Adapting Algebraic Multigrid

Scott MacLachlan[†]

April 7, 2003

Abstract

Our ability to numerically simulate physical processes is severely constrained by our ability to solve the complex linear systems that are often at the core of the computation. Multigrid methods offer an efficient solution technique for many such problems. However, fixed multigrid processes are based on an overall assumption of smoothness that may not be appropriate for a given problem. Our aim is to develop an adaptive multigrid scheme that replaces this predetermined sense of smoothness by one that is determined automatically. This paper focuses on the principal component of such a scheme: adaptive interpolation. Our method is based on computing a representative error component that is not quickly reduced by relaxation and fitting interpolation so that it is eliminated by the coarse-grid correction process. Numerical results are given to support the efficiency of this approach.

1 Introduction

We are interested in the numerical solution of the partial differential equations resulting from the mathematical modeling of physical systems. We assume that these PDEs have already been discretized in a sensible manner through the use of finite differences or finite elements [1], and our primary focus is on the efficient solution of the linear systems that arise from such discretizations. These systems are typically large (current problems of interest involve millions or even billions of degrees of freedom (dofs)), sparse (a fixed number of non-zero entries per row or column, regardless of problem size), and ill-conditioned (with condition number approaching infinity as problem size increases).

It has been demonstrated that classical linear solvers are quite impractical for these problems ([8], §1.4); however, there are a number of different methods that offer nearly optimal efficiency, all based on multiscale analysis. We consider multigrid methods because of their proven effectiveness for finite element discretizations [2]. These methods are based on the complementary use of stationary linear iterations and coarse-grid correction to obtain what is known as *optimal multigrid efficiency.* Multigrid methods are not, however, currently applicable to every problem of interest. Indeed, the performance of multigrid algorithms can be quite problem dependent and quite discretization dependent.

[†]Department of Applied Mathematics, Campus Box 526, University of Colorado at Boulder, Boulder, CO 80302
Email: maclachl@colorado.edu
Research in collaboration with M Brezina, R Falgout, T Manteuffel, S McCormick, and J Ruge.

Thus, a multigrid algorithm that works well for one problem may not work at all for a similar problem, although careful parameter tuning can result in acceptable performance.

Overcoming this sensitivity is one of our primary research goals. We envision an algorithm that is readily applicable to a large class of problems, without any significant need for parameter tuning.

2 The Adaptive (Algebraic) Multigrid Algorithm (αAMG)

Multigrid is founded on a principle of complementarity. Given a relaxation scheme, coarse-grid correction is used to efficiently eliminate all errors that relaxation does not. In seeking to generalize classical AMG schemes, we must not move away from this principle: Efficient multigrid performance is dependent on the appropriate complementarity of relaxation and coarse-grid correction. In this study, we consider choosing a relaxation scheme a priori (such as Gauss-Seidel) and then defining interpolation to ensure the complementarity needed.

In developing the new interpolation procedure, we consider the case of purely 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 4 are from this implementation in the case of a scalar PDE.

Since the success of our methods depends on the complementarity of relaxation and coarse-grid correction, we see that a good starting point for defining interpolation would be to consider a vector, e, that is not well treated by relaxation. Using a simple (point-wise) relaxation scheme, such as Gauss-Seidel, this also means that $Ae \approx 0$, or

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

assuming we already have a splitting of $N_i = \{j | a_{ij} \neq 0\}$ (the algebraic neighborhood of i) into C_i (the coarse-grid neighbors of i) and F_i (the fine-grid neighbors of i). If, for $k \in F_i$, we can write

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

we determine a general interpolation formula for a point $i \in F$,

$$e_i = -\sum_{j\in C_i} \left(\frac{a_{ij} + \sum\limits_{k\in F_i} a_{ik}w_{kj}}{a_{ii} + \sum\limits_{k\in F_i} a_{ik}w_{ki}} \right) e_j. \tag{1}$$

The difference between our interpolation and the interpolation used in classical AMG [7] is that we choose $\{w_{kj}\}$ to depend on both the entries in the matrix A and some algebraically smooth vector, $x^{(1)}$, which we treat as a representative of a number of algebraically smooth components.

One way of looking at the choice of $\{w_{kj}\}$ is to consider the idea called twice-removed interpolation. 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 fine-fine connections is equivalent to that of determining a way to interpolate to a 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)

$$e_k = \sum_{j \in C_i} w_{kj} e_j. \tag{2}$$

If we have a particular vector, $x^{(1)}$, that we want in the range of interpolation, we ask that Equation 2 hold for $x^{(1)}$. For this interpolation problem, however, we have many choices for the $\{w_{kj}\}$. We choose to take this interpolation to F_i of the form $-D^{-1}A_{fc}$ (where A_{fc} is the matrix of connections between F and C, and D is an arbitrary diagonal matrix - this choice is motivated by the discussion in [3]). We thus write

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

so that (if $x_k^{(1)} \neq 0$)

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

Choosing $w_{kj} = d_{kk}^{-1} a_{kj}$, we get the interpolation formula

$$\begin{aligned} e_k &= -\sum_{j \in C_i} \frac{a_{kj}}{d_{kk}} e_j \\ &= \sum_{j \in C_i} \frac{a_{kj} x_k^{(1)}}{\sum_{j' \in C_i} a_{kj'} x_{j'}^{(1)}} e_j \\ &= \sum_{j \in C_i} w_{kj} e_j. \end{aligned}$$

Interpolation to $i \in F$ is then given by Equation 1 and has the particular form

$$e_i = -\sum_{j \in C_i} \left(\frac{a_{ij} + \sum_{k \in F_i} a_{ik} \frac{a_{kj} x_k^{(1)}}{\sum_{j' \in C_i} a_{kj'} x_{j'}^{(1)}}}{a_{ii}} \right) e_j. \tag{3}$$

Treating the interpolation operator, P, as an operator from C to $F \cup C$, we see that it has the form

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

where W is the matrix of coefficients as in Equations 1 and 3.

Another way of looking at twice-removed interpolation is to consider how it relates to the interpolation used in its place in Ruge-Stueben AMG. For strongly connected points, AMG uses interpolation of the form

$$w_{kj} = \frac{a_{kj}}{\sum_{m \in C_i} a_{km}}$$

One of the primary assumptions of AMG is that the global smoothest error component is the constant vector [7], and thus it must be in the range of interpolation. In our terms, AMG assumes that $x^{(1)} \equiv 1$. Looking at our interpolation formula, we see that it reduces to the classical AMG strong interpolation formula in this case. So, we can look at the αAMG interpolation formula as a simple generalization of the classical AMG formula for the case where the smoothest error component (and thus one that needs to be in the range of interpolation) is any known vector.

Successful implementation of this scheme for interpolation thus relies upon having an appropriate vector, $x^{(1)}$, to put in the range of interpolation. Since we need the complementarity of relaxation and coarse-grid correction, it follows that the best choice for $x^{(1)}$ would be a representative of the vectors for which relaxation is inefficient. Thus, a straight-forward method for generating such a representative would be to start with a vector equally rich in all components (i.e. eigenvectors of symmetric A), construct a b so that the error in $Ax = b$ matches this initial vector, and then determine the error in the approximate solution after a sufficient number of relaxations.

Getting a representative vector is thus easy if we apply relaxation to an equation whose solution is known. We are interested in the error after a number of relaxations on $Ax = b$. Clearly, this error is easiest to calculate if we choose b so that we know the true solution. This is easily done by choosing a solution, x, and then calculating the appropriate right side, b, but we can make this problem even easier by choosing $x \equiv 0$, so that $b \equiv 0$. Then, starting with any (non-zero) initial guess and relaxing on $Ax = 0$, we can identify the error in the solution after relaxation as simply being the current approximation, since the true solution is 0. So, starting with a random initial guess and performing relaxation on $Ax = 0$ generates a vector, $x^{(1)}$, representative of the slow to converge components that we can then use in the interpolation formula.

In practice, however, it requires (in our opinion) too many relaxation sweeps to generate a suitable representative using only relaxation. Thus, to improve performance, we have implemented a multilevel scheme to quickly generate a vector, $x^{(1)}$, such that $Ax^{(1)} \approx 0$ and so for which relaxation is slow to resolve. To do this, we start with a random guess on the fine grid and perform a few relaxation sweeps to generate a tentative $x^{(1)}$. From this, we generate an interpolation operator (as above) and form the CGO using the Galerkin condition. We use injection (direct restriction of the values on the C-points) to form a coarse-grid initial guess and recurse to the coarsest level. From this coarsest level, we interpolate the vector back to the finest level, possibly relaxing as we do, and repeat this process, using that vector as an overall initial guess. In Section 4, we investigate this procedure, where we focus on choosing the number of relaxation sweeps to do in each stage of the algorithm. We have found that, with enough relaxation sweeps in this initial stage, we do indeed recover grid independent convergence factors all the way to 1024×1024 grids for many scalar problems.

3 Theoretical Properties

We have already seen one advantageous property of our method, namely, that it reduces to a form of the classical Ruge-Stueben AMG algorithm when $x^{(1)} = 1$. We claim this property to be advantageous in that we can reasonably expect our algorithm to perform well on the class of problems that AMG performs well on (dependent, of course, on sufficient resolution of $x^{(1)}$). However, we believe that our choice of interpolation for $x^{(1)} \neq 1$ leads to an algorithm that is much more successful.

One situation that causes difficulty for AMG is when the matrix for a problem it is effective on is simply rescaled. In particular, consider taking the matrix A and multiplying it by a positive, diagonal scaling matrix on both the left and the right sides (to preserve symmetry). That is, replace A with $\tilde{A} = DAD$ for some positive, diagonal matrix D. If $Ax^{(1)} = 0$, then $\tilde{A}(D^{-1}x^{(1)}) = 0$, so the new near null-space component is actually $D^{-1}x^{(1)}$. If the diagonal entries of D have significant variation in them, then $D^{-1}x^{(1)}$ has a significantly different character than $x^{(1)}$. In the case of classical AMG, this can cause a significant deterioration in convergence rates.

Our algorithm, however, is not as sensitive to this distinction. In fact, under minimal assumptions, our algorithm can be shown to be unaffected by such scaling. Let A and D have the forms

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

so that

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

Suppose that if the process for generating the representative vector generates $x^{(1)}$ when given the matrix A, then it generates $\tilde{x}^{(1)} = D^{-1}x^{(1)}$ when given the matrix $\tilde{A}$. Our choice for interpolation for the matrix $\tilde{A}$ is then given by taking

$$\begin{aligned} \tilde{w}_{kj} &= \frac{\tilde{a}_{kj}\tilde{x}_k^{(1)}}{\sum_{j' \in C_i} \tilde{a}_{kj'}\tilde{x}_{j'}^{(1)}} \\ &= \frac{d_k a_{kj} d_j d_k^{-1} x_k^{(1)}}{\sum_{j' \in C_i} d_k a_{kj'} d_{j'} d_{j'}^{-1} x_{j'}^{(1)}} \\ &= d_k^{-1} w_{kj} d_j, \end{aligned}$$

which gives us

$$\begin{aligned} e_i &= -\sum_{j \in C_i} \left(\frac{\tilde{a}_{ij} + \sum_{k \in F_i} \tilde{a}_{ik}\tilde{w}_{kj}}{\tilde{a}_{ii}} \right) e_j \\ &= -\sum_{j \in C_i} \left(\frac{d_i a_{ij} d_j + \sum_{k \in F_i} d_i a_{ik} d_k d_k^{-1} w_{kj} d_j}{d_i a_{ii} d_i} \right) e_j \\ &= -\sum_{j \in C_i} d_i^{-1} \left(\frac{a_{ij} + \sum_{k \in F_i} a_{ik} w_{kj}}{a_{ii}} \right) d_j e_j \end{aligned}$$

for $i \in F$. For $i \in C$, we simply take the value from the coarse-grid and assign it as the value on the fine-grid. Thus, we get an interpolation operator $\tilde{P}$ of the form

$$\tilde{P} = \begin{bmatrix} \tilde{W} \\ I \end{bmatrix} = \begin{bmatrix} D_f^{-1} W D_c \\ I \end{bmatrix} = D^{-1} P D_c,$$

where P is the interpolation operator from the non-scaled case. Further, if we consider the coarse-grid operators A_c and $\tilde{A}_c$, we see that

$$\tilde{A}_c = \tilde{P}^T \tilde{A} \tilde{P} = (D_c P^T D^{-1})(DAD)(D^{-1} P D_c) = D_c P^T A P D_c = D_c A_c D_c.$$

That is, the coarse-grid operator for the scaled problem is simply the scaled version of the coarse-grid operator for the unscaled problem. Noticing that standard relaxation techniques such as Gauss-Seidel or Jacobi (both point-wise and block relaxations) are scaling invariant (that is, if we scale the matrix A to DAD as above, the initial guess $x^{(0)}$ to $D^{(-1)} x^{(0)}$, and the initial right side b to Db, then the approximation generated changes from $x^{(1)}$ to $D^{-1} x^{(1)}$), we see that the entire process is independent of any diagonal scaling. That is, we expect to get similar convergence factors (measured in the energy norm) for the diagonally scaled problem as we get for the unscaled problem.

The one assumption this result uses is that, for the scaled problem, we can generate the scaled vector, $D^{-1} x^{(1)}$, just as easily as the unscaled vector, $x^{(1)}$. In practice this is more difficult. If we do happen to know the scaling matrix D a priori, then it is true that, by scaling the initial guess, we can ensure equivalence between the two vectors. This is, of course, an unrealistic situation - if we knew the scaling beforehand, we could simply unscale the problem. Starting with the same initial guess for both the scaled and unscaled problems tends to give slightly worse convergence rates for the unscaled problem (as is shown in Section 4), but this is not a significant performance hit.

4 Numerical Results

To examine the feasibility of our approach, we implemented our solver for the special case of a rectangular grid in 2-dimensions with full coarsening. This restriction in generality has a notable effect on the range of problems that we are able to reasonably consider (anisotropy, for example, becomes much more difficult to account for in this setting). However, if we examine problems without such difficulties, we feel that we can get a good indication of the performance of our approach.

While a significant range of tests would be necessary to catalog the overall strengths and weaknesses of our approach (as in [5] and [4]), we concentrate here on determining the appropriate form of relaxation to perform in the setup stage. To determine overall efficiency, we must consider the costs of both the setup and the solution phases. We measure these in work units - the cost of a sweep of relaxation on the finest grid.

To simplify our comparison, we consider a fixed solution stage consisting of V(1,1) cycles. We consider our problem solved if we have reduced the error in the energy norm by a factor of 10^{-6}. Given an algorithm with asymptotic convergence factor ρ, we know that $n \geq \frac{-6}{\log \rho}$ steps are needed to do this. Since a V(1,1) cycle costs approximately $\frac{8}{3}$ work units ([6], p. 47), we get an approximate cost of $\frac{-16}{\log \rho}$ work units for the entire solution stage.

The cost of the setup stage is similarly computed. We begin by performing a given number, ν_0, of relaxation sweeps on the finest grid. We then perform two sweeps of the process described in Section 2, performing ν_1 relaxations on each level of the downward sweep in both cycles and ν_2 relaxations of each level in the upward sweep of the first cycle (the MG hierarchy is not changed after the second downward sweep, so relaxation on the upward stage is unnecessary). Thus, the cost in work units of these relaxations is $\nu_0 + \frac{8}{3}\nu_1 + \frac{4}{3}\nu_2$.

We compute these costs for four test matrices. Our first two problems come from bilinear finite element discretizations on the canonical unit square. Problem 1 is Laplace's Equation with pure Dirichlet boundary conditions, Problem 2 is

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

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

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

Our second two problems come from scaling these matrices, as described in Section 3, with the nodal scaling function

$$1 + \sin(547\pi x_i)\sin(496\pi y_j) + 10^{-7}.$$

This function gives a nearly-independent scaling on each node by a constant in the range $[10^{-7}, 2 + 10^{-7}]$, but does not change character with h.

As a basis for comparison, we consider the convergence properties of a full-coarsening multigrid using the AMG strongly-connected formula for collapsing fine-fine connections. That is, we force AMG-style interpolation onto our coarsening scheme. The approximate numbers of work units to achieve convergence (that is, a reduction in the energy-norm of a factor of 10^{-6}) are reported in Table 1, computed as discussed above.

Problem Size	Problem 1	Problem 2	Problem 3	Problem 4
32×32	12.9	14.5	1297	59.4
64×64	13.4	15.6	4075	112.1
128×128	13.6	14.9	6122	218.7
256×256	13.8	16.4	6122	430.6
512×512	13.9	15.2	7350	858.6
1024×1024	13.9	16.7	7350	1656

Table 1: Work Units for standard AMG

These results show that classical AMG interpolation gives a scalable solver for Problems 1 and 2, coming directly from discretization. If, however, the discretization matrices are scaled, there is a significant increase in the needed work units for solution, especially as the problem grows but also on the coarsest grids. We are unable to improve upon the results from Problems 1 and 2, so our aim should be to determine a balance where we significantly improve the results from the scaled problems, while not driving up the cost of solution for the directly discretized problems.

We now investigate two points in determining an appropriate form for the initial relaxation. First, we determine where it is appropriate to invest our work (as relaxations on the fine level, on the downward part of the cycle, or on the upward part of the cycle), then we determine how much work is necessary to obtain scalable results.

We answer the first question by fixing the number of work units performed in the setup stage at twelve. There are a plentiful number of ways to allocate these work units so that $\nu_0 + \frac{8}{3}\nu_1 + \frac{4}{3}\nu_2 = 12$, and we present, for illustration, the cases $\nu_0 = 4, \nu_1 = 2, \nu_2 = 2$ in Table 2, and $\nu_0 = 0, \nu_1 = 3, \nu_2 = 3$ in Table 3. These tables show the number of work units required for the solution phase only, not counting the 12 work units required for the setup. The results for $\nu_0 = 4, \nu_1 = 3, \nu_2 = 0$ and $\nu_0 = 4, \nu_1 = 1, \nu_2 = 4$ are quite similar to those in Table 2, whereas the results for $\nu_0 = 4, \nu_1 = 0, \nu_2 = 6$ were quite poor, closer to (though slightly better than) the results in Table 3.

Problem Size	Problem 1	Problem 2	Problem 3	Problem 4
32×32	12.9	14.7	12.9	14.7
64×64	13.4	15.6	13.5	15.4
128×128	13.6	14.9	13.7	14.7
256×256	13.9	16.4	13.9	25.2
512×512	13.9	15.8	13.9	16.4
1024×1024	13.9	23.2	13.9	25.6

Table 2: Work units for αAMG with $\nu_0 = 4, \nu_1 = 2, \nu_2 = 2$

Problem Size	Problem 1	Problem 2	Problem 3	Problem 4
32×32	12.9	14.8	12.9	14.7
64×64	13.4	15.6	13.5	15.3
128×128	13.6	14.7	13.8	15.4
256×256	13.8	16.4	13.9	30.3
512×512	13.9	24.0	13.9	25.8
1024×1024	749.0	926.1	103.7	977.2

Table 3: Work units for αAMG with $\nu_0 = 0, \nu_1 = 3, \nu_2 = 3$

These results suggest that the best form for the setup phase is to divide the relaxation effort somewhat evenly between relaxations on the fine grid and relaxations in the multilevel cycle. Within this cycle, it seems that the best results are achieved by balancing the relaxations between the upward and the downward paths. Following these results, we tested the combinations $\nu_0 = 2, \nu_1 = 1, \nu_2 = 1$ and $\nu_0 = 4, \nu_1 = 3, \nu_2 = 3$. Using 6 work units gave mediocre results for the two Poisson-based problems, showing similar results to those in Table 2 for all grids except 1024×1024, where some growth in the number of work units needed is seen. The results for the two discontinuous-coefficient based problems are significantly worse, however, than those shown in Table 3. Increasing the number of works units to 18, by choosing $\nu_0 = 6, \nu_1 = 3, \nu_2 = 3$, provided quite excellent results, as shown in Table 4

These results are very much as we hoped. For the latter two problems, we see a tremendous improvement in the amount of effort required for solution as compared to the AMG-interpolation based results in Table 1. The solution phase of the algorithm scales well across all 6 grids, and the actual costs are

Problem Size	Problem 1	Problem 2	Problem 3	Problem 4
32×32	12.9	14.9	12.9	14.9
64×64	13.4	15.6	13.5	15.3
128×128	13.6	15.2	13.7	15.3
256×256	13.8	16.4	13.8	16.4
512×512	13.9	15.2	13.9	15.2
1024×1024	13.9	16.7	13.9	16.8

Table 4: Work units for αAMG with $\nu_0 = 6, \nu_1 = 3, \nu_2 = 3$

quite reasonable. It must be noted, however, that once we account for the cost of the setup phase of our algorithm, AMG is a much better solver for the two unscaled matrices. Put simply, if one knows the algebraically smoothest component of an elliptic PDE exactly, one can do no better than designing the multigrid interpolation based on that component. Indeed, if we give this component as input to our method for creating the multigrid hierarchy, we can solve the problem with the same cost as AMG on the unscaled problem, simply by using it as $x^{(1)}$ in Equation 3, as discussed in Section 3.

5 Conclusions

Our numerical results indicate that our full-coarsening-based version of αAMG does offer some advantages when compared to geometric multigrid or classical AMG. The foci of our future work are improving and extending this algorithm as well as understanding the theoretical roots of its performance. There are many possible avenues for this investigation, and we hope that, by careful study, we can produce an algorithm offering significant improvements in robustness over earlier methods without adding undue computational expense.

References

[1] D. Braess, *Finite Elements*, Cambridge University Press, Cambridge, 2001. Second Edition.

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

[3] M. Brezina, A. J. Cleary, R. D. Falgout, V. E. Henson, J. E. Jones, T. A. Manteuffel, S. F. McCormick, and J. W. Ruge, *Algebraic multigrid based on element interpolation (AMGe)*, SIAM J. Sci. Comput., 22 (2000), pp. 1570–1592.

[4] M. Brezina, R. D. Falgout, S. P. MacLachlan, T. A. Manteuffel, S. F. McCormick, and J. W. Ruge, *Adaptive algebraic multigrid (αAMG)*, in preparation, (2003).

[5] ——, *Adaptive smoothed aggreation (αSA)*, submitted, SIAM J. Sci. Comput., (2003).

[6] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, SIAM Books, Philadelphia, 2000. Second edition.

[7] J. W. Ruge and K. Stüben, *Algebraic multigrid (AMG)*, in Multigrid Methods, S. F. McCormick, ed., vol. 3 of Frontiers in Applied Mathematics, SIAM, Philadelphia, PA, 1987, pp. 73–130.

[8] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, London, 2001.