OPERATOR-BASED INTERPOLATION FOR BOOTSTRAP ALGEBRAIC MULTIGRID

T. MANTEUFFEL*, S. MCCORMICK*, M. PARK*, AND J. RUGE*

Abstract. Bootstrap Algebraic Multigrid (BAMG) is a multigrid-based solver for matrix equations of the form $Ax = b$. Its aim is to automatically determine the interpolation weights used in algebraic multigrid (AMG) by locally fitting a set of test vectors that have been relaxed as solutions to the corresponding homogeneous equation, $Ax = 0$, and are then possibly improved later using a multilevel eigensolver. This paper introduces a flexible variant of BAMG that determines the interpolation weights indirectly by "collapsing" the unwanted connections in "operator interpolation". Compared to BAMG, this *indirect* BAMG approach (iBAMG) is more in the spirit of classical AMG, which collapses unwanted connections in operator interpolation based on the (restrictive) assumption that smooth error is locally constant. This paper studies the numerical performance of iBAMG and establishes an equivalence, under certain assumptions, between it and a slightly modified (standard) BAMG scheme. To focus on camparing BAMG and iBAMG and exposing their behavior as the problem size grows, the numerical experiments concentrate on Poisson-like scalar problems.

Key words. iterative methods, multigrid, algebraic multigrid, adaptive multigrid

AMS subject classifications.

1. Introduction.

Multigrid methods consist of two complementary processes: *smoothing* and *coarse-grid correction*. In the classical geometric multigrid setting, the premise is that relaxation methods like Jacobi and Gauss-Seidel are effective at reducing high-frequency (oscillatory) error, but are often poor at reducing low-frequency (smooth) error. The aim, then, of the coarsening process is to eliminate the smooth error that remains after relaxation. Unfortunately, many problems cannot be easily treated by coarsening in a geometrically based way (e. g., those arising from discretizations based on highly irregular grids) and many more still do not exhibit the property that relaxation produces geometrically smooth error (e. g., highly anisotropic problems, stochastic problems like those that arise in quantum chromodynamics, and even problems whose unknowns are scaled in a way that is not available to the solver).

Classical *algebraic multigrid* (AMG [6]) is a linear solver for matrix equations of the form $Ax = b$ that is based on multigrid principles but that requires no knowledge of the underlying geometry. Instead, AMG attempts to use the concept of *algebraic smoothness* to choose the coarse "grids" and intergrid transfer operators automatically based on the matrix entries. Classical AMG is founded on two core premises: relaxation produces relatively small residuals and the errors associated with small residuals are locally constant. The first premise allows a coarsening process that is based on "operator interpolation" and the second premise, which constitutes algebraic smoothness, allows *unwanted* connections in the operator interpolation process to be "collapsed". (We avoid referring to the terms we intend to collapse in the usual way as "$F - F$" connections because, in addition to fine-grid points, we may want to eliminate coarse-grid points that are not used for interpolation.) See [8] for more detail. While appropriate use of the characteristics of algebraic smoothness seems essential for obtaining effective solvers, these additional assumptions limit the scope of applicability of such methods. What is needed are self-learning algebraic multigrid solvers that automatically determine the full character of algebraically smooth

*Department of Applied Mathematics, Campus Box 526, University of Colorado at Boulder, Boulder, CO 80309-0526. *email:* **{ tmanteuf, stevem, parkmh, jruge}@colorado.edu**

errors. Robust multigrid solvers with this capability could dramatically increasing the applicability of optimal multigrid solvers over a wide range of discrete problems.

Extensive research effort has already been devoted to the development of such self-learning algebraic multigrid algorithms, starting from the original *adaptive* AMG algorithm introduced in [6], the bootstrap AMG approach introduced in [3] and developed further for quantum chromodynamics in [5], an adaptive scheme based on smoothed aggregation (SA) developed in [9] and [10], and adaptive AMG schemes developed further in [12] and [11]. The principal difference between these self-learning multigrid schemes is that the adaptive approaches typically start with just one test vector, while bootstrap starts with several. Both schemes produce the test vectors initially by starting with random initial guesses to the corresponding homogeneous problem, $Ax = 0$. The adaptive approach constructs interpolation to fit its single initial test vector, then tests the resulting solver on the homogeneous problem, starting from another random initial vector. If observed convergence is not yet acceptable, then the resulting vector is either used to enhance the original test vector or else added to the test-vector set. The process then continues until acceptable convergence is observed. BAMG instead constructs interpolation to fit (in a least-squares sense) its several initial test vectors. It uses the constructed solver a little more reluctantly, and only after it observes slow improvement of interpolation based on other possible choices for enhancing the set of test vectors (e. g., interpolating eigenvector approximations from the coarsest grids). The adaptive schemes are advantageous in that they naturally sort out a rich and locally independent set of test vectors: a properly implemented adaptive scheme should be able to generate a local representation of algebraic smoothness that is rich and independent in the sense that each local vector represents a new character of algebraic smoothness. Unfortunately, adaptive approaches can be costly in their initial stages because an effective coarse-grid solver must be developed before returning to the fine grid if the quality of interpolation is to be safely assessed. Otherwise, it would be difficult to determine whether slow convergence is due to a poor interpolation operator or a poor coarse-level solver. Bootstrap methods are more problematic for sorting out local linearly independent test vectors, but their initial stages are potentially much less expensive because they tend to delay the need for an effective coarse-grid solver.

It is therefore compelling to study bootstrap methods further, particularly in terms of gaining some insight into how many test vectors should be used initially for a given class of problems. Our intention here is not to compare bootstrap and adaptive methods nor develop related hybrids, but rather to study the performance of the basic approach and to show performance dependence on problem size. We also do not intend to analyze cost or determine optimal parameters. Comparisons and optimization would necessarily be involved and extensive, and are in any case beyond the scope of this paper. Nevertheless, since the adaptive methods can in a loose sense be regarded as a bootstrap approach with one initial test vector, then the specific knowledge we gain here should facilitate comparison of the adaptive and bootstrap approaches. Ideally, this study would be done by developing strategies that allow for fewer test vectors than are currently used in bootstrap methods. That is, bootstrap methods have tended to use enough initial test vectors to yield well-determined local least-squares problems, which means at least as many as the number of local interpolatory points. One of the outcomes of our work here is an effective scheme that allows for very underdetermined least-squares problems, so this paper can be considered as taking us a step closer to comparing the adaptive and bootstrap

approaches.

With this goal in mind, we begin in the next section with a brief description of classical AMG to set the stage. Then, in section 3, we introduce an improved version of BAMG. This new scheme determines the interpolation weights indirectly by "collapsing" the unwanted connections in "operator interpolation". Compared to BAMG, this *indirect* BAMG approach (*i*BAMG) is more in the spirit of classical AMG, which collapses unwanted connections in operator interpolation based on the (restrictive) assumption that smooth error is locally constant. Continuing section 3, we describe two ways to collapse unwanted connections and prove that they are equivalent under certain assumptions. More importantly, we then show the equivalence between a slightly modified standard BAMG scheme and our *i*BAMG approach, again under certain assumptions. Finally, in section 4, we use numerical experiments to confirm that these are indeed improvements to standard BAMG, and end with a few concluding remarks in section 5.

We should point out here that our focus is on improving the process for choosing weights in bootstrap methods, not the other aspects of coarsening that have been developed in this context. Thus, in particular, we do not address the question of determining good coarse grids or interpolatory points. For example, our numerical tests all use standard coarse grids, which we know to be adequate for the simple Poisson-type problems we consider here. For recent work on how to choose effective coarse grids within the BAMG setting, see [5].

2. Classical AMG. Assume in what follows that $A = (a_{ij}) \in \Re^{n\times n}$. This section is devoted to describing how classical AMG applied to $Ax = b$ determines the weights in the interpolation operator, which we denote by P. Because our focus is on the weights of interpolation as opposed to how the coarse grids are selected, we assume that the fine-level points have already been partitioned into points that are identified with the coarse grid, the set of which we denote by C, and its complement, which we denote by F. This partition into C and F points gives rise to the AMG form of interpolation described as follows: the ith entry of Pe be given by

$$(Pe)_i = \begin{cases} e_i & \text{if } i \in C, \\ \sum\limits_{j\in C_i} w_{ij}e_j & \text{if } i \in F. \end{cases} \tag{2.1}$$

Here, C_i is the subset of C of points that are used to interpolate to point $i \notin C$. Our task now is to describe in abstract terms how the interpolation weights, w_{ij}, are determined. Therefore, for the remainder of this section, we consider a given fixed $i \in F$.

Interpolation only needs to approximate error that is not easily attenuated by relaxation. This observation gives rise to the first AMG premise: relaxation produces error, e, that is algebraically smooth in the sense that it exhibits a relatively small residual. Therefore, we can assume that $(Ae)_i \approx 0$, that is, that

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

Consider now the splitting of this sum into its component sums over C_i, the coarse interpolatory set, and C_i^c, the remaining grid points in the *neighborhood* of i, by which we mean the set of points that are connected to i in A (i. e., all points ℓ such that $a_{i\ell} \neq 0$). We assume for simplicity in this paper that C_i^c is taken only from the

neighborhood of i, so it consists of connected F points and other connected C points that are not in C_i. With this splitting, we obtain

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

The message here is that, if the second component sum happens to vanish in this approximation (e. g., $a_{ik} = 0$ for $k \in C_i^c$), then we would immediately have a formula that expresses the value of any algebraically smooth error at point i by its value at points of C_i. This "operator interpolation" formula would then yield appropriate weights for P given by $w_{ij} = -a_{ij}/a_{ii}$. This observation suggests, for the general case $\sum_{k\in i} a_{ik}e_k \neq 0$, that we need to "collapse" the unwanted connections (a_{ik} for $k \in C_i^c$) to C_i. Thus, we need to replace the e_k in the second sum on the right side of (2.2) with sums that involve only e_j for $j \in C_i$.

To replace each e_k, $k \in C_i^c$, with a linear combination of the e_j, $j \in C_i$, we need to make a further assumption about the nature of smooth error. Since the historical target for AMG is partial differential equations of elliptic type, the classical AMG premise is that smooth error is locally almost constant. This second AMG premise means that we can assume that each e_k is any convex linear combination of the e_j, $j \in C_i$, that preserves constants. AMG is based on the particular linear combination where the coefficients are proportional to the connections from point k to each point j, that is, it is based on the approximation

$$e_k \approx \sum_{j\in C_i} \frac{a_{kj}}{\sum\limits_{\ell\in C_i} a_{k\ell}} e_j. \tag{2.3}$$

Substituting this expression into (2.2) and dividing the result by a_{ii} yields interpolation weights given by

$$w_{ij} = \frac{1}{a_{ii}} \left(-a_{ij} - \sum_{k\in C_i^c} \left(a_{ik} \frac{a_{kj}}{\sum\limits_{\ell\in C_i} a_{k\ell}} \right) \right). \tag{2.4}$$

This process of collapsing the unwanted connections in the operator interpolation formula expressed by (2.2) can be viewed as using a crude but properly scaled *truncated* interpolation formula, expressed by (2.3), to interpolate from C_i to e_k. (We refer to (2.3) as truncated because it amounts to operator interpolation at point k where we have simply deleted the unwanted terms–those that do not belong to C_i. It is properly scaled in the sense that it is exact for constants.) This indirect process has the effect of collapsing the unwanted connections, and it leads to the direct formula for interpolation weights defined in (2.4).

3. BAMG and *i*BAMG methods. Assume for the remainder of this paper that $A \in \Re^{n\times n}$ is symmetric and positive definite. Suppose now that we are given a set of test vectors, $e^{(l)}$, $l = 1, 2, \ldots, q$, that result from several fine-level relaxation sweeps on the homogeneous equation, $Ae = 0$, starting from q distinct random approximations. (We assume that the initial random vectors are of unit length in the Euclidean norm to avoid significant scale disparity in the least-squares processes that follow.) Since the focus is on the process of determining interpolation weights, we continue to assume that the fine-level points have already been partitioned into the C-points that are identified with the coarse grid and its F-point complement set. We

also assume that the coarse-grid interpolatory set, C_i, has already been determined for each F-point i. Also, unless otherwise noted (in particular, see section 3.5), we assume that the vectors are locally rich in that they are locally independent and numerous enough to ensure that all of the least-squares problems we introduce are uniquely solvable.

3.1. BAMG. The general form of interpolation operator, P, for AMG is given in (2.1). As described in the previous section, the choice of weights w_{ij} is dictated by two basic premises: relaxation produces small residuals and relaxed errors are locally almost constant. The first premise is very general: many matrix equations can be treated by relaxation schemes that produce small residuals in a sense that leads to useable local operator interpolation formulas. However, many circumstances arise where local errors are not approximately constant in any local sense, so the second premise seriously restricts the applicability of AMG methods. The basic idea behind BAMG is to glean the local character of algebraically smooth errors from the set of test vectors. This leads to a determination of the interpolation weights by direct least-squares fit of the target vectors. Thus, for each $i \in F$, we compute

$$\text{(BAMG)} \qquad \{w_{ij} : j \in C_i\} = \arg\min_{w_{ij}} \sum_{l=1}^{q} (e_i^{(l)} - \sum_{j \in C_i} w_{ij} e_j^{(l)})^2. \tag{3.1}$$

BAMG takes a direct approach to determining the weights of interpolation. More in the spirit of classical AMG as described in section 2, we now introduce an indirect approach based on collapsing the unwanted connections in operator interpolation.

3.2. *i*BAMG. As with classical AMG, the starting point for determining the interpolation weights is the residual relation expressed in (2.2), again with C_i^c denoting the complement of C_i in the neighborhood of i. In particular, we assume nonzero *unwanted* connections: $a_{ik} \neq 0$ for all $k \in C_i^c$. The objective now is to collapse these connections by approximating the last sum in the residual equation by a linear combination of the errors at the C_i points. The departure point here is that we can no longer assume that the target error is approximately constant. Instead, we use the test vectors to provide the sense of smoothness that we need. As before, once the unwanted connections have been collapsed, we can use the residual relation to write the F-point error, e_i, directly as a linear combination of the e_j for $j \in C_i$, which then yields the desired interpolation weights.

In classical AMG, an approximation is made separately for each e_k with $k \in C_i^c$, so this is a natural approach to take here: for each $k \in C_i^c$, we seek weights β_{kj}, dependent on i, such that

$$e_k \approx \sum_{j \in C_i} \beta_{kj} e_j.$$

Analogous to the BAMG approach, for this indirect interpolation problem, we use least squares to determine each β_{kj}:

$$(i\text{BAMGa}) \qquad \{\beta_{kj} : j \in C_i\} = \arg\min_{\beta_{kj}} \sum_{l=1}^{q} (e_k^{(l)} - \sum_{j \in C_i} \beta_{kj} e_j^{(l)})^2. \tag{3.2}$$

This process results in the approximation

$$\sum_{k \in C_i^c} a_{ik} e_k \approx \sum_{j \in C_i} \sum_{k \in C_i^c} a_{ik} \beta_{kj} e_j$$

and resulting interpolation weights

$$w_{ij} = \frac{1}{a_{ii}} \left(-a_{ij} - \sum_{k \in C_i^c} a_{ik} \beta_{kj} \right). \tag{3.3}$$

Compare this expression with the weights for classical AMG given in (2.4). Note that these weight formulas agree for the case where the β_{kj} reduce to the truncated interpolation formula given in (2.3).

An alternative to separately collapsing unwanted connections is to approximate all of the connections at once: for each $k \in C_i^c$, we seek weights α_j, again dependent on i, such that

$$(i\text{BAMGb}) \qquad \{\alpha_j : j \in C_i\} = \arg\min_{\alpha_j} \sum_{l=1}^{q} (\sum_{k \in C_i^c} a_{ik} e_k^{(l)} - \sum_{j \in C_i} \alpha_j e_j^{(l)})^2. \tag{3.4}$$

This yields the simpler approximation

$$\sum_{k \in C_i^c} a_{ik} e_k \approx \sum_{j \in C_i} \alpha_j e_j$$

and resulting interpolation weights

$$w_{ij} = \frac{1}{a_{ii}} (-a_{ij} - \alpha_j). \tag{3.5}$$

3.2.1. *i*BAMGa and *i*BAMGb equivalence. Fitting the interpolation weights of all of the unwanted connections at once for each $i \in F$ as iBAMGb does is simpler and less expensive than fitting these weights individually as iBAMGa does. So it is important to know that these two approaches actually yield the same weights, provided the least-squares problems are well posed in the sense that the normal equation operator is nonsingular.

LEMMA 3.1. *Denote the vector of values of $e^{(l)}$ at points of C_i by $\varepsilon^{(l)}$ and let L be the $|C_i| \times |C_i|$ matrix defined by*

$$L = \sum_{l=1}^{q} \varepsilon^{(l)} \varepsilon^{(l)T}. \tag{3.6}$$

If L is nonsingular, then definitions (3.3) and (3.5) are equivalent.

Proof. For each $k \in C_i^c$, let β_k denote the vector of values β_{kj}, $j \in C_i$. Also let α denote the vector of values α_j, $j \in C_i$. Then least-squares problem (3.2) results in the normal equation

$$L\beta_k = \sum_{l=1}^{q} e_k^{(l)} \varepsilon^{(l)}, \tag{3.7}$$

for each $k \in C_i^c$, while least-squares problem (3.3) results in just the one normal equation

$$L\alpha = \sum_{l=1}^{q} \sum_{k \in C_i^c} a_{ik} e_k^{(l)} \varepsilon^{(l)}. \tag{3.8}$$

The equivalence between (3.3) and (3.5) now follows from the unique solvability of (3.7) and (3.8) and the relation

$$\alpha = \sum_{k \in C_i^c} a_{ik} \beta_k. \tag{3.9}$$

□

An important implication of this lemma is that, if each connection to C_i^c is collapsed to all C_i points using a rich-enough set of test vectors (note, in particular, that we must have at least $|C_i|$ test vectors), then the combined approach of iBAMGb is to be preferred because it is equivalent to, but less expensive than, iBAMGa. However, because of its greater flexibility, iBAMGa may be useful in cases where different subsets of the interpolatory points are used for each unwanted connection or the set of test vectors is somehow deficient. We demonstrate this flexibility is section 3.5 below.

3.3. BAMG and *i*BAMG conditional equivalence. The motive behind iBAMG is to attempt to insulate coarsening from a crude interpolation formula by relegating this formula to the unwanted and hopefully less important connections to i from C_i^c. The hope is that any crudeness in determining the weights would have less impact if it were used for collapsing the connections indirectly than it would with the approach of determining interpolation weights directly. It is interesting to observe that the indirect and direct approaches are also equivalent in the special case that the residuals for all test vectors at point i are 0.

LEMMA 3.2. *Suppose again that the $|C_i| \times |C_i|$ matrix L defined by (3.6) is nonsingular. Then BAMG and iBAMG (either version) are conditionally equivalent in the sense that they give the same interpolation weights at any point i for which all test-vector residuals are null: $r_i^{(l)} \equiv \left(Ae^{(l)}\right)_i = 0$, $l = 1, 2, \cdots, q$.*

Proof. Denote the vector of values of w_{ij} at points of C_i by w_i and note that the normal equation for the BAMG least-squares problem in (3.1) can be written as

$$Lw_i = \sum_{l=1}^{q} e_i^{(l)} \varepsilon^{(l)}. \tag{3.10}$$

The right side of this equation can be rewritten as

$$\sum_{l=1}^{q} e_i^{(l)} \varepsilon^{(l)} = \frac{1}{a_{ii}} \left(\sum_{l=1}^{q} (r_i - \sum_{j \in C_i} a_{ij} e_j^{(l)} - \sum_{k \in C_i^c} a_{ik} e_k^{(l)}) \varepsilon^{(l)} \right).$$

Letting a_i be the vector of coefficients a_{ij}, $j \in C_i$, and using the premise that $r_i = 0$, we then have

$$\sum_{l=1}^{q} e_i^{(l)} \varepsilon^{(l)} = -\frac{1}{a_{ii}} \left(La_i + \sum_{l=1}^{q} \sum_{k \in C_i^c} a_{ik} e_k^{(l)} \varepsilon^{(l)} \right).$$

From (3.8), we can then rewrite the right side of (3.10) as

$$\sum_{l=1}^{q} e_i^{(l)} \varepsilon^{(l)} = -\frac{1}{a_{ii}} L \left(a_i + \alpha \right),$$

which in turn yields

$$w_i = \frac{1}{a_{ii}}(-a_i - \alpha).$$

□

3.4. *r*BAMG. This equivalence for the case that $r_i = 0$ can be exploited to improve BAMG simply by incorporating the residual in the least-squares process. Specifically, the least-squares problem for BAMG given by (3.1) can be modified by the addition of the local scaled residual as follows:

$$(r\text{BAMG}) \qquad \{w_{ij} : j \in C_i\} = \arg\min_{w_{ij}} \sum_{l=1}^{q} (e_i^{(l)} - \sum_{j \in C_i} w_{ij} e_j^{(l)} - \frac{r_i^{(l)}}{a_{ii}})^2. \qquad (3.11)$$

This change to the fitting process yields a new scheme, which we call rBAMG, that is equivalent to iBAMG for the case that unwanted connections are collapsed to all of C_i and the target vectors are rich enough locally to guarantee a unique fit. This change should therefore improve the direct approach insofar as our numerical tests show the superiority of iBAMG. Thus, we can expect this improved approach to offer better performance for a given number of target vectors and relaxation steps applied to them.

Note that this modification to the direct scheme is equivalent to temporarily relaxing the equation at point i and then applying the standard BAMG minimization approach. As such, rBAMG is related in spirit to the adaptive relaxation scheme described by Brandt in [1] (and suggested in [2]) that applies relaxation selectively to points exhibiting especially large residuals. However, it should be noted that iBAMG offers significant improvement over existing bootstrap methods in two respects. First, iBAMG is equivalent to BAMG only when relaxation is applied to *all points*, and then only *temporarily in a pointwise fashion during the least-squares fit.* This equivalence breaks down when local and/or permanent relaxation is applied to the target vectors. Second, this equivalence also breaks down when either collapsing the unwanted connections does not include all of C_i or there are not enough target vectors to determine a unique least-squares fit. For these reasons, iBAMG represents a more flexible and significant improvement over the standard bootstrap approach.

An important implication of the equivalence between these bootstrap methods is that, when it does hold, then all of the machinery that has so far been developed for BAMG applies in effect to iBAMG. For example, this includes processes for assessing the quality of the current coarse level (i. e., the $C - F$ partition and C_i) as well as the processes that are designed to improve them (see [3] and [5] for further detail).

A result in [11] shows that adaptive AMG is invariant to diagonal scaling in the sense that symmetrically scaling A by any positive diagonal matrix does not change the results, provided the random test vectors are commensurately scaled. This invariance property is important in part because it confirms some sense of stability of the algorithm. As our next lemma shows, rBAMG is also scale invariant.

To be specific, let D be any positive diagonal matrix. With the given C/F-splitting, matrices A and D can be written in block form as follows:

$$A = \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix} \text{ and } D = \begin{bmatrix} D_f & 0 \\ 0 & D_c \end{bmatrix}.$$

LEMMA 3.3. *Let $\hat{A} = DAD$. Then rBAMG applied to $Ax = b$ with target vectors $e^{(l)}$, $l = 1, ..., q$, and rBAMG applied to $\hat{A}\hat{x} = \hat{b}$ with target vectors $\hat{e}^{(l)} = D^{-1}e^{(l)}$, $l =$*

$1, ..., q$, *are equivalent in the sense that the resulting interpolation operators are related by* $\hat{P} = D^{-1}PD_c$.

Proof. Noting that $\hat{r}_i^{(l)} = \left(\hat{A}\hat{e}\right)_i = \left(DADD^{-1}e\right)_i = (DAe)_i = d_i r_i$, then the weights for $\hat{A}$ are given by

$$\begin{aligned}\{\hat{w}_{ij}\} &= \arg\min \sum_{l=1}^{q} (\hat{e}_i^{(l)} - \sum_{j\in C_i} \hat{w}_{ij}\hat{e}_j^{(l)} - \frac{\hat{r}_i^{(l)}}{\hat{a}_{ii}})^2 \\ &= \arg\min \sum_{l=1}^{q} (\frac{e_i^{(l)}}{d_i} - \sum_{j\in C_i} \hat{w}_{ij}\frac{e_j^{(l)}}{d_j} - \frac{d_i r_i^{(l)}}{d_i^2 a_{ii}})^2 \\ &= \arg\min \sum_{l=1}^{q} \frac{1}{d_i^2}(e_i^{(l)} - \sum_{j\in C_i} \hat{w}_{ij}\frac{d_i}{d_j}e_j^{(l)} - \frac{r_i^{(l)}}{a_{ii}})^2 .\end{aligned}$$

Thus, if w_{ij} minimizes (3.11), then so does $\hat{w}_{ij}\frac{d_i}{d_j}$. Hence, we can write the interpolation operator, $\hat{P}$, for $\hat{A}$ in the form

$$\hat{P} = \begin{pmatrix} \hat{W} \\ I \end{pmatrix} = \begin{pmatrix} D_f^{-1}WD_c \\ I \end{pmatrix} = D^{-1}PD_c ,$$

where P is the interpolation operator for A. This proves the assertion. □

This lemma confirms that rBAMG is invariant under symmetric positive diagonal scaling in the sense that the convergence of the process is unchanged and the resulting interpolation operators are related via the diagonal transformation. This also confirms that the resulting multigrid solvers are related in the same way, provided the relaxation processes possess this invariance property.

3.5. Underdetermined case. The equivalence results obtained above, together with the improved performance of iBAMG observed in the next section, suggests that our slight modification to BAMG should generally lead to the need for fewer targets smoothed fewer times. In fact, we may want to consider how well rBAMG performs when the number of targets is smaller than the number of points of C_i, that is, when $q < |C_i|$ so that least-squares problem (3.11) has infinitely many solutions. To decide how to select a sensible solution, we take our cue from iBAMG. When the least-squares problem for the indirect approach has many solutions, it is important for accuracy alone to control the size of the resulting weights. It thus seems natural to select the solution of (3.4) with minimal Euclidean norm. This translates to computing the weights for rBAMG that deviate least in the Euclidean norm from those obtained by operator truncation: find the least-squares solution, w_{ij}, of (3.11) with *minimal deviation* from $-a_{ij}/a_{ii}$ in the sense of minimizing

$$\sum_{j\in C_i} (w_{ij} + \frac{a_{ij}}{a_{ii}})^2. \tag{3.12}$$

The scaled operator coefficients given by $-a_{ij}/a_{ii}$ in (3.12) are "default" weights in the sense that the objective is to stay as close to them as possible when the least-squares fit to the targets is underdetermined. These defaults are not necessarily good weights to use in the adaptive process because they correspond to truncating the unwanted connections, which, in the model problem, leads to improperly scaled weights. (Properly scaling in this case can instead be obtained by collapsing the

unwanted connections to the diagonal, for example.) However, it should be kept in mind that, generally, these defaults would be selected only in the unrealistic case that no targets are available. Just one target is usually enough to adjust the weights from these targets to obtain interpolation that is properly scaled.

Note that we are not prevented from using i in the definition of the weights in (3.12) and the form of interpolation in (3.1). Note also that nothing is forcing us to restrict C_i to the immediate neighborhood of i: it may include points outside of i's nearest neighborhood, perhaps even *only* those points. However, studying these possibilities is beyond the scope of this paper and is therefore left for further research.

A possible unexplored alternative to (3.12) in the iBAMG underdetermined case is to is to restrict the number of interpolation points to the number of test vectors. A similar approach, using iBAMGa, would be to restrict the subset of C_i used to approximate e_j for each $j \in C_i^c$. In each approach, questions arise as to how to choose these subsets. This is also a subject of further research.

4. Numerical experiments. Here we compare the results of BAMG with those obtained by iBAMG. As is usual in AMG, except where noted, the methods are applied recursively to the coarse-grid operator formed by a Galerkin approach based on interpolation (i. e., $A^c = P^T AP$). Since our focus is on how these approaches compare in their determination of the weights of interpolation, we force standard coarsening on every level. In both the BAMG and iBAMG approaches, we relax q different random vectors of unit Euclidean length ν times for the homogeneous problem. We use Gauss-Seidel iteration with lexicographic ordering of the grid points as the relaxation method. All of the results in the following tables reflect an average residual reduction factor over a total residual reduction by a factor of 10^{10} (or over 50 cycles, whichever comes first).

4.1. 2D Laplacian. Consider the two-dimensional Poisson problem with homogeneous Dirichlet boundary conditions on the unit square given by

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega = (0,1)^2, \\ u &= 0 \quad \text{on } \delta\Omega. \end{aligned} \tag{4.1}$$

We discretize (4.1) using standard bilinear elements on a uniform grid, which yields the nine-point stencil given by

$$A = \frac{1}{3h^2} \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}.$$

Although this model problem is not the ultimate target for these methods, it is important to compare the two bootstrap approaches in this simple setting because we can take advantage of knowing optimal coarsening and interpolation weights in a classical AMG approach.

Because of our use of standard coarsening, some of the F points have four neighbors in their interpolatory set. Thus, the use of fewer than four targets ensures an underdetermined least-squares system for these points. Accordingly, the first three rows of each table show the results of using the minimal-deviation rBAMG approach described in the previous section. Note that the use of just one target in Tables 4.1 to 4.3 yields remarkably good performance in all cases, while performance degrades substantially for $q = 2$ and 3. This result is somewhat special to this problem. For $q = 1$, minimizing (3.12) amounts to choosing the equal interpolation weights that

q/ν	1	2	3	4	5	6	7	8	9	10
1	(.54)	(.12)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)
2	(.87)	(.86)	(.85)	(.84)	(.84)	(.84)	(.81)	(.81)	(.80)	(.81)
3	(.81)	(.70)	(.59)	(.50)	(.41)	(.35)	(.39)	(.30)	(.28)	(.26)
4	.87	.87	.87	.87	.87	.87	.87	.87	.87	.87
	(.87)	(.86)	(.86)	(.86)	(.86)	(.86)	(.85)	(.85)	(.85)	(.85)
5	.86	.83	.79	.78	.78	.78	.76	.72	.77	.79
	(.72)	(.52)	(.39)	(.33)	(.49)	(.48)	(.33)	(.56)	(.29)	(.31)
6	.81	.69	.57	.61	.51	.59	.61	.55	.57	.59
	(.44)	(.24)	(.11)	(.11)	(.09)	(.09)	(.13)	(.17)	(.09)	(.11)
7	.77	.54	.39	.34	.35	.40	.30	.27	.37	.34
	(.31)	(.12)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)
8	.73	.45	.31	.23	.23	.19	.25	.18	20	.21
	(.26)	(.10)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)
9	.69	.36	.22	.19	.17	.17	.16	.16	.15	.15
	(.22)	(.09)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)
10	.66	.33	.20	.16	.14	.16	.12	.12	.12	.12
	(.20)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)

Table 4.1: Average five-level V(1,1) convergence factors for residual reduction by a factor of 10^{10} (or at most 50 cycles) on a 64×64 grid 9-point discretization of 2-dimensional model problem (4.1) for various combinations of the number of relaxation sweeps, ν, and the number of random approximations, q. Shown here are the average convergence factors using BAMG (iBAMG). In all cases, a random initial guess was used to test the resulting cycle.

match the target vector (with the sum of the weights becoming more accurate as the number of relaxation sweeps is increased). This choice is exactly what is needed for the Poisson problem, so one vector is sufficient here. In fact, using two vectors results in degradation of convergence, as the tables show. The results of the next section show a somewhat more monotone pattern of increasing improvement with increasing q.

For $q \geq 4 \geq |C_i|$, the consistent trend is that better performance of the resulting solver is obtained by more vectors and/or more relaxation sweeps. Moreover, iBAMG tends to provide substantially better performance for the same number of vectors and sweeps. Also observe the moderate degredation in performance as the size of the problem increases. For example, to obtain a convergence factor of 0.08 with a minimal number of total relaxation sweeps requires $q \times \nu = 7 \times 3 = 21$ sweeps for a 64×64 grid, $q \times \nu = 8 \times 3 = 24$ sweeps for a 128×128 grid, and $q \times \nu = 8 \times 4 = 32$ sweeps for a 256×256 grid. This is to be expected because interpolation must approximate the smoothest components increasingly well on increasingly finer levels for V-cycles to be optimal.

q/ν	1	2	3	4	5	6	7	8	9	10
1	(.75)	(.31)	(.11)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)
2	(.86)	(.86)	(.86)	(.86)	(.86)	(.85)	(.85)	(.85)	(.85)	(.85)
3	(.84)	(.80)	(.76)	(.73)	(.67)	(.65)	(.61)	(.56)	(.51)	(.48)
4	.87	.87	.87	.87	.87	.87	.87	.87	.87	.87
	(.86)	(.86)	(.86)	(.86)	(.86)	(.86)	(.86)	(.86)	(.86)	(.86)
5	.86	.85	.85	.85	.84	.84	.84	.84	.84	.84
	(.81)	(.75)	(.69)	(.68)	(.67)	(.72)	(.72)	(.70)	(.69)	(.67)
6	.84	.79	.74	.71	.74	.73	.75	.73	.75	.75
	(.66)	(.35)	(.21)	(.22)	(.21)	(.26)	(.18)	(.18)	(.15)	(.21)
7	.82	.71	.61	.56	.58	.54	.51	.53	.60	.60
	(.53)	(.16)	(.10)	(.09)	(.08)	(.08)	(.11)	(.08)	(.09)	(.08)
8	.80	.61	.43	.36	.40	.43	.34	.35	.39	.35
	(.47)	(.13)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)
9	.79	.53	.33	.26	.24	.24	.19	.22	.20	.23
	(.42)	(.11)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)
10	.77	.48	.27	.21	.17	.17	.14	.15	.16	.14
	(.38)	(.10)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)

Table 4.2: Average six-level V(1,1) convergence factors for residual reduction by a factor of 10^{10} (or at most 50 cycles) on a 128×128 grid 9-point discretization of 2-dimensional model problem (4.1) for various combinations of the number of relaxation sweeps, ν, and the number of random approximations, q. Shown here are the average convergence factors using BAMG (iBAMG). In all cases, a random initial guess was used to test the resulting cycle.

q/ν	1	2	3	4	5	6	7	8	9	10
1	(.82)	(.61)	(.29)	(.14)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)
2	(.86)	(.85)	(.85)	(.85)	(.85)	(.85)	(.85)	(.85)	(.85)	(.85)
3	(.84)	(.83)	(.82)	(.82)	(.80)	(.78)	(.79)	(.76)	(.77)	(.73)
4	.86	.86	.86	.86	.86	.86	.86	.86	.86	.86
	(.86)	(.85)	(.85)	(.85)	(.85)	(.85)	(.85)	(.85)	(.85)	(.85)
5	.85	.85	.84	.84	.84	.84	.84	.84	.84	.84
	(.83)	(.82)	(.82)	(.81)	(.81)	(.82)	(.82)	(.82)	(.82)	(.82)
6	.84	.82	.81	.81	.81	.82	.82	.82	.82	.82
	(.77)	(.53)	(.43)	(.31)	(.46)	(.47)	(.43)	(.38)	(.51)	(.36)
7	.83	.78	.75	.73	.74	.76	.74	.76	.77	.77
	(.72)	(.31)	(.17)	(.13)	(.14)	(.15)	(.09)	(.08)	(.09)	(.08)
8	.83	.73	.62	.61	.61	.65	.64	.66	.65	.65
	(.69)	(.26)	(.10)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)
9	.82	.68	.48	.43	.39	.41	.41	.39	.36	.35
	(.66)	(.22)	(.09)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)
10	.82	.65	.38	.28	.23	.25	.23	.27	.28	.22
	(.64)	(.20)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)	(.08)

Table 4.3: Average seven-level V(1,1) convergence factors for residual reduction by a factor of 10^{10} (or at most 50 cycles) on a 256×256 grid 9-point discretization of 2-dimensional model problem (4.1) for various combinations of the number of relaxation sweeps, ν, and the number of random approximations, q. Shown here are the average convergence factors using BAMG (iBAMG). In all cases, a random initial guess was used to test the resulting cycle.

4.2. Scaled 2D Laplacian. To maintain the geometric simplicity of our model problem but test performance in the presence of widely varying coefficients, our next example is produced by symmetrically scaling the matrix resulting from (4.1) by a positive diagonal matrix $D = (D_{ii}) = \left(e^{(10*r_i)}\right)$, where r_i is a random number between $-.5$ and $.5$. The scaling is done as follows:

$$A \leftarrow DAD. \tag{4.2}$$

q/ν	1	2	3	4	5	6	7	8	9	10
1	(.77)	(.58)	(.43)	(.35)	(.30)	(.26)	(.22)	(.23)	(.22)	(.21)
2	(.77)	(.65)	(.52)	(.42)	(.40)	(.38)	(.36)	(.35)	(.36)	(.35)
3	(.76)	(.54)	(.32)	(.25)	(.22)	(.21)	(.19)	(.19)	(.19)	(.19)
4	.79	.69	.58	.49	.42	.43	.39	.38	.37	.37
	(.76)	(.54)	(.34)	(.27)	(.26)	(.23)	(.22)	(.20)	(.20)	(.21)
5	.78	.66	.47	.35	.30	.28	.26	.26	.24	.24
	(.75)	(.43)	(.20)	(.13)	(.13)	(.11)	(.10)	(.09)	(.11)	(.11)
6	.78	.64	.43	.27	.23	.20	.20	.20	.19	.19
	(.74)	(.38)	(.15)	(.09)	(.07)	(.07)	(.06)	(.06)	(.06)	(.06)
7	.77	.63	.38	.24	.19	.18	.16	.15	.15	.14
	(.74)	(.35)	(.13)	(.07)	(.05)	(.06)	(.06)	(.06)	(.06)	(.06)
8	.77	.62	.36	.21	.15	.14	.13	.13	.12	.12
	(.73)	(.34)	(.12)	(.06)	(.06)	(.06)	(.06)	(.06)	(.06)	(.06)
9	.77	.62	.33	.19	.14	.12	.12	.10	.10	.10
	(.73)	(.33)	(.11)	(.06)	(.06)	(.06)	(.06)	(.06)	(.06)	(.05)
10	.76	.62	.33	.19	.13	.11	.10	.10	.09	.09
	(.73)	(.32)	(.11)	(.06)	(.05)	(.05)	(.05)	(.05)	(.05)	(.05)

Table 4.4: Average two-level V(1,1) convergence factors for residual reduction by a factor of 10^{10} (or at most 50 cycles) on a 64×64 grid 9-point discretization of 2-dimensional model problem (4.2) for various combinations of the number of relaxation sweeps, ν, and the number of random approximations, q. Shown here are the average convergence factors using BAMG (iBAMG). In all cases, a random initial guess was used to test the resulting cycle.

Results for this test are shown in Table 4.4. For both methods, we see mostly improved convergence as number of target vectors and/or smoothing steps increase. The principal exception is again the case $q = 1$, although the results are not as remarkable as they were for the standard model problem. Note that the performance of iBAMG is again substantially better than that of BAMG in most cases.

4.3. Long-range interpolation. BAMG naturally allows for long-range interpolation, using any of the given coarse grids. To demonstrate that *i*BAMG also provides this capability, Table 4.5 shows two-level results for the 5-point Laplacian with standard coarsening, where some F points have only F-point neighbors.

q/ν	1	2	3	4	5	6	7	8	9	10
4	.74	.65	.58	.55	.52	.50	.50	.51	.49	.49
	(.67)	(.60)	(.52)	(.45)	(.42)	(.40)	(.41)	(.39)	(.41)	(.38)
5	.72	.55	.44	.39	.38	.37	.36	.36	.35	.35
	(.58)	(.44)	(.35)	(.31)	(.28)	(.26)	(.26)	(.25)	(.25)	(.24)
6	.69	.45	.36	.34	.30	.30	.29	.27	.28	.30
	(.53)	(.34)	(.28)	(.23)	(.21)	(.19)	(.18)	(.18)	(.19)	(.17)
7	.66	.39	.31	.28	.27	.25	.24	.24	.22	.23
	(.46)	(.31)	(.22)	(.16)	(.15)	(.14)	(.14)	(.14)	(.16)	(.13)
8	.65	.35	.27	.24	.23	.22	.21	.21	.20	.20
	(.43)	(.25)	(.17)	(.14)	(.13)	(.12)	(.12)	(.12)	(.13)	(.12)
9	.63	.31	.24	.22	.21	.19	.19	.18	.18	.17
	(.38)	(.22)	(.15)	(.12)	(.11)	(.11)	(.11)	(.12)	(.12)	(.12)
10	.61	.28	.23	.20	.19	.18	.18	.16	.16	.17
	(.36)	(.19)	(.13)	(.11)	(.11)	(.11)	(.11)	(.11)	(.12)	(.12)

Table 4.5: Average two-level V(1,1) convergence factors for residual reduction by a factor of 10^{10} (or at most 50 cycles) on a 64×64 grid 5-point discretization of 2-dimensional model problem (4.1) for various combinations of the number of relaxation sweeps, ν, and the number of random approximations, q. Shown here are the average convergence factors using BAMG(*i*BAMG) with standard coarsening. In all cases, a random initial guess was used to test the resulting cycle.

4.4. Variable-coefficient 2D diffusion. Our next example represents a more taxing problem for optimal solvers. It was chosen here because it can cause difficulty with some implementations of standard AMG. The results for this specific case are also fairly representative of our experience so far with variable-coefficient problems, including those with much larger jumps in the coefficients.

Consider the two-dimensional variable-coefficient problem with homogeneous Dirichlet boundary conditions on the unit square given by

$$\begin{aligned} -\nabla \cdot (d(x,y)\nabla u) &= f \quad \text{in } \Omega = (0,1)^2, \\ u &= 0 \quad \text{on } \delta\Omega\,. \end{aligned} \tag{4.3}$$

where, as shown in Fig. 4.1, we have

$$d(x,y) = \begin{cases} 1 & .25 < \max(|x-.5|, |y-.5|) < .375\,, \\ 1000 & \text{otherwise.} \end{cases} \tag{4.4}$$

We discretize (4.3) by finite element method using piecewise bilinear elements, which yields the nine-point stencil given by

$$A = \frac{1}{3h^2} \begin{pmatrix} -d_{nw} & -\frac{(d_{nw}+d_{ne})}{2} & -d_{ne} \\ -\frac{(d_{nw}+d_{sw})}{2} & 2(d_{nw}+d_{ne}+d_{sw}+d_{se}) & -\frac{(d_{se}+d_{ne})}{2} \\ -d_{sw} & -\frac{(d_{sw}+d_{se})}{2} & -d_{se} \end{pmatrix}.$$

Our element boundaries align with the discontinuities in d so that the entries in this stencil refer in an obvious way to the values of d in neighboring elements of each grid point. Table 4.6 shows convergence factors, comparisons, and trends for a 64×64 grid that are quite similar to what we saw for the Poisson case.

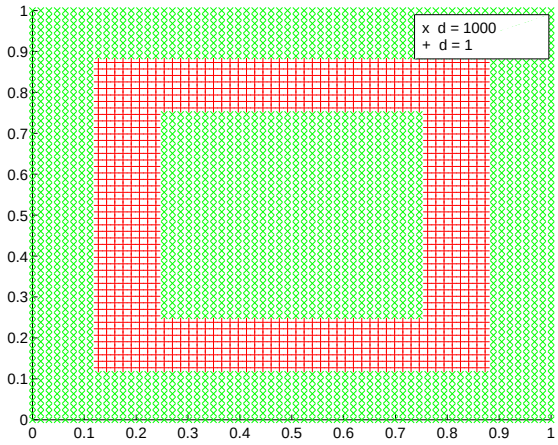

Fig. 4.1: The distribution of the coefficient for (4.3) on $[0\ 1]^2$.

To see what happens when the coefficient discontinuites do not align with the coarse grid, we shifted the red region in Fig 4.1 up and right by $h = \frac{1}{64}$:

$$d(x,y) = \begin{cases} 1 & .25 < \max(|x - .5 - \frac{1}{64}|, |y - .5 - \frac{1}{64}|) < .375\,, \\ 1000 & \text{otherwise.} \end{cases} \tag{4.5}$$

Table 4.7 shows similar performance to the coarse-grid-aligned case.

q/ν	1	2	3	4	5	6	7	8	9	10
1	(.80)	(.80)	(.72)	(.60)	(.45)	(.35)	(.25)	(.21)	(.16)	(.15)
2	(.81)	(.81)	(.78)	(.74)	(.66)	(.62)	(.54)	(.51)	(.47)	(.46)
3	(.81)	(.79)	(.70)	(.58)	(.47)	(.36)	(.28)	(.27)	(.27)	(.28)
4	.81	.81	.78	.73	.67	.59	.54	.50	.45	.44
	(.81)	(.79)	(.70)	(.58)	(.44)	(.35)	(.29)	(.26)	(.24)	(.26)
5	.81	.81	.73	.65	.55	.43	.35	.32	.28	.30
	(.81)	(.75)	(.57)	(.36)	(.23)	(.16)	(.14)	(.11)	(.12)	(.11)
6	.81	.80	.73	.58	.45	.32	.27	.23	.23	.22
	(.81)	(.71)	(.49)	(.30)	(.17)	(.11)	(.08)	(.09)	(.09)	(.07)
7	.81	.79	.68	.51	.36	.25	.20	.17	.15	.16
	(.80)	(.69)	(.44)	(.23)	(.13)	(.09)	(.07)	(.06)	(.06)	(.06)
8	.81	.78	.65	.47	.30	.21	.16	.14	.13	.13
	(.80)	(.67)	(.41)	(.20)	(.11)	(.08)	(.06)	(.06)	(.06)	(.06)
9	.81	.78	.63	.43	.25	.17	.13	.12	.11	.12
	(.80)	(.65)	(.39)	(.19)	(.09)	(.07)	(.06)	(.06)	(.06)	(.06)
10	.81	.78	.62	.39	.22	.15	.11	.10	.11	.11
	(.80)	(.62)	(.33)	(.17)	(.08)	(.06)	(.06)	(.06)	(.06)	(.06)

Table 4.6: Average two-level V(1,1) convergence factors for residual reduction by a factor of 10^{10} (or at most 50 cycles) on a 64×64 grid 9-point discretization of 2-dimensional model problem (4.3) with coefficient in (4.4) for various combinations of the number of relaxation sweeps, ν, and the number of random approximations, q. Shown here are the average convergence factors using BAMG(iBAMG). In all cases, a random initial guess was used to test the resulting cycle.

q/ν	1	2	3	4	5	6	7	8	9	10
1	(.76)	(.78)	(.73)	(.61)	(.49)	(.35)	(.28)	(.24)	(.24)	(.23)
2	(.77)	(.79)	(.77)	(.74)	(.70)	(.63)	(.54)	(.53)	(.49)	(.47)
3	(.78)	(.78)	(.70)	(.57)	(.46)	(.36)	(.33)	(.26)	(.27)	(.25)
4	.76	.78	.77	.74	.65	.60	.55	.50	.48	.43
	(.78)	(.77)	(.70)	(.57)	(.49)	(.36)	(.29)	(.31)	(.29)	(.27)
5	.77	.78	.74	.65	.54	.42	.37	.33	.32	.31
	(.78)	(.73)	(.58)	(.37)	(.27)	(.22)	(.20)	(.18)	(.18)	(.17)
6	.78	.78	.72	.56	.45	.32	.27	.24	.24	.24
	(.78)	(.72)	(.52)	(.30)	(.20)	(.19)	(.18)	(.17)	(.17)	(.15)
7	.78	.78	.68	.51	.37	.27	.22	.19	.20	.19
	(.77)	(.69)	(.44)	(.25)	(.19)	(.20)	(.17)	(.16)	(.14)	(.13)
8	.77	.76	.66	.47	.30	.23	.20	.19	.17	.19
	(.77)	(.66)	(.44)	(.23)	(.18)	(.17)	(.14)	(.15)	(.15)	(.12)
9	.78	.76	.63	.43	.26	.20	.20	.19	.18	.18
	(.77)	(.65)	(.38)	(.20)	(.18)	(.16)	(.16)	(.14)	(.14)	(.13)
10	.78	.75	.60	.38	.26	.20	.19	.16	.16	.17
	(.77)	(.62)	(.35)	(.19)	(.18)	(.15)	(.14)	(.15)	(.13)	(.11)

Table 4.7: Average two-level V(1,1) convergence factors for residual reduction by a factor of 10^{10} (or at most 50 cycles) on a 64×64 grid 9-point discretization of 2-dimensional model problem (4.3) with coefficient in (4.5) for various combinations of the number of relaxation sweeps, ν, and the number of random approximations, q. Shown here are the average convergence factors using BAMG(iBAMG) with standard coarsening. In all cases, a random initial guess was used to test the resulting cycle.

q/ν	1	2	3	4	5	6	7	8	9	10
1	(.79)	(.79)	(.81)	(.81)	(.81)	(.82)	(.82)	(.82)	(.83)	(.83)
2	(.77)	(.77)	(.79)	(.80)	(.80)	(.81)	(.80)	(.80)	(.80)	(.80)
3	(.77)	(.78)	(.79)	(.79)	(.79)	(.75)	(.74)	(.74)	(.69)	(.59)
4	.77	.76	.78	.79	.79	.79	.77	.76	.72	.70
	(.77)	(.77)	(.78)	(.76)	(.75)	(.71)	(.72)	(.65)	(.55)	(.44)
5	.78	.77	.78	.78	.76	.74	.69	.65	.57	.47
	(.77)	(.78)	(.75)	(.69)	(.57)	(.45)	(.44)	(.38)	(.27)	(.25)
6	.78	.76	.77	.77	.75	.69	.63	.56	.48	.34
	(.77)	(.77)	(.71)	(.64)	(.48)	(.38)	(.27)	(.25)	(.24)	(.18)
7	.78	.78	.77	.76	.70	.66	.61	.49	.40	.30
	(.76)	(.77)	(.71)	(.59)	(.47)	(.33)	(.26)	(.21)	(.19)	(.16)
8	.78	.77	.77	.75	.68	.66	.54	.43	.34	.31
	(.75)	(.76)	(.70)	(.57)	(.42)	(.32)	(.25)	(.21)	(.18)	(.15)
9	.78	.78	.77	.75	.70	.63	.55	.40	.33	.26
	(.77)	(.75)	(.67)	(.53)	(.38)	(.28)	(.24)	(.18)	(.16)	(.14)
10	.78	.77	.76	.72	.68	.59	.49	.38	.31	.26
	(.76)	(.76)	(.67)	(.50)	(.37)	(.27)	(.22)	(.18)	(.14)	(.14)

Table 4.8: Average two-level V(1,1) convergence factors for residual reduction by a factor of 10^{10} (or at most 50 cycles) on a 64×64 grid 9-point discretization of 2-dimensional model problem (4.6) with coefficient in (4.4) for various combinations of the number of relaxation sweeps, ν, and the number of random approximations, q. Shown here are the average convergence factors using BAMG(iBAMG). In all cases, a random initial guess was used to test the resulting cycle.

4.5. Diagonally scaled variable-coefficient 2D diffusion. Our final example comes from symmetrically scaling the matrix resulting from (4.3) by the positive diagonal matrix $D = (D_{ii}) = \left(a_{ii}^{-1/2}\right)$ as follows:

$$A \leftarrow DAD. \tag{4.6}$$

The results as shows in Tables 4.8 again are similar to what we have seen in the other examples.

4.6. 2D Gauge Laplacian. The so-called gauge Laplacian from partical physics is a scalar five-point difference operator with unitary but otherwise random complex coefficients. This problem has plagued traditional solvers, including conventional multigrid and algebraic multigrid methods. Just to suggest the general capabilities of BAMG and iBAMG (see [5] for a related study on more general problems from partical physics), we tested them on the eqivalent real form of the gauge Laplacian for a typical random gauge field. Tables 4.9 and 4.10 show that both methods work relatively well for this problem, again with iBAMG showing marked improvement.

q/ν	1	2	3	4	5	6	7	8	9	10
8	.74	.74	.72	.72	.71	.70	.70	.70	.69	.69
	(.30)	(.21)	(.18)	(.15)	(.15)	(.13)	(.14)	(.15)	(.12)	(.11)
9	.73	.68	.63	.59	.58	.54	.51	.50	.51	.47
	(.26)	(.14)	(.11)	(.09)	(.08)	(.08)	(.08)	(.09)	(.08)	(.09)
10	.70	.61	.53	.43	.37	.34	.33	.27	.27	.26
	(.22)	(.11)	(.07)	(.07)	(.07)	(.07)	(.06)	(.07)	(.07)	(.07)

Table 4.9: Average four-level V(1,1) convergence factors for residual reduction by a factor of 10^{10} (or at most 50 cycles) on the equivalent real form of a 32×32 grid 5-point discretization of 2-dimensional Gauge Laplacian for various combinations of the number of relaxation sweeps, ν, and the number of random approximations, q. Shown here are the average convergence factors using BAMG(iBAMG). In all cases, a random initial guess was used to test the resulting cycle.

q/ν	1	2	3	4	5	6	7	8	9	10
8	.73	.72	.71	.70	.70	.69	.69	.69	.68	.68
	(.37)	(.33)	(.32)	(.30)	(.30)	(.29)	(.27)	(.27)	(.27)	(.27)
9	.71	.68	.64	.62	.59	.59	.60	.57	.57	.56
	(.30)	(.24)	(.21)	(.21)	(.20)	(.20)	(.21)	(.18)	(.18)	(.20)
10	.69	.61	.54	.51	.48	.47	.44	.48	.44	.45
	(.25)	(.16)	(.12)	(.13)	(.13)	(.13)	(.13)	(.13)	(.14)	(.13)

Table 4.10: Average five-level V(1,1) convergence factors for residual reduction by a factor of 10^{10} (or at most 50 cycles) on the equivalent real form of a 64×64 grid 5-point discretization of 2-dimensional Gauge Laplacian for various combinations of the number of relaxation sweeps, ν, and the number of random approximations, q. Shown here are the average convergence factors using BAMG(iBAMG). In all cases, a random initial guess was used to test the resulting cycle.

5. Conclusions. The iBAMG scheme introduced in this paper, as a generalization of the classical AMG scheme, seems to be substantially more efficient than the original BAMG approach, at least on the simple problems we studied here. Thus, rBAMG, which is equivalent to iBAMG, seems to provide more accurate interpolation for these problem. The hope is that these improvements carry over to more relevant applications with a sense of smoothness that is either difficult to obtain or wholly unknown (as is the case for quantum chromodynamics). One of the most important benefits of the indirect approach is that modifications to it translate to equivalent modifications to the modified direct approach. As such, iBAMG is an important tool if for no other reason than to suggest improvements to rBAMG.

Acknowledgements. We would like to thank the referees for many helpful comments and suggestions. We also thank Achi Brandt of the Weizmann Institute, Geoff Sanders, Christian Ketelsen, and Marian Brezina of the University of Colorado, and Scott MacLachlan of Tufts University for sharing their extensive experiences with AMG and their many valuable suggestions. This work was sponsored by the US Department of Energy under grants DE-FG02-03ER25574 and DE-FC02-06ER25784, Lawrence Livermore National Laboratory under contract B568677, and the National Science Foundation under grant DMS-0749317 and DMS-0811275.

REFERENCES

[1] Brandt A, Multi-level adaptive solutions to boundary value problems. *Math. Comp.* 1977; **31**:333-390.
[2] Brandt A, General highly accurate algebraic coarsening. *E. T. N. A.* 2000; **10**:1-20.
[3] Brandt A, Multiscale scientific computation: review 2001. In Barth, T.J., Chan, T.F. and Haimes, R. (eds.): Multiscale and Multiresolution Methods: Theory and Application, Springer Verlag, Heidelberg, 2001, pp. 1-96.
[4] Brandt A, Algebraic multigrid theory: the symmetric case. *Applied Mathematics and Computation* 1986; **9**:23-26.
[5] Brandt A, Brannick J, Kahl K, livshits I. A least squares based algebraic multigrid solver for Hermitian and positive definite systems. Manuscript.
[6] Brandt A, McCormick S, Ruge J. Algebraic multigrid (AMG) for sparse matrix equations. In *Sparsity and its Applications*, Evans DJ (ed.). Cambridge University Press: Cambridge, U.K., 1984.
[7] Briggs W, Henson VE, McCormick SF. A *Multigrid Tutorial* (2nd edn). SIAM: Philadelphia, PA, 2000.
[8] Ruge J, Stüben K. Algebraic multigrid (AMG). In *Multigrid Methods*, vol. 5, McCormick SF (ed.). SIAM: Philadelphia, PA, 1986.
[9] Brezina M, Falgout R, MacLachlan S, Manteuffel T, McCormick S, and Ruge J. Adaptive smoothed aggregation (αSA). *SIAM J. Sci. Comp.*. **25**(6), 18961920. (2004).
[10] Brezina M, Falgout R, MacLachlan S, Manteuffel T, McCormick S, and Ruge J. Adaptive smoothed aggregation (αSA) multigrid. *SIAM Rev.*. **47**(2), 317346. (2005).
[11] Brezina M, Falgout R, MacLachlan S, Manteuffel T, McCormick S, and Ruge J. Adaptive algebraic multigrid. *SIAM J. Sci. Comp.*. **27**, 12611286. (2006).
[12] MacLachlan S, Manteuffel T. McCormick S, Adaptive reduction-based AMG, *Num. Lin. Alg. Appl.*.2005; **0**:1-19.