LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-JRNL-203501

# Adaptive Algebraic Multigrid Methods

M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, J. Ruge

April 13, 2004

SIAM Journal on Scientific Computing

# ADAPTIVE ALGEBRAIC MULTIGRID METHODS

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

**Abstract.** Our ability to simulate physical processes numerically 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 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. The principles which guide the adaptivity are highlighted, as well as 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. [5, 17]), the current level of interest and research activity is fairly recent. This dramatic increase is probably due mostly 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, [22, §A.8], or [1]–[24] and their cited references.

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" (see Section 2) to determine effective coarsening and/or relaxation processes. Solvers of this type typically assume some *defining* characteristic of "algebraic smoothness" that specifies error components that are not quickly eliminated by the relaxation that is being used. For example, in standard AMG (cf. [20]), all such components are assumed to vary slowly along so-called strong connections in the matrix, or, in standard smoothed aggregation (SA; cf. [24]), 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 these 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 or 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 ($Ax = 0$) to determine its performance and expose whatever

†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 cannot effectively handle. The resulting prototype errors that these tests produce are then used to improve the algebraic multigrid process.

The concept of using a multigrid algorithm to improve itself began with standard AMG [17], where interpolation was adjusted to fit vectors obtained by relaxation on the homogeneous problem. In [5], a variation of this idea was used for recovering typical AMG convergence for a badly scaled scalar elliptic problem. While the method there was very basic and used only one prototype, it contained many of the ingredients of the adaptive process developed here. These concepts were developed further in [16, 18, 19, 21]. The idea of fitting of eigenvectors corresponding to the smallest eigenvalues was advocated in [16] and [21], 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. Most of these ideas were later summarized in [16].

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 idea of operator-induced interpolation in [1]. 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 smooth components to ensure optimal performance. By integrating information regarding algebraically smooth vectors into the definition of interpolation, we develop a multigrid scheme that is hopefully optimal for elliptic problems where the discrete system is not necessarily given in terms of an M-matrix. This operator- and relaxation-induced interpolation approach can, if properly implemented, greatly enlarge the class of problems that admits optimal performance by a black-box multigrid technique.

We also introduce the idea of adaptivity into algebraic multigrid. While classical multigrid methods can be viewed as stationary iterative methods [2], the method presented here is dynamic. 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 [4, 6]. The work presented here is the application to AMG of the methods presented in [8].

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. These include the methods of [7, 9, 10, 13]. 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 method we describe below attempts to achieve the good convergence properties of the element-based methods without the overhead of element storage that they require.

We refer to the approach developed here as adaptive because it involves self-testing to expose slowly-converging error components and adaptation of the scheme's components to improve itself. The acronym $\alpha$MG 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 only use the defining characteristic of smoothness and must use an automatic algebraic process to determine additional characteristics that enable effective determination of the full MG algorithm. The additional acronyms $\alpha$AMG and $\alpha$SA, respectively, are used to refer to the specific AMG and SA versions of $\alpha$MG.

In the next section, we give a brief description of standard AMG to set the stage for the development of $\alpha$AMG 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 classical AMG algorithm begins with the *defining* property of algebraic smoothness that, on average, the residual is small after a few sweeps of relaxation: $(Ae)_i \approx 0$ for each point $i$. It then uses the assumed *additional* property that these smooth errors vary slowly along strong matrix connections, which is typical of naturally-formulated discrete elliptic systems. AMG proceeds to determine a coarse-level system whose solution, when interpolated back to the fine level, provides a good correction to such smooth fine-level errors. This is basically a task of choosing coarse variables and then determining interpolation based on the definition of smoothness and the additional, assumed characteristic. The aim is to approximate all algebraically smooth errors well by interpolation, and then use standard variational processes to define the other coarsening components.

To construct interpolation to approximate a general smooth error, $e$, we use the premise of a small residual (that, for example, $\|Ae\| \ll \|A\| \cdot \|e\|_A$) to conclude that

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

Now, suppose that a coarse set of points has been chosen that forms a subset of the fine degrees of freedom (DOFs). We do not discuss how this might properly be done here, but instead refer the reader to [21], for example. 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. Since $A$ is sparse, we also introduce "neighborhood" notation: $N_i = \{j : a_{ij} \neq 0\}$, $C_i = C \cap N_i$, 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 (2.2), this expression could be used to define interpolation because it gives 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 the 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 $a_{ij}$, $j \in C_i$, in some way to compensate for the inaccuracy this elimination introduces. The main assumption needed to collapse the stencil is that the values of smooth $e$ at $F_i$ points can be written in terms of its values at points in $C_i \cup \{i\}$:

$$e_k \approx \sum_{j\in C_i} w_{kj}e_j + w_{ki}e_i, \quad k \in F_i. \tag{2.3}$$

This expression could then be substituted into the latter sum in (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 (2.3) is a special form of interpolation from $C_i \cup \{i\}$ to $F_i$. This special interpolation formula is used in the stencil connections to determine the final interpolation formula used, so the overall process is sometimes called "twice-removed" or "iterated" interpolation.

In classical AMG algorithms, the $\{w_{kj}\}$ in (2.2) are determined by $\{a_{ij}\}$ and $\{a_{kj}\}$ based on the additional assumption that $e$ is constant along certain of these connections. In practice, the Ruge-Stüben algorithm [21] for collapsing $F-F$ connections relies on identifying them as being either strong or weak. We do not need the distinction for this purpose in the algorithm presented below. This sense of strength of connection is also important for determining the $C$ points in the classical AMG algorithm, although there are other possible ways to accomplish this selection process (especially notable being the recent work on compatible relaxation; cf. [12, 14]). 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 positive definite matrix, it is natural to define these operators by the so-called Galerkin conditions (cf. [21]):

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

These definitions arise from finite element 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 $\alpha$AMG algorithm are quite complex. We arrived upon them by careful consideration of basic principles of the methodology of an adaptive algorithm. For this reason, the 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 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,

$$Ax = b.$$

**3.1. Algorithm.** An efficient multigrid process for solving $Ax = b$ relies on the appropriate complementarity of relaxation and coarse-grid correction. Because of this, 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, and then choose interpolation to fit these vectors. Such an approach is, however, quite inefficient and a multiscale viewpoint proves more useful.

Suppose we know matrix $A$, but nothing more. Relaxation is then the only possible way to expose the algebraically smooth components we are interested in. With no further knowledge, however, there is no way of knowing how many distinct components are needed to achieve good results. Experience (and, in the case of SA, theory [23]) 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. To arrive at an optimal solver with a minimal amount of work, it thus seems necessary to start with a single vector and introduce new prototypes as the evolving method proves inadequate.

The situation is now 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, this can require a significant number of relaxations, and so we seek to expose algebraically smooth components through multiscale development. After just a few steps of relaxation on the homogeneous problem on the finest grid can quickly reduce a significant portion of a random initial guess, the remaining error can 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 multigrid 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. 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 perfectly complementary, and that there are distinct algebraically smooth components that are not being accounted for. 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 can sketch the adaptive procedure as

1. Let $k = 1$ and $x_1$ be a random vector. Define, for all grids $l$, the methods SOLVE$_l(x_l,b_l)$ to be $\nu$ relaxation sweeps on $A_l x_l = b_l$.
2. SOLVE$_k(x_k,0)$.
3. If not sufficiently coarsened, form interpolation and its coarse-grid operator. Let $x_{k+1} = (x_k)_c$ (that is, $x_k$ evaluated at the grid $k+1$ points) and $k = k+1$. Goto Step 2.
   Otherwise, continue.
4. While $k > 1$, let $k = k - 1$, interpolate the coarse-grid approximation, $x_k = Px_{k+1}$, and run SOLVE$_k(x_k,0)$.
5. Let $k = 1$ and $x_1$ be a random vector, Define, for all grids $l$, SOLVE$_l(x_l,b_l)$ to be $\nu$ current V-cycles on $A_l x_l = b_l$. If performance of SOLVE$_1(x_1,0)$ 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 on which it is based. Here we list the core ideas that

motivate and provide a foundation for the $\alpha$MG methods, with the primary focus on the $\alpha$AMG scheme. The pragmatic reader may prefer to defer reading this discussion until after Section 4.

**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, $u$, of $A$, interpolation must reconstruct $u$ to an accuracy proportional to its eigenvalue [3, 15].

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 relaxation process cannot effectively reduce. For example, classical AMG is usually developed based on properties of a polynomial iterative method such as the Richardson iteration:

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

It is easy to show that the error, $e = x - A^{-1}b$, converges slowly in the $A$-norm, $\|e\|_A = \sqrt{< Ae, e >}$, if and only if $e$ yields a small generalized Rayleigh quotient:

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

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.

Almost all algebraic methods, however, make *additional* assumptions about algebraically smooth error that allow them to capitalize on the special nature of algebraic smoothness that is assumed. For example, classical AMG rests on two 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, the vector $\mathbf{1}$ 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 $\alpha$AMG 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. As mentioned above, for example, classical AMG uses the prototype vector $\mathbf{1}$ to build its matrix-based interpolation coefficients (see Equation (2.1) and the discussion in Section 2). $\alpha$AMG differs in that it attempts to generate its 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 non-optimal preconditioned conjugate gradient method can be formulated as an $\alpha$MG method that uses the generated prototypes globally as representatives of slow-to-converge error components (here, the smoother is the preconditioner and the coarse-grid corrections are the Krylov subspace projections; see [11] for details). The point is that, in general, errors that are 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 is to achieve this approximation property by only using a small number, $O(1)$, of computed prototypes.

The use of a small number of prototypes to achieve approximation of a much larger space is a cornerstone of multigrid methods. It is essential that each prototype be used effectively as a representative of many components with similar local character. Recall, again, the use of the vector $\mathbf{1}$ in the classical AMG method, where $\mathbf{1}$ is used locally to define an interpolation whose range represents all smooth errors. This is analogous to how local basis functions are used in finite elements: piecewise polynomial basis functions are used locally as pieces of a global polynomial basis to represent 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, $Ax = 0$, is especially appropriate because it avoids trouble with machine representation when the approximation is very close to that solution. For our $\alpha$AMG scheme, we can test the current version of the algorithm on $Ax = 0$ 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 it uses the additional smoothness assumptions. 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 $Ax = 0$ when all you presumably get is $x = 0$? 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. However, 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 $\alpha$AMG scheme, we can choose to improve $x$ on the coarsest level by minimizing the generalized Rayleigh quotient, $RQ_A(x)$. This becomes less clear when there are several prototypes because of the need to keep them separate. It may be necessary to use a Ritz projection and perhaps a Rayleigh quotient involving the correction operator from the current method ($RQ_{BA}$, where $I - BA$ represents

the current error propagation matrix) or a more sophisticated measure [12]. In the scalar PDE case considered here, we have not yet found any need for improving the coarsest-level prototype, so this is not addressed further in what follows. The Rayleigh quotient can, however, be useful as a diagnostic tool in assessing how the adaptive process is behaving.

**Range of Interpolation.** The primary aim of coarsening in any multigrid process is to allow algebraically smooth error to be represented by the range of interpolation. For the adaptive process, this means approximating the prototypes as much as possible. When enough DOFs are used on the coarse level, it is possible to fit them exactly: $x = Px_c$ for some coarse-level $x_c$. In our $\alpha$AMG scheme, $x_c$ is chosen simply as $x$ restricted to the coarse points, and so $x = Px_c$ only in the case that $Ax = 0$.

The fine-level problem is coarsened so that the coarse-grid representation of a prototype is just as good as the fine-grid representation, but requires less effort to resolve. For this reason, we can improve our representation of the near-null space on the coarse grid, but only if the range of interpolation admits an algebraically smoother component than the prototype. By using the prototype locally, we ensure that this is possible. Thus, in the adaptive process, we overwrite each fine-level prototype by its coarse-level interpolant that is, in general, a better prototype.

**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 it 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 approach assumes that a suitable coarsening process is available (such as the compatible relaxation in [12, 14]), and with the attendant assumption that there is sufficient reduction in grid sizes from fine to coarse level. When fitting multiple prototypes, however, it is tempting to abandon the tight control on the stencil of interpolation (to that of $A_{fc}$, the submatrix of $A$ linking fine- to coarse-grid nodes) 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 $C_i$ for each fine-grid point $i$ to the number of prototypes being fit. This 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 controlling 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. This is seen in $\alpha$SA, where the prototype set is locally orthogonalized when determining interpolation to ensure new columns of the prolongator (and thus coarse-grid points) are not introduced unnecessarily [8]. For this reason, it is more efficient to consider improvement of the existing candidate(s) than to add a new candidate. As prototypes emerge, we can consider improvement of their representation by removing each in turn from the prototype set, constructing the multigrid method based on this reduced set, and then applying

the reduced multigrid method to the removed prototype. This either improves the prototype as a representative of the smooth components that is not well-represented by the rest of the prototype set, or signals that the removed prototype is not needed in the set and the reduced multigrid algorithm can replace the previous one. In either case, the prototype set is improved, either by reducing its size or enhancing its representation of algebraically smooth error.

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 full approximation scheme manner. This means that the prototype on a given level is discarded when it can be improved from a coarser grid. 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 will 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 V-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 use the heuristic of allowing only a single setup cycle and choosing the number of relaxations in this cycle to 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 multigrid method is to move away from the weaknesses of classical AMG schemes. Thus, we first concern ourselves with generalizing 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 $Ax = b$ on error components, $e$, whose residuals, $Ae$, are small relative to $e$ in some sense; and
- efficient multigrid performance depends on effective complementarity of relaxation and coarsening to cooperate to eliminate all error components.

In developing the 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. The numerical results presented in Section 6 are from this implementation 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, $e$, that is not quickly reduced by relaxation. Using a simple (point-wise) relaxation scheme, such as Gauss-Seidel, this

also means that $Ae \approx 0$, or

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

where, as in (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

$$e_k \approx \sum_{j\in C_i} w_{kj}e_j + w_{ki}e_i,$$

a general interpolation formula for point $i \in F$ is then:

$$e_i = -\sum_{j\in C_i} \left( \frac{a_{ij} + \sum_{k\in F_i} a_{ik}w_{kj}}{a_{ii} + \sum_{k\in F_i} a_{ik}w_{ki}} \right) e_j. \tag{4.1}$$

The $\alpha$AMG interpolation is different from that used in classical AMG [21] in that $\{w_{kj}\}$ are chosen to depend on both the entries in $A$ and a (computed) prototype, $x^{(1)}$, that represents many algebraically smooth components. How this prototype is computed is the subject of Section 5.

To be specific about the choice of $\{w_{kj}\}$, consider the idea of twice-removed interpolation [5]. 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)

$$e_k = \sum_{j\in C_i} w_{kj}e_j, \tag{4.2}$$

dropping the term $w_{ki}e_i$, 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, $x^{(1)}$, that we want to be in the range of interpolation, then we ask that (4.2) hold when $e$ is replaced by $x^{(1)}$. This constraint with one vector, $x^{(1)}$, fixes one DOF of the possibly many for set $\{w_{kj}\}$, but this specification leads to a unique $F_i$-interpolation formula if it is taken to be of the form $-D^{-1}A_{fc}$, where $A_{fc}$ is the matrix of connections between $F$ and $C$, and $D$ is a diagonal matrix. (This choice is motivated by the discussion in [7].) $D$ is thus determined by

$$d_{kk}x_k^{(1)} = -\sum_{j\in C_i} a_{kj}x_j^{(1)}$$

or

$$d_{kk} = \frac{-\sum_{j\in C_i} a_{kj}x_j^{(1)}}{x_k^{(1)}}. \tag{4.3}$$

Thus, choosing $w_{kj} = d_{kk}^{-1}a_{kj}$ in (4.2), the $F_i$-interpolation formula is

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

Interpolation to $i \in F$, given by (4.1), then has the particular form

$$e_i = -\sum_{j \in C_i} \left( \frac{a_{ij} + \sum_{k \in F_i} a_{ik} \frac{a_{kj} x_k^{(1)}}{\sum_{j' \in C_i} a_{kj'} x_{j'}^{(1)}}}{a_{ii}} \right) e_j. \tag{4.4}$$

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 (4.4).

This $\alpha$AMG 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 [21]. Thus, classical AMG interpolation is recovered from the formula in (4.4) by choosing $x^{(1)} \equiv 1$.

The focus of this paper is on methods involving just one prototype vector, $x^{(1)}$, appropriate for scalar PDEs. Note, however, that 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, 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 $D$ ($D_{kk}$) become $d \times d$ matrices, and $x^{(1)}$ remains a single vector prototype. However, several more prototypes must generally be used, which are denoted by $x^{(2)}, \ldots, x^{(d)}$ and appended to $x^{(1)}$ to form the matrix $X^{(1)} = [x^{(1)}, \ldots, x^{(d)}]$. Its restriction to the $d$ DOFs at node $k$ is denoted by $X_k^{(1)}$. The analogue of (4.3) is then

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

The $F_i$-interpolation formula for systems thus becomes

$$e_k = -\sum_{j \in C_i} D_{kk}^{-1} A_{kj} e_j,$$

which yields the final nodal interpolation formula

$$e_i = -A_{ii}^{-1} \left( \sum_{j \in C_i} \left( A_{ij} + \sum_{k \in F_i} A_{ik} D_{kk}^{-1} A_{kj} \right) \right) e_j.$$

**4.2. Theoretical Properties.** One situation that can cause difficulty for classical AMG is when the matrix is simply 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 the left and the right (to preserve symmetry) by a positive diagonal matrix can dramatically change this property. Thus, if $A$ is replaced by $\tilde{A} = SAS$ for some positive diagonal matrix $S$, then $\tilde{A}(S^{-1}\mathbf{1}) \approx 0$, so 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 significant deterioration in convergence rates. This can be prevented if the scaling is supplied to AMG so that the original matrix can essentially be recovered (as in [5], but this is not always possible in practice. Fortunately, as the following result shows, this causes no problem for $\alpha$AMG, provided the scaled prototype can be accurately computed.

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

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

The weights, $\tilde{w}_{kj}$, for the matrix $\tilde{A}$ are given by

$$\begin{aligned} \tilde{w}_{kj} &= \frac{\tilde{a}_{kj}\tilde{x}_k^{(1)}}{\sum_{j' \in C_i} \tilde{a}_{kj'}\tilde{x}_{j'}^{(1)}} \\ &= \frac{s_k a_{kj} s_j s_k^{-1} x_k^{(1)}}{\sum_{j' \in C_i} s_k a_{kj'} s_{j'} s_{j'}^{-1} x_{j'}^{(1)}} = s_k^{-1} w_{kj} s_j. \end{aligned}$$

Equation 4.4 then gives (with some algebra)

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

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, $\tilde{P}$, is of the form

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

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

That is, 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 point-wise and block forms) are scaling invariant (that is, if $A$ is scaled to $SAS$ as above, initial guess $x^{(0)}$ to $S^{-1}x^{(0)}$ and initial right side $b$ to $Sb$, then the approximation generated changes from $x^{(1)}$ to $S^{-1}x^{(1)}$), 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 with nodal relaxation.*

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

**5. Determining $x^{(1)}$.** Successful implementation of these schemes for interpolation relies upon having an appropriate prototype vector, $x^{(1)}$ (or set of vectors, $X^{(1)}$). Since we rely on the complementarity of relaxation and coarsening, the best choice for $x^{(1)}$ 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., eigenvectors of symmetric $A$), relax on $Ax = b$ for some $b$, 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 it is possible that the slow-to-converge modes of the relaxation iteration, $I - BA$, are not modes for which $Ae \approx 0$, in most 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 inadequate or even unavailable. We thus concentrate on the case that a good prototype, $x^{(1)}$, is not known.

Start with a vector generated randomly from a uniform distribution on $(0, 1)$. The consideration of positive vectors is motivated by the case of scalar, second-order operators, which tend to 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 $\alpha$AMG breaks down if $x_i^{(1)} = 0$. 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, $Ax = 0$, produces 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, $x = 0$. Thus, starting with a random initial guess and performing relaxation on $Ax = 0$ generates a prototype vector, $x^{(1)}$, that represents the slow-to-converge components and that can then be used in the interpolation formula developed in Section 4.

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 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 minimal eigenvalue on the finest grid, requires significant 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

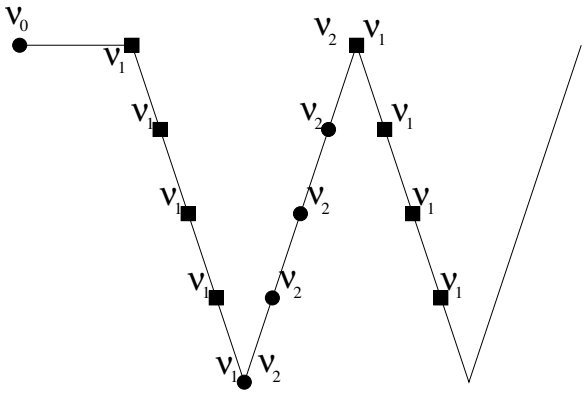


FIG. 5.1. *The setup scheme for determining* $x^{(1)}$

efficiently represented by calculation on these grids. Thus, we start with a random guess on the fine level and perform a few ($\nu_0$) relaxation sweeps there to generate a tentative $x^{(1)}$. Using this current prototype, an interpolation operator is computed (as in Section 4) and the coarse-level and restriction operators are formed using the Galerkin condition. We use injection (direct restriction of the values on the $C$-points) to form a coarse-level initial guess, relax $\nu_1$ times and recurse to the coarsest level. From this coarsest level, interpolate and relax $\nu_2$ times on the vector all the way to the finest level, but do not recompute the coarse-level and restriction operators. The cycle can then be repeated, using the resulting vector as an overall initial guess. 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 relaxation 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 apparently yields multigrid solvers with level-independent convergence factors tested up to $1024 \times 1024$ 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 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 and higher-order problems, an added wrinkle is the need to generate multiple prototype vectors. We expect that a technique similar to the one described above can generate the components well enough to produce an efficient multigrid scheme, but further investigation is necessary to ensure that the generated prototypes are rich enough and are not redundant. We expect that a strategy similar to the one developed for $\alpha$SA [8] can be used to produce an effective $\alpha$AMG approach for systems.

**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 a good initial indication of the performance of this approach can be had. 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 [14] 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 $Ax = b$. Solving a problem is defined here as reducing the residual by 10 orders of magnitude, and the wall-clock time to solution on a modern desktop workstation (2.66 GHz Pentium 4) is measured. Another relevant measure is the asymptotic convergence factor of the cycle that $\alpha$AMG produces. 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 the lower the factor the fewer iterations are required 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 has indicated that it is usually significantly more efficient to perform more relaxation sweeps (i.e., increase $\nu_0$ and $\nu_1$) in a single setup cycle than it is to perform multiple setup cycles with fewer iterations per cycle. Thus, in the results that follow, we perform only a single setup cycle and track the number of relaxations (values of $\nu_0$ and $\nu_1$) necessary to achieve highly efficient solver performance.

We consider four PDEs as test problems, all discretized using bilinear finite elements on the canonical 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

$$-\nabla \cdot D(x,y)\nabla p(x,y) = 0,$$

with Dirichlet boundary conditions on the East and West boundaries and Neumann boundary conditions along the North and South boundaries. For Problem 3, $D(x,y)$ is chosen as

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

For Problem 4, $D(x,y)$ 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 $\alpha$AMG 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 $s_{ii} = 10^{5r}$, where again $r$ 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).

A baseline for these problems is established by considering the performance of standard AMG under the same assumptions (primarily that coarsening is performed geometrically). Because 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. Wall-clock times and iteration counts are shown in Table 6.1, while convergence factors are shown in Table 6.2.

| | $64 \times 64$ | $128 \times 128$ | $256 \times 256$ | $512 \times 512$ | $1024 \times 1024$ |
|---|---|---|---|---|---|
| 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 | $3.8 \cdot 10^{-5}$ | $3.5 \cdot 10^{-5}$ | $2.5 \cdot 10^{-5}$ | $1.8 \cdot 10^{-5}$ | $1.3 \cdot 10^{-5}$ |
| Problem 2 | 0.04 (8) | 0.20 (8) | 0.81 (8) | 3.12 (8) | 12.30 (8) |
| Problem 2u | 0.10 (25) | 0.77 (41) | 4.58 (72) | 28.49 (123) | $5.8 \cdot 10^{-9}$ |
| Problem 2r | $4.3 \cdot 10^{-6}$ | $2.4 \cdot 10^{-6}$ | $3.8 \cdot 10^{-6}$ | $3.6 \cdot 10^{-6}$ | $3.3 \cdot 10^{-6}$ |
| Problem 3 | 0.05 (10) | 0.25 (9) | 0.90 (9) | 3.32 (9) | 13.18 (9) |
| Problem 3u | 0.08 (25) | 0.74 (38) | 4.51 (68) | 26.79 (114) | $3.5 \cdot 10^{-10}$ |
| Problem 3r | $5.3 \cdot 10^{-5}$ | $2.9 \cdot 10^{-5}$ | $1.8 \cdot 10^{-5}$ | $1.3 \cdot 10^{-5}$ | $9.9 \cdot 10^{-6}$ |
| Problem 4 | 0.05 (12) | 0.29 (12) | 1.07 (12) | 4.50 (14) | 24.71 (22) |
| Problem 4u | 0.17 (60) | 3.20 (178) | $2.0 \cdot 10^{-6}$ | $1.7 \cdot 10^{-5}$ | $1.1 \cdot 10^{-5}$ |
| Problem 4r | $3.9 \cdot 10^{-5}$ | $2.8 \cdot 10^{-5}$ | $1.8 \cdot 10^{-5}$ | $1.2 \cdot 10^{-5}$ | $9.2 \cdot 10^{-6}$ |

TABLE 6.1

*Wall-clock time in seconds (and iteration count) (or residual reduction after 200 iterations in case of failure) for standard AMG to reduce residuals by* $10^{-10}$.

| | $64 \times 64$ | $128 \times 128$ | $256 \times 256$ | $512 \times 512$ | $1024 \times 1024$ |
|---|---|---|---|---|---|
| Problem 1 | 0.106 | 0.115 | 0.121 | 0.126 | 0.132 |
| Problem 1u | 0.106 | 0.115 | 0.121 | 0.126 | 0.132 |
| Problem 1r | 0.991 | 0.996 | 0.996 | 0.996 | 0.996 |
| Problem 2 | 0.070 | 0.072 | 0.072 | 0.072 | 0.073 |
| Problem 2u | 0.559 | 0.723 | 0.842 | 0.916 | 0.957 |
| Problem 2r | 0.990 | 0.991 | 0.992 | 0.992 | 0.993 |
| Problem 3 | 0.128 | 0.129 | 0.133 | 0.138 | 0.143 |
| Problem 3u | 0.488 | 0.648 | 0.796 | 0.883 | 0.941 |
| Problem 3r | 0.995 | 0.997 | 0.996 | 0.996 | 0.996 |
| Problem 4 | 0.200 | 0.220 | 0.275 | 0.334 | 0.559 |
| Problem 4u | 0.757 | 0.914 | 0.976 | 0.994 | 0.998 |
| Problem 4r | 0.996 | 0.996 | 0.996 | 0.996 | 0.995 |

TABLE 6.2

*Asymptotic convergence factors (measured after at most 200 iterations) for standard AMG*

These results show that classical AMG interpolation gives a scalable solver for Problems 1, 2, and 3, coming directly from discretization. If, however, the discretiza-

tion matrices are scaled, then there is a significant increase in the needed work units for solution, especially as the problem grows (but also on the coarsest grids). Improving upon the results from these unscaled problems is difficult, so our aim should be to determine a balance where significantly improvement on the results from the scaled problems is found, while not driving up the cost of solution for problems such as 1,2, and 3. The results for the unscaled Problem 4 are typical of this situation where, as $h$ decreases, the problem becomes more difficult to solve due to the increase in internal boundary points.

For the adaptive AMG method, we consider several questions around the same problems. Since experience has shown that a single setup cycle is most efficient, parameters $\nu_0$ and $\nu_1$ must be chosen such that this cycle yields an effective solver. How to do so actually depends on our interests. For solving only the matrix equation $Ax = b$ for a single vector, $b$, choose $\nu_0$ and $\nu_1$ such that the total time to solution is smallest. This may mean sacrificing performance of the solver to save cost of the setup stage. For solving the matrix equation for many right sides, the parameters should be chosen such that the solver performs optimally. We consider this latter situation, and demonstrate that doing so does not severely impact the time-to-solution for a single right side.

Our experiments were thus performed with the goal of (approximately) minimizing the asymptotic convergence factors of the resulting methods. We found that, for Problems 1, 2, and 3, good asymptotic convergence factors could be achieved with relatively small values of $\nu_0$ and $\nu_1$. For Problem 4, the same performance as standard AMG on the unscaled system can be recovered with small values of $\nu_0$ and $\nu_1$. 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, $\alpha$AMG 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 affect on our ability to determine an efficient solver for these problems.

Finally, in Table 6.5, we consider the total cost of solving $Ax = 0$ a single time, with random initial guess, using the near-optimal solver. $\alpha$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 adaptive AMG were not significantly higher than those of AMG. On a $1024 \times 1024$ mesh, AMG setup required approximately 5 seconds of CPU, and so the adaptive setup needed between 30 and 120% more time, but tended to produce a slightly better solver than classical AMG. For these reasons, the overall cost of adaptive AMG is close to that of standard AMG in the cases where standard AMG works well. When standard AMG fails, there is no contest. Adaptive AMG was able to solve those problems that caused difficulty for standard AMG in a small fraction of the time. Considering that 200 iterations of standard AMG for a $1024 \times 1024$ problem took approximately 180 seconds, $\alpha$AMG was able to reduce the residual by over 4 orders of magnitude more in under a tenth of the time on the first three randomly scaled $1024 \times 1024$ grid problems, and in a sixth of the time for the fourth.

| | $64 \times 64$ | $128 \times 128$ | $256 \times 256$ | $512 \times 512$ | $1024 \times 1024$ |
|---|---|---|---|---|---|
| Problem 1 | 0.02 (2,2) | 0.09 (2,2) | 0.39 (3,3) | 1.58 (4,5) | 6.80 (7,7) |
| Problem 1u | 0.01 (2,2) | 0.10 (2,2) | 0.36 (3,3) | 1.51 (4,5) | 6.64 (7,7) |
| Problem 1r | 0.03 (4,3) | 0.10 (5,5) | 0.44 (8,7) | 1.91 (11,11) | 8.68 (16,17) |
| Problem 2 | 0.02 (2,2) | 0.08 (3,3) | 0.39 (5,5) | 1.74 (8,7) | 7.73 (12,11) |
| Problem 2u | 0.02 (3,2) | 0.08 (4,3) | 0.39 (5,5) | 1.69 (7,8) | 7.56 (12,11) |
| Problem 2r | 0.02 (5,6) | 0.12 (8,7) | 0.51 (12,11) | 2.21 (18,18) | 11.07 (28,28) |
| Problem 3 | 0.02 (2,2) | 0.10 (4,4) | 0.37 (4,4) | 1.59 (6,6) | 6.66 (7,7) |
| Problem 3u | 0.02 (2,2) | 0.07 (4,4) | 0.40 (4,4) | 1.59 (6,6) | 6.66 (7,7) |
| Problem 3r | 0.02 (5,4) | 0.12 (6,6) | 0.45 (8,7) | 1.89 (10,11) | 8.52 (16,16) |
| Problem 4 | 0.02 (2,2) | 0.09 (3,2) | 0.40 (5,4) | 1.64 (6,5) | 6.64 (6,6) |
| Problem 4u | 0.01 (3,2) | 0.10 (2,3) | 0.42 (4,4) | 1.65 (6,5) | 6.62 (6,6) |
| Problem 4r | 0.02 (6,5) | 0.12 (9,9) | 0.49 (13,12) | 2.30 (19,19) | 9.20 (21,20) |

TABLE 6.3
*Wall-clock time in seconds (and values of $\nu_1$, $\nu_2$) for adaptive AMG setup phase.*

| | $64 \times 64$ | $128 \times 128$ | $256 \times 256$ | $512 \times 512$ | $1024 \times 1024$ |
|---|---|---|---|---|---|
| Problem 1 | 0.068 | 0.081 | 0.078 | 0.080 | 0.079 |
| Problem 1u | 0.068 | 0.081 | 0.078 | 0.080 | 0.079 |
| Problem 1r | 0.080 | 0.072 | 0.077 | 0.078 | 0.080 |
| Problem 2 | 0.080 | 0.080 | 0.077 | 0.080 | 0.079 |
| Problem 2u | 0.070 | 0.075 | 0.078 | 0.080 | 0.080 |
| Problem 2r | 0.080 | 0.080 | 0.080 | 0.079 | 0.080 |
| Problem 3 | 0.076 | 0.098 | 0.098 | 0.109 | 0.111 |
| Problem 3u | 0.077 | 0.097 | 0.097 | 0.109 | 0.109 |
| Problem 3r | 0.078 | 0.099 | 0.097 | 0.113 | 0.113 |
| Problem 4 | 0.200 | 0.223 | 0.274 | 0.327 | 0.551 |
| Problem 4u | 0.201 | 0.225 | 0.275 | 0.324 | 0.549 |
| Problem 4r | 0.206 | 0.227 | 0.279 | 0.331 | 0.565 |

TABLE 6.4
*Asymptotic convergence factors for adaptive AMG.*

These results are very much as we hoped. For the scaled problems, there is a tremendous improvement in the amount of effort required for solution as compared to the standard AMG-interpolation based results 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 a slightly more efficient solver for the unscaled matrices when we consider solving for only a single vector. Put simply, if the algebraically smoothest component of an elliptic PDE is known exactly, $\alpha$AMG 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 it as $x^{(1)}$ in Equation 4.4, 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-

| | $64 \times 64$ | $128 \times 128$ | $256 \times 256$ | $512 \times 512$ | $1024 \times 1024$ |
|---|---|---|---|---|---|
| Problem 1 | 0.06 (8) | 0.23 (8) | 0.84 (8) | 3.30 (8) | 13.74 (8) |
| Problem 1u | 0.04 (8) | 0.20 (8) | 0.83 (8) | 3.35 (8) | 13.69 (8) |
| Problem 1r | 0.04 (7) | 0.22 (7) | 0.82 (7) | 3.33 (7) | 14.69 (7) |
| Problem 2 | 0.04 (8) | 0.24 (8) | 0.84 (8) | 3.40 (8) | 14.37 (8) |
| Problem 2u | 0.05 (8) | 0.26 (8) | 0.86 (8) | 3.47 (8) | 14.56 (8) |
| Problem 2r | 0.05 (7) | 0.23 (7) | 0.89 (7) | 3.74 (7) | 16.92 (7) |
| Problem 3 | 0.05 (8) | 0.23 (8) | 0.84 (8) | 3.39 (8) | 13.77 (8) |
| Problem 3u | 0.05 (8) | 0.23 (8) | 0.85 (8) | 3.54 (8) | 14.05 (8) |
| Problem 3r | 0.04 (8) | 0.25 (8) | 0.90 (8) | 3.65 (8) | 15.75 (8) |
| Problem 4 | 0.06 (11) | 0.31 (12) | 1.16 (12) | 4.82 (14) | 26.61 (23) |
| Problem 4u | 0.05 (11) | 0.29 (12) | 1.11 (12) | 4.79 (14) | 27.45 (24) |
| Problem 4r | 0.05 (11) | 0.33 (11) | 1.21 (12) | 5.28 (13) | 28.61 (22) |

TABLE 6.5

*Total solution wall-clock time in seconds (and iteration count) for adaptive AMG to reduce residuals by* $10^{-10}$.

null-space components of the matrix are not locally constant. This can be done in both the case of a known non-constant near-kernel component or of an unknown near-kernel 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] 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.

[2] J. H. BRAMBLE, *Multigrid Methods*, vol. 294 of Pitman Research Notes in Mathematical Sciences, Longman Scientific & Technical, Essex, England, 1993.

[3] A. BRANDT, *Algebraic multigrid theory: The symmetric case*, Appl. Math. Comput., 19 (1986), pp. 23–56.

[4] ———. a lecture given at CASC, Lawrence Livermore National Lab, June 2001.

[5] 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.

[6] A. BRANDT AND D. RON, *Multigrid solvers and multilevel optimization strategies*. manuscript, 2002.

[7] 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.

[8] M. BREZINA, R. FALGOUT, S. MACLACHLAN, T. MANTEUFFEL, S. MCCORMICK, AND J. RUGE, *Adaptive smoothed aggregation ($\alpha$SA)*, SIAM J. Sci. Comp., (2003). submitted.

[9] 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.

[10] T. CHARTIER, R. FALGOUT, V. HENSON, J. JONES, T. MANTEUFFEL, S. MCCORMICK, J. RUGE, AND P. VASSILEVSKI, *Spectral AMGe ($\rho$AMGe)*, SIAM J. Sci. Comp., (2003), pp. 1–26.

[11] R. FALGOUT, *A note on the relationship between adaptive AMG and PCG*, technical report, Lawrence Livermore National Laboratory, 2004.

[12] R. FALGOUT AND P. VASSILEVSKI, *On generalizing the AMG framework*, SIAM J. Numer. Anal., (2003). submitted.

[13] J. Fish and V. Belsky, *Generalized aggregation multilevel solver*, International Journal for Numerical Methods in Engineering, 40 (1997), pp. 4341–4361.
[14] O. Livne, *Coarsening by compatible relaxation*, Numer. Lin. Alg. App., (2003). submitted.
[15] S. F. McCormick and J. W. Ruge, *Multigrid methods for variational problems*, SIAM J. Numer. Anal., 19 (1982), pp. 924–929.
[16] ———, *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.
[17] 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.
[18] ———, *Algebraic multigrid applied to systems of partial differential equations*, in Proc. Int'l Multigrid Conf., S. McCormick, ed., Amsterdam, Apr. 1985, North Holland.
[19] ———, *Final report on amg02*, tech. rep., Gesellschaft für Mathematik und Darenverarbeitung, St. Augustin, 1985.
[20] J. W. Ruge and K. Stüben, *Efficient solution of finite difference and finite element equations by algebraic multigrid (AMG)*, in Multigrid Methods for Integral and Differential Equations, D. J. Paddon and H. Holstein, eds., The Institute of Mathematics and its Applications Conference Series, Clarendon Press, Oxford, 1985, pp. 169–212.
[21] ———, *Algebraic multigrid (AMG)*, in Multigrid Methods, S. F. McCormick, ed., vol. 3 of Frontiers in Applied Mathematics, SIAM, Philadelphia, PA, 1987, pp. 73–130.
[22] 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.
[23] P. Vaněk, M. Brezina, and J. Mandel, *Convergence of algebraic multigrid based on smoothed aggregation*, Numer. Math., 88 (2001), pp. 559–579.
[24] 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.

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