxation scheme, $\mathbf{x}^{(1)}$, is not known.

Start with a vector generated randomly from a uniform distribution on $(0, 1)$. (Consideration of positive vectors is motivated by the case of scalar, second-order operators, which tend to discretize as M-matrices and so have positive near null space vectors. A more general choice is appropriate when considering problems such as linear elasticity, but care must be taken because the definition of interpolation for αAMG presented here breaks down if $\sum_{j\in C_i} a_{kj}x_j^{(1)} = 0$ for any $i \in F$, $k \in F_i$.) Such a vector is, in general, not equally rich in all error components. However, in the scalar PDE case, it tends to be rich enough that a few relaxation sweeps on the homogeneous problem, $A\mathbf{x} = \mathbf{0}$, produce a good representative of the slow-to-converge components. Note that the homogeneous problem is advantageous to use here because the prototype is simply the error in approximating the exact solution, $\mathbf{x} = \mathbf{0}$. Thus, starting with a random initial guess and performing relaxation on $A\mathbf{x} = \mathbf{0}$ generates a prototype vector, $\mathbf{x}^{(1)}$, that represents the slow-to-converge components and that can then be used in the interpolation formula developed in Section 4.1.

Unfortunately, generating the prototype by fine-grid relaxation alone is effective only in terms of the first coarse level and is, in general, quite inefficient in a multilevel setting. To produce a prototype that is smooth enough to represent the components associated with smoothness on very coarse levels, a multilevel scheme is needed. Here, we again measure smoothness by the eigenvector approximation criterion. Basing every interpolation operator on a single component, whose Rayleigh quotient is near the

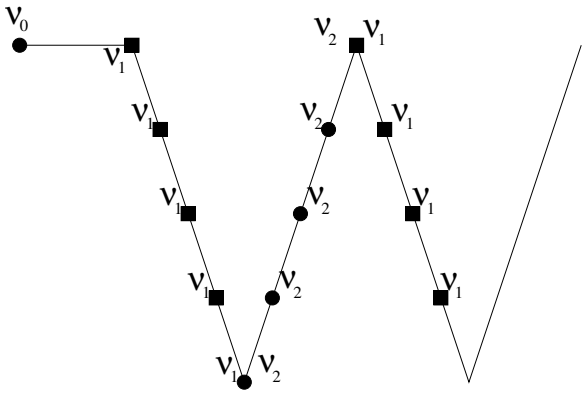

FIG. 5.1. *The Setup Scheme for Determining* $x^{(1)}$

minimal eigenvalue on the finest grid, requires significant algebraic smoothness in that component as the eigenvector corresponding to this eigenvalue must be interpolated with accuracy proportional to this small value. For coarser grids, such smoothness is much more efficiently represented by calculation on these grids. Thus, we start with a random guess on the fine level and perform a few (ν_0) relaxation sweeps there to generate a tentative $\mathbf{x}^{(1)}$. Using this current prototype, an interpolation operator is computed (as in Section 4.1) and the coarse-level and restriction operators are formed using the Galerkin condition. We use injection of $\mathbf{x}^{(1)}$ (direct restriction of the values on the C-points) to form a coarse-level initial guess, relax ν_1 times, and recurse to the coarsest level. From this coarsest level, interpolate and relax ν_2 times on the vector all the way to the finest level, but do not recompute the coarse-level and grid-transfer operators.

The cycle can then be repeated, using the resulting vector as an overall initial guess in an attempt to further improve the prototype. This cycling strategy is illustrated in Figure 5.1, where boxes indicate stages where coarse-level operators are computed and circles indicate stages where only application of the current solver is necessary. Note that since the multigrid operators are computed only on the downward part of the cycle, no relaxation is necessary on the upward part of the final setup cycle. As is discussed in Section 6, this procedure yields multigrid solvers with convergence factors bounded independently of mesh size (tested up to 1024×1024 grids) for many scalar problems.

One important benefit of generating the initial prototype vector in a multilevel fashion is the ability to implement a proper transition to simplicity in the algorithm. That is, since we begin by relaxing on a random vector (assumed to be rich in all components), it is easy to tell if relaxation alone is sufficient to solve either the fine grid problem or one of the generated coarse-level problems. If this is indeed the case, then no additional labor is needed in designing an algorithm because an efficient solver already exists.

For systems of PDEs and higher-order problems, an added wrinkle is the need to generate multiple prototype vectors. The cycling scheme described above can easily be generalized to account for multiple prototypes, by replacing the role of relaxation with the current solver. That is, once a single prototype is known, additional prototypes need to be generated that are both algebraically smooth, and that represent distinct slow-to-converge components from the existing prototype. One way to gen-

erate such components is to apply the current solver to the homogeneous problem, in much the same way as we currently use relaxation to expose its own slow-to-converge components. Further investigation is necessary, however, to ensure that the generated prototypes represent a full set of algebraically smooth errors, without being redundant. This is the subject of current research.

6. Numerical Results. To examine the feasibility of this approach, we implemented a solver for the special case of a rectangular grid in two dimensions with full coarsening. This restriction in generality has a notable effect on the range of problems that are able to be reasonably considered (anisotropy, for example, becomes much more difficult to account for in this setting). However, examining problems without such difficulties, we feel that we can obtain a good initial indication of the performance of this approach. In particular, the aim here is to test the quality of the interpolation operator, not that of the coarsening procedure. Indeed, given current research into new coarsening techniques [7, 16, 19], it is difficult to say which coarsening method would be most appropriate for comparison.

We consider several measures of the effectiveness of the algorithm. Of primary importance is the total time to solution, given only the matrix equation, $A\mathbf{x} = \mathbf{b}$. Here, solving a problem is defined as reducing the residual by 10 orders of magnitude, and the wall-clock time to solution on a modern desktop workstation (2.66 GHz Intel Pentium 4) is measured. Another relevant measure is the asymptotic convergence factor of the resulting cycle. While setup costs may form a significant portion of the cost of solving a linear system with a single right side, many problems require repeated solution with multiple right sides (such as in implicit time stepping). In these cases, the (possibly large) setup cost can be amortized over the number of solutions and the cost of only the solution stage is important. The asymptotic convergence factor reflects this cost, as a lower factor requires fewer iterations in the solution phase. We discard the usual AMG measures of grid and operator complexity because, in the structured coarsening framework considered here, these measures are constant for all fine-grid operators of the same mesh and stencil sizes.

As discussed in Section 5, the setup may be performed in an iterative fashion. This is an appealing feature because, in practice, convergence of the prototype vector can be measured as an indicator of convergence of the method. This iteration is, however, quite expensive as it involves computation of new interpolation and new Galerkin coarse-grid operators at each iteration. Testing indicates that it is usually significantly more efficient to perform more relaxation sweeps (i.e., increase ν_0 and ν_1) in a single setup cycle than it is to perform multiple setup cycles with fewer iterations per cycle to generate a single prototype [20]. Thus, in the results that follow, we first consider performing only a single setup cycle and track the number of relaxations (values of ν_0 and ν_1) necessary to achieve highly efficient solver performance. For the problems considered here, this calibrated AMG method is typically more efficient than the adaptive AMG method, whose results appear later. In the adaptive procedure, we fix the number of relaxations used in each setup phase and perform setup iterations until an acceptable solver is determined (measured by the error reduction in the solver). Here, we use the adaptive approach described in Section 5 and Figure 5.1, where a single prototype is iteratively improved until the resulting solver is deemed to perform well.

We consider four PDEs as test problems, all discretized using bilinear finite elements on the unit square. Problem 1 is Laplace's Equation with pure Dirichlet boundary conditions. Problem 2 is Laplace's Equation with pure Neumann boundary

conditions. Problems 3 and 4 are discretizations of

$$-\nabla \cdot \mathcal{K}(\mathbf{x}) \nabla p(\mathbf{x}) = 0,$$

with Dirichlet boundary conditions on the East and West boundaries and Neumann boundary conditions along the North and South boundaries. For Problem 3, $\mathcal{K}(\mathbf{x})$ is chosen as

$$\mathcal{K}(\mathbf{x}) = \begin{cases} 10^{-8} & \mathbf{x} \in [\frac{1}{3}, \frac{2}{3}]^2, \\ 1 & \text{otherwise.} \end{cases}$$

For Problem 4, $\mathcal{K}(\mathbf{x})$ is assumed to be constant on each element and chosen to have value 10^{-8} on 20% of the elements (chosen randomly) and value 1 everywhere else.

A significant advantage of the αAMG method is its invariance to diagonal scaling, as shown in Section 4.2. Thus, for each of these problems, we consider the results of such scaling. A common scaling is to make the diagonal entries of A all have value 1, using the diagonal matrix given by $s_{ii} = \frac{1}{\sqrt{a_{ii}}}$. For Problems 1-4 above, the problems with matrices thus scaled are referred to as Problems 1u-4u (where the u refers to the unit diagonal of the matrix). We also consider a more drastic scaling given by $s_{ii} = 10^{5r_i}$, where again r_i is chosen from a uniform distribution on $[0, 1]$ for each i. We call these Problems 1r-4r (where the r refers to the random scaling). Note that this scaling is dependent on the mesh size, and so it is expected that the overall work (measured relative to the cost of a fine-scale matrix-vector product), including both setup and solution phases, will grow as the mesh size is reduced.

6.1. AMG Benchmarks. A baseline for these problems is established by considering the performance of standard AMG under the same assumptions (primarily that coarsening is performed geometrically). Since we expect the scalings employed to destroy any sense of strong connection in the matrix coefficients, we consider here a "strong-connection-only" version of AMG, equivalent to setting $\theta = 0$ in the definitions of strong influence and dependence in Section 2. Wall-clock times and iteration counts are shown in Table 6.1, while convergence factors are shown in Table 6.2.

These results show that classical AMG interpolation gives a scalable solver for Problems 1, 2, and 3, coming directly from discretization. If, however, the discretization matrices are scaled, then there is a significant increase in the needed work for solution, especially noticeable as the problem grows (but also present with smaller fine grids). Improving on the results from these unscaled problems is difficult, so our aim should be to determine a balance where significant improvement on the results from the scaled problems is found, while not excessively increasing the cost of solution for problems such as 1,2, and 3. The results for the unscaled Problem 4 are typical of the situation where, as h decreases, the problem becomes more difficult to solve due to the increase in the number of internal material interfaces.

6.2. Calibrated AMG results. For the adaptive AMG methodology, we consider several questions around the same problems. Experience has shown that a single setup cycle is often most efficient, if parameters ν_0 and ν_1 can be chosen such that this cycle yields an effective solver. How to best choose these parameters depends on our interests. For solving the matrix equation, $A\mathbf{x} = \mathbf{b}$, for a single vector, $\mathbf{b}$, we would like to choose ν_0 and ν_1 such that the total time to solution is smallest. This may mean sacrificing performance of the solver to save cost in the setup stage. For solving the matrix equation for many right sides, the parameters should be chosen such that

	64×64	128×128	256×256	512×512	1024×1024
Problem 1	0.04 (9)	0.22 (9)	0.91 (9)	3.32 (9)	13.13 (9)
Problem 1u	0.05 (9)	0.24 (9)	0.86 (9)	3.33 (9)	13.15 (9)
Problem 1r	> 200	> 200	> 200	> 200	> 200
Problem 2	0.04 (8)	0.20 (8)	0.81 (8)	3.12 (8)	12.30 (8)
Problem 2u	0.09 (27)	0.93 (52)	5.64 (90)	36.94 (158)	> 200
Problem 2r	> 200	>200	> 200	> 200	> 200
Problem 3	0.04 (9)	0.25 (9)	0.89 (9)	3.41 (9)	13.23 (9)
Problem 3u	0.11 (26)	0.78 (41)	4.73 (69)	29.81 (124)	> 200
Problem 3r	> 200	> 200	> 200	> 200	> 200
Problem 4	0.05 (12)	0.28 (12)	1.04 (11)	4.40 (13)	17.64 (14)
Problem 4u	0.18 (64)	3.14 (179)	> 200	> 200	> 200
Problem 4r	> 200	> 200	> 200	> 200	> 200

TABLE 6.1

Standard AMG: *Wall-Clock Time in Seconds (and Iteration Count) to Reduce Residuals by* 10^{10}. *Results marked "> 200" indicate that the residual convergence criterion was not met after 200 iterations.*

	64×64	128×128	256×256	512×512	1024×1024
Problem 1	0.104	0.115	0.124	0.131	0.137
Problem 1u	0.104	0.115	0.124	0.131	0.137
Problem 1r	0.991	0.997	0.996	0.996	0.996
Problem 2	0.068	0.069	0.070	0.071	0.071
Problem 2u	0.594	0.754	0.864	0.929	0.964
Problem 2r	0.991	0.993	0.993	0.993	0.992
Problem 3	0.111	0.122	0.130	0.136	0.141
Problem 3u	0.488	0.656	0.793	0.886	0.940
Problem 3r	0.995	0.997	0.996	0.996	0.996
Problem 4	0.209	0.212	0.233	0.290	0.375
Problem 4u	0.760	0.914	0.976	0.994	0.998
Problem 4r	0.996	0.996	0.996	0.996	0.995

TABLE 6.2

Standard AMG: *Asymptotic Convergence Factors (Measured After at Most 200 Iterations)*

the solver performs optimally. Here, we primarily consider the latter situation, and demonstrate that doing so does not severely impact the time to solution for a single right side.

We first present what might best be described as a calibrated AMG approach, reflecting that parameters ν_0 and ν_1 are chosen to calibrate the AMG performance, rather than a truly adaptive approach, where ν_0 and ν_1 remain fixed and setup is performed until an efficient solver is exposed. Thus, these results indicate the theoretical best performance of an adaptive AMG algorithm, with the minimal amount of total setup work, but do not represent a realistic adaptive algorithm, where adaptation is performed until a termination criterion is met. Such an algorithm is presented in the following section.

Our experiments were performed with the goal of (approximately) minimizing the asymptotic convergence factors of the resulting methods. To do this, we computed the asymptotic convergence factors for very large ν_0 and ν_1, corresponding to the

	64×64	128×128	256×256	512×512	1024×1024
Problem 1	0.02 (2,2)	0.09 (2,2)	0.38 (3,3)	1.67 (4,5)	7.21 (7,7)
Problem 1u	0.02 (2,2)	0.09 (2,2)	0.37 (3,3)	1.63 (4,5)	7.27 (7,7)
Problem 1r	0.02 (4,4)	0.11 (5,5)	0.47 (8,7)	2.13 (11,11)	9.52 (16,17)
Problem 2	0.02 (3,2)	0.10 (4,4)	0.42 (6,6)	1.88 (9,9)	8.57 (13,13)
Problem 2u	0.04 (3,2)	0.09 (4,4)	0.43 (7,6)	1.83 (10,9)	8.03 (14,13)
Problem 2r	0.03 (7,7)	0.11 (9,9)	0.51 (14,14)	2.50 (21,21)	11.84 (31,31)
Problem 3	0.02 (2,2)	0.11 (4,4)	0.45 (4,4)	1.69 (6,6)	6.86 (7,7)
Problem 3u	0.02 (2,2)	0.10 (4,4)	0.41 (4,4)	1.66 (6,6)	6.94 (7,7)
Problem 3r	0.03 (5,5)	0.10 (6,6)	0.45 (9,8)	1.90 (10,11)	8.50 (17,16)
Problem 4	0.02 (2,2)	0.06 (3,2)	0.35 (4,4)	1.60 (6,6)	6.80 (8,8)
Problem 4u	0.02 (2,2)	0.09 (3,2)	0.39 (5,5)	1.60 (6,6)	6.98 (9,9)
Problem 4r	0.02 (6,5)	0.12 (9,9)	0.50 (13,14)	2.37 (22,21)	17.09(22,20*)

TABLE 6.3

Calibrated AMG: *Wall-Clock Time in Seconds (and Values of ν_0, ν_1) for Setup Phase. * Indicates 2 setup cycles were more efficient*

optimal case where $\mathbf{x}^{(1)}$ is the slowest-to-converge mode of relaxation, then chose the smallest values for ν_0 and ν_1 such that the convergence factor of the resulting method was within 0.005 of the optimal factor when that factor was less than 0.1, and within 0.01 of the optimal factor otherwise. We found that, for Problems 1, 2, and 3, good asymptotic convergence factors could be achieved with relatively small values of ν_0 and ν_1. For Problem 4, the same performance as standard AMG on the unscaled system can be recovered with small values of ν_0 and ν_1. Table 6.3 shows the time required for the setup phase for given values of these parameters and different grid sizes. As always happens when measuring computational performance, the timings are accurate only to within a few hundredths of a second, and so there is some variation in the timings of program stages that require the same number and ordering of operations.

Table 6.4 presents the asymptotic convergence factors for the methods resulting from the setup stages as outlined in Table 6.3. Note that, for the first three problems and for all grid sizes, αAMG achieves convergence factors bounded well below 1, with very small growth as the mesh size decreases. For Problem 4, we see growth like that in the standard AMG results. Note also that scaling the matrices has no effect on our ability to determine an efficient solver for these problems.

Finally, in Table 6.5, we consider the total cost of solving $A\mathbf{x} = \mathbf{0}$ a single time, with random initial guess, using the near-optimal solver. Calibrated AMG did not (and could not be expected to) beat the overall performance of standard AMG on the four unscaled problems. However, the setup costs of calibrated AMG were not significantly higher than those of AMG. On a 1024×1024 mesh, AMG setup required approximately 5 seconds of CPU, and so the adaptive setup needed between 30% and 240% more time, but tended to produce a better solver than classical AMG. For these reasons, the overall cost of calibrated AMG is close to that of standard AMG in the cases where standard AMG works well.

When standard AMG fails, there is no contest. Calibrated AMG was able to solve those problems that caused difficulty for standard AMG in a small fraction of the time. For the first three randomly scaled problems, the calibrated AMG setup time requires at most 12 seconds on the 1024×1024 grid, equivalent in cost to approximately 15 AMG V-cycles. Yet, the resulting convergence factors of at most 0.108 are much lower

	64×64	128×128	256×256	512×512	1024×1024
Problem 1	0.067	0.073	0.079	0.080	0.079
Problem 1u	0.067	0.073	0.079	0.080	0.079
Problem 1r	0.069	0.078	0.077	0.078	0.079
Problem 2	0.069	0.069	0.071	0.071	0.073
Problem 2u	0.069	0.071	0.071	0.071	0.072
Problem 2r	0.072	0.071	0.071	0.072	0.073
Problem 3	0.070	0.097	0.081	0.110	0.103
Problem 3u	0.072	0.097	0.080	0.109	0.106
Problem 3r	0.070	0.100	0.084	0.111	0.108
Problem 4	0.194	0.202	0.243	0.288	0.376
Problem 4u	0.189	0.212	0.231	0.294	0.374
Problem 4r	0.187	0.212	0.235	0.292	0.383

TABLE 6.4
Calibrated AMG: *Asymptotic Convergence Factors.*

	64×64	128×128	256×256	512×512	1024×1024
Problem 1	0.04 (8)	0.25 (8)	0.89 (8)	3.52 (8)	14.70 (8)
Problem 1u	0.06 (8)	0.21 (8)	0.90 (8)	3.55 (8)	14.73 (8)
Problem 1r	0.03 (7)	0.22 (7)	0.91 (7)	3.64 (7)	15.64 (7)
Problem 2	0.05 (8)	0.22 (8)	0.92 (8)	3.83 (8)	15.85 (8)
Problem 2u	0.04 (8)	0.24 (8)	0.95 (8)	3.83 (8)	15.94 (8)
Problem 2r	0.05 (7)	0.24 (7)	0.97 (7)	4.23 (7)	18.60 (7)
Problem 3	0.05 (8)	0.20 (8)	0.93 (8)	3.63 (8)	14.58 (8)
Problem 3u	0.03 (8)	0.21 (8)	0.92 (8)	3.66 (8)	14.43 (8)
Problem 3r	0.04 (7)	0.24 (8)	0.94 (8)	3.81 (8)	16.32 (8)
Problem 4	0.03 (11)	0.31 (11)	1.09 (11)	4.84 (13)	22.06 (16)
Problem 4u	0.04 (11)	0.32 (11)	1.10 (11)	4.79 (13)	20.40 (14)
Problem 4r	0.05 (10)	0.27 (10)	1.22 (11)	5.35 (12)	28.27 (12)

TABLE 6.5
Calibrated AMG*: Total Solution Wall-Clock Time in Seconds (and Iteration Count) to Reduce Residuals by* 10^{10}.

than the AMG convergence factors, the smallest of which was 0.992. Thus, after setup and a single cycle of the calibrated AMG approach, the error will be smaller than if the equivalent amount of work, 16 classical AMG cycles, was performed, which would reduce the error by a factor of approximately 0.879. Even for the fourth randomly scaled problem, where the setup cost is equivalent to approximately 20 AMG V-cycles, the error reduction after a single cycle will be significantly better than the equivalent in cost of 21 classical AMG cycles, which would yield a total reduction of approximately 0.900.

6.3. Adaptive AMG (αAMG) results. While the calibrated AMG approach yields solvers with optimal performance characteristics, its results are achieved at a significant cost of user time in tuning the setup parameters of ν_0 and ν_1. Details of this tuning process are documented in [20]. In practice, an adaptive approach may be used instead, moving the burden of calibration from the user to the solver itself.

In the αAMG approach, a small, fixed number of relaxation sweeps are initially performed on the finest grid to locally expose algebraically smooth error. This error

is used to create an interpolation operator and (through the Galerkin conditions) a coarse-grid problem. Relaxation is then performed on this grid and the problem is further coarsened. After the initial multigrid hierarchy is created in this fashion, another identical setup cycle is performed with the current prototype as an initial guess. This is done to further improve the prototype as a representative of algebraically smooth error.

Alternately, we could look for a representative of the error missed by the multigrid cycle that results from the first setup cycle and use a prototype of this error to further improve interpolation and generate a more robust solver. Here, however, we consider problems similar to those for which classical AMG is effective. In particular, we consider those problems where the near null space can be represented by a single vector. The question of how to improve an existing AMG interpolation operator with information from a second prototype vector is not easily addressed. Instead, here we redefine the interpolation operator based on improving the prototype vector and not multiple distinct algebraically smooth components.

The results in Tables 6.6, 6.7, and 6.8 reflect performing 8 iterations of Gauss-Seidel relaxation on the homogeneous problem at each level of the setup phase. After each setup cycle, the resulting solver is tested by running 8 iterations of the solver on the homogeneous problem with a random initial guess. The fine-grid multigrid cycle is accepted when the error reduction factor (measured in the A-norm) of the last of these iterations is less than 0.4. Performing fewer iterations of relaxation on each level in the setup phase results in less exposure of slow-to-converge error and typically increases the number of setup cycles needed. Considering the high cost in recomputation of the interpolation and coarse-grid operators, we prefer to avoid this. Performing more iterations of relaxation on the homogeneous problem on each level better exposes the error sought, but at the cost of potentially significant unnecessary computation. Another additional cost in the adaptive setup phase is that of the "test drive" that performs 8 iterations of the current V-cycle on the homogeneous problem (to ensure adequate performance) after each setup cycle. Here, fewer iterations may give a poor indication of asymptotic cycle performance, while the cost of more iterations to better determine the true cycle performance is wasted if the cycle is to be further improved by another setup phase. Choosing the threshold for accepting the solver is less ambiguous, as setting a high threshold reduces the setup time but admits poor solvers, whereas setting the threshold to be small improves the solver but potentially requires more setup cycles.

Table 6.6 demonstrates the typical higher computational cost of the adaptive approach as compared to the calibrated approach. Some of this cost is certainly unavoidable. Fixing the number of iterations performed on the homogeneous problem undoubtedly results in some extraneous relaxations being performed, particularly on coarser grids, where few relaxation sweeps are "necessary". Likewise, the cost of testing the solver can only be avoided if we know, a priori, whether the solver will be acceptable. While these costs are significantly higher than those for calibrated AMG (as in Table 6.3), they do remain relatively low, still under one minute of CPU time for a 1024×1024 grid in all but the most difficult case.

Convergence factors for the adaptive AMG approach (Table 6.7) demonstrate its robustness. While the requirement for a successful setup was a single-step convergence factor less than 0.4 when measured after 8 iterations, we see that the asymptotic convergence factors are generally bounded by 0.4 as well. The asymptotic performance of the cycle for Problem 3r is relatively poor, but, in our tests, the first 15 steps of the

	64×64	128×128	256×256	512×512	1024×1024
Problem 1	0.03 (1)	0.19 (1)	0.79 (1)	3.31 (1)	13.33 (1)
Problem 1u	0.04 (1)	0.20 (1)	0.82 (1)	3.25 (1)	13.41 (1)
Problem 1r	0.04 (1)	0.19 (1)	0.83 (1)	6.30 (2)	25.70 (2)
Problem 2	0.04 (1)	0.20 (1)	0.89 (1)	3.35 (1)	13.59 (1)
Problem 2u	0.04 (1)	0.20 (1)	0.83 (1)	3.37 (1)	13.48 (1)
Problem 2r	0.04 (1)	0.20 (1)	0.84 (1)	6.44 (2)	51.62 (4)
Problem 3	0.04 (1)	0.20 (1)	0.82 (1)	3.36 (1)	13.20 (1)
Problem 3u	0.04 (1)	0.19 (1)	0.82 (1)	3.33 (1)	13.26 (1)
Problem 3r	0.04 (1)	0.21 (1)	0.82 (1)	9.53 (3)	38.57 (3)
Problem 4	0.04 (1)	0.20 (1)	0.84 (1)	3.25 (1)	25.81 (2)
Problem 4u	0.04 (1)	0.19 (1)	0.82 (1)	6.36 (2)	25.67 (2)
Problem 4r	0.04 (1)	0.39 (2)	1.56 (2)	31.06 (10)	229.11 (18)

TABLE 6.6
Adaptive AMG: *Wall-Clock Time in Seconds (and Number of Setup Cycles) for Setup Phase*

	64×64	128×128	256×256	512×512	1024×1024
Problem 1	0.065	0.069	0.070	0.086	0.201
Problem 1u	0.065	0.069	0.070	0.086	0.201
Problem 1r	0.068	0.085	0.210	0.071	0.071
Problem 2	0.067	0.069	0.089	0.156	0.335
Problem 2u	0.068	0.069	0.091	0.159	0.338
Problem 2r	0.099	0.160	0.355	0.075	0.338
Problem 3	0.067	0.097	0.080	0.118	0.294
Problem 3u	0.068	0.097	0.080	0.121	0.298
Problem 3r	0.075	0.113	0.293	0.110	0.867
Problem 4	0.186	0.195	0.243	0.395	0.384
Problem 4u	0.185	0.195	0.231	0.282	0.382
Problem 4r	0.227	0.202	0.235	0.282	0.385

TABLE 6.7
Adaptive AMG: *Asymptotic Convergence Factors.*

resulting method reduced the residuals by factors better than 0.3 at each step. This example, however, does indicate that a more reliable performance indicator would be of use. Note that some of the factors are worse than the corresponding convergence factors for the calibrated AMG approach; they could also be improved at the cost of more expensive setup cycles. A constant convergence factor of 0.4 is still sufficient, however, to reduce the error by a factor of 10^{10} in 25 iterations.

The overall time to solution of the adaptive approach, as in Table 6.8, reflects the expected behavior. Iteration counts are low - at most 18 iterations are required for residual reduction by a factor of 10^{10} - and so the time to solution, ignoring the setup phase, is quite low. The increased setup costs due to multiple adaptations are reflected in the total time to solution. Comparing these results to the calibrated AMG approach is somewhat disappointing; however, in all cases, the solution was still obtained quickly, with total time for the αAMG approach within a factor of at most 8.5 (and often only 2) of the total time for the calibrated AMG results. Comparing with the standard AMG results shows that the overall time to solution for the unscaled problems is higher than that for standard AMG, whereas, for the scaled problems,

	64×64	128×128	256×256	512×512	1024×1024
Problem 1	0.06 (8)	0.32 (8)	1.29 (8)	5.15 (8)	23.48 (11)
Problem 1u	0.06 (8)	0.32 (8)	1.29 (8)	5.13 (8)	23.58 (11)
Problem 1r	0.06 (7)	0.30 (7)	1.41 (10)	8.14 (7)	32.38 (7)
Problem 2	0.06 (8)	0.35 (8)	1.37 (8)	5.84 (9)	26.54 (13)
Problem 2u	0.06 (8)	0.34 (8)	1.34 (8)	5.67 (9)	24.58 (11)
Problem 2r	0.07 (8)	0.33 (8)	1.64 (13)	8.33 (7)	64.48 (12)
Problem 3	0.06 (8)	0.33 (8)	1.31 (8)	5.30 (8)	25.92 (13)
Problem 3u	0.06 (8)	0.34 (8)	1.33 (8)	5.31 (8)	25.83 (13)
Problem 3r	0.06 (7)	0.33 (8)	1.61 (13)	11.37 (7)	46.88 (8)
Problem 4	0.07 (11)	0.38 (11)	1.55 (11)	7.69 (18)	38.63 (13)
Problem 4u	0.07 (11)	0.37 (10)	1.53 (11)	9.22 (11)	38.67 (13)
Problem 4r	0.07 (10)	0.55 (10)	2.20 (10)	34.72 (11)	241.56 (13)

TABLE 6.8

Adaptive AMG: *Total Solution Wall-Clock Time in Seconds (and Iteration Count) to Reduce Residuals by* 10^{10}.

adaptive AMG always yields a quickly converging solver and, so, yields lower overall solution times in all cases where standard AMG performance suffered.

The calibrated and adaptive AMG results are very encouraging. For the scaled problems, there is a tremendous improvement in the amount of effort required for solution of the linear systems when compared to the results for standard AMG interpolation in Table 6.2. The solution phase of the algorithm scales well across all 5 grid sizes, and the actual costs are quite reasonable. It must be acknowledged, however, that once we account for the cost of the setup phase of our algorithm, classical AMG is still a slightly more efficient solver for the unscaled matrices when we consider solving with only a single right-hand side. Put simply, if the algebraically smoothest component of an elliptic PDE is known exactly, αAMG can do no better than designing multigrid interpolation based on that component. Indeed, if this component is given as input to the adaptive AMG method for creating the multigrid hierarchy, we can solve the problem with the same cost as classical AMG on the unscaled problem, simply by using this prototype as $\mathbf{x}^{(1)}$ in Equation 4.5, as discussed in Section 4.2.

7. Conclusions. The adaptive multigrid strategies outlined here provide an opportunity for the recovery of classical multigrid performance in cases where the near null space components of the matrix are not locally constant. This can be done in both the case of a known non-constant near null space component or of an unknown near null space component. The interpolation presented is a generalization of the classical Ruge-Stüben scheme and, when coupled with the multilevel prototype of the near null space it provides a scalable algorithm for matrix systems that the classical interpolation does not.

REFERENCES

[1] M. ADAMS, M. BREZINA, J. HU, AND R. TUMINARO, *Parallel multigrid smoothing: polynomial versus Gauss-Seidel*, J. Comput. Phys., 188 (2003), pp. 593–610.

[2] R. E. ALCOUFFE, A. BRANDT, J. E. DENDY, AND J. W. PAINTER, *The multigrid method for the diffusion equation with strongly discontinuous coefficients*, SIAM J. Sci. Stat. Comput., 2 (1981), pp. 430–454.

[3] P. Bochev, C. Garasi, J. Hu, A. Robinson, and R. Tuminaro, *An improved algebraic multigrid method for solving Maxwell's equations*, Siam. J. Sci. Comput., 25 (2003), pp. 623–642.
[4] J. H. Bramble, *Multigrid Methods*, vol. 294 of Pitman Research Notes in Mathematical Sciences, Longman Scientific & Technical, Essex, England, 1993.
[5] A. Brandt, *Multi–level adaptive solutions to boundary–value problems*, Math. Comp., 31 (1977), pp. 333–390.
[6] ———, *Algebraic multigrid theory: The symmetric case*, Appl. Math. Comput., 19 (1986), pp. 23–56.
[7] ———, *General highly accurate algebraic coarsening*, Elect. Trans. Numer. Anal., 10 (2000), pp. 1–20.
[8] 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, 1984.
[9] A. Brandt and D. Ron, *Multigrid solvers and multilevel optimization strategies*, in Multilevel Optimization in VLSICAD, J. Cong and J. Shinnerl, eds., vol. 14 of Comb. Optim., Kluwer Academic Publishers, Dordrecht, The Netherlands, 2003, pp. 1–69.
[10] M. Brezina, A. Cleary, R. Falgout, V. Henson, J. Jones, T. Manteuffel, S. McCormick, and J. Ruge, *Algebraic multigrid based on element interpolation (AMGe)*, SIAM J. Sci. Comp., 22 (2000), pp. 1570–1592.
[11] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *Adaptive smoothed aggregation (αSA)*, SIAM J. Sci. Comp., 25 (2004), pp. 1896–1920.
[12] M. Brezina, C. I. Heberton, J. Mandel, and P. Vaněk, *An iterative method with convergence rate chosen a priori*, UCD/CCM Report 140, Center for Computational Mathematics, University of Colorado at Denver, February 1999. http://www-math.cudenver.edu/ccmreports/rep140.ps.gz.
[13] M. Brezina, C. Tong, and R. Becker, *Parallel algebraic multigrids for structural mechanics*, SIAM Journal of Scientific Computing, To appear (2004). Also available as LLNL technical report UCRL-JRNL-204167.
[14] T. Chartier, R. Falgout, V. Henson, J. Jones, T. Manteuffel, S. McCormick, J. Ruge, and P. Vassilevski, *Spectral AMGe (ρAMGe)*, SIAM J. Sci. Comp., (2003), pp. 1–26.
[15] R. D. Falgout, *A note on the relationship between adaptive AMG and PCG*, Tech. Rep. UCRL-TR-205838, Lawrence Livermore National Laboratory, August 2004.
[16] R. D. Falgout and P. S. Vassilevski, *On generalizing the AMG framework*, SIAM J. Numer. Anal., 42 (2004), pp. 1669–1693. Also available as LLNL technical report UCRL-JC-150807.
[17] J. Fish and V. Belsky, *Generalized aggregation multilevel solver*, International Journal for Numerical Methods in Engineering, 40 (1997), pp. 4341–4361.
[18] L. Hageman and R. Kellogg, *Estimating optimum overrelaxation parameters*, Math. Comp., 22 (1968), pp. 60–68.
[19] O. Livne, *Coarsening by compatible relaxation*, Num. Lin. Alg. Appl., 11 (2004), pp. 205–227.
[20] S. MacLachlan, *Improving Robustness in Multiscale Methods*, PhD thesis, Univerity of Colorado at Boulder, Boulder, Colorado, 2004.
[21] T. Manteuffel, *Adaptive procedure for estimating parameters for the nonsymmetric Tchebychev iteration*, Numer. Math., 31 (1978), pp. 183–208.
[22] S. F. McCormick and J. W. Ruge, *Multigrid methods for variational problems*, SIAM J. Numer. Anal., 19 (1982), pp. 924–929.
[23] ———, *Algebraic multigrid methods applied to problems in computational structural mechanics*, in State-of-the-Art Surveys on Computational Mechanics, ASME, New York, 1989, pp. 237–270.
[24] J. Ruge, *Algebraic multigrid (AMG) for geodetic survey problems*, in Prelimary Proc. Internat. Multigrid Conference, Fort Collins, CO, 1983, Institute for Computational Studies at Colorado State University.
[25] ———, *Algebraic multigrid applied to systems of partial differential equations*, in Proc. Int'l Multigrid Conf., S. McCormick, ed., Amsterdam, Apr. 1985, North Holland.
[26] ———, *Final report on amg02*, tech. rep., Gesellschaft für Mathematik und Darenverarbeitung, St. Augustin, 1985.
[27] J. W. Ruge and K. Stüben, *Algebraic multigrid (AMG)*, in Multigrid Methods, S. F. McCormick, ed., vol. 3 of Frontiers in Applied Mathematics, SIAM, Philadelphia, PA, 1987, pp. 73–130.
[28] K. Stüben, *An introduction to algebraic multigrid*, in Multigrid, U. Trottenberg, C. Oosterlee, and A. Schüller, eds., Academic Press, San Diego, CA, 2001, pp. 413–528.
[29] P. Vaněk, M. Brezina, and J. Mandel, *Convergence of algebraic multigrid based on smoothed aggregation*, Numer. Math., 88 (2001), pp. 559–579.

[30] P. Vaněk, J. Mandel, and M. Brezina, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, Computing, 56 (1996), pp. 179–196.
[31] C. Wagner and G. Wittum, *Adaptive filtering*, Numerische Mathematik, 78 (1997), pp. 305–328.
[32] ———, *Filtering decompositions with respect to adaptive test vectors*, in Multigrid Methods V, W. Hackbusch and G. Wittum, eds., vol. 3 of Lecture Notes in Computational Science and Engineering, Berlin, 1998, Springer, pp. 320–334. Proc. of the Fifth European Multigrid Conference held in Stuttgart, Germany, October 1–4, 1996.
[33] U. Yang, *On the use of relaxation parameters in hybrid smoothers*, Num. Lin. Alg. Appl., 11 (2004), pp. 155–172.

View publication stats