LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-JRNL-220851

An Introduction to Algebraic Multigrid

R. D. Falgout

April 25, 2006

Computing in Science and Engineering

An Introduction to Algebraic Multigrid

Robert D. Falgout*

June 12, 2006

Abstract

Algebraic multigrid (AMG) solves linear systems based on multigrid principles, but in a way that only depends on the coefficients in the underlying matrix. The author begins with a basic introduction to AMG methods, and then describes some more recent advances and theoretical developments.

Introduction

Multigrid methods are called *scalable* or *optimal* because they can solve a linear system with N unknowns with only $O(N)$ work. Since this work can be effectively distributed across a parallel machine, multigrid methods are able to solve ever larger problems on proportionally larger parallel computers in essentially constant time, making them an ideal solver for large-scale scientific simulation.

Multigrid methods achieve optimality by employing two complementary processes: *smoothing* and *coarse-grid correction*. Smoothing involves the application of a *smoother* (also called a *relaxation method*), which is generally a simple iterative method like Gauss-Seidel. Coarse-grid correction involves transferring information to a coarser grid through *restriction*, solving a coarse-grid system of equations, then transferring the solution back to the fine grid through *interpolation* (also called *prolongation*). In the classical geometric multigrid setting (see Yavneh's article in this issue), smoothing reduces (oscillatory) high-frequency error, while coarse-grid correction eliminates (smooth) low-frequency error. Although this geometric interpretation of multigrid was critical to the early development of the method and still plays an important role in simulation today, there are classes of problems for which geometric techniques fall short (see the "Operator-Dependent Interpolation and AMG" sidebar).

Algebraic multigrid (AMG) [2, 3, 4] was introduced as a method for solving linear systems based on multigrid principles, but in a way that requires no explicit knowledge of the problem geometry. The AMG method determines coarse "grids", inter-grid transfer operators, and coarse-grid equations based solely on the matrix entries. Since the original introduction of the method, a wide variety of AMG algorithms have been developed that target different problem classes and have different robustness and efficiency properties. In this paper, we give a basic introduction to AMG methods, beginning with a description of the classical algorithm of Brandt, McCormick, Ruge, and Stüben, then moving on to some more recent advances and theoretical developments. For a more thorough introduction to AMG, see [5, 6]. For an overview of parallel AMG methods, see [7].

AMG Basics

Before we get into the specifics of particular AMG algorithms, it is important to set the stage by introducing the basic ingredients of the method. Since AMG is a matrix-based method, we start with a purely linear algebra point-of-view.

We are interested in solving the linear system

$$A\mathbf{u} = \mathbf{f}, \tag{1}$$

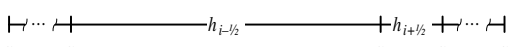
where A is a real $n \times n$ matrix and $\mathbf{u}, \mathbf{f}$ are vectors in $\mathbb{R}^n$. To keep things simple, we assume that

*Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, P.O. Box 808, L-561, Livermore, CA 94551. E-mail: rfalgout@llnl.gov. This work was performed under the auspices of the U. S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract W-7405-Eng-48.

Operator-Dependent Interpolation and AMG

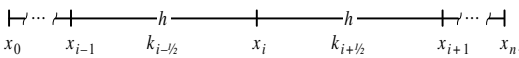
Operator-dependent interpolation was introduced in 1981 [1] to solve diffusion problems with large jumps in coefficients. Unlike geometric interpolation techniques, operator-dependent interpolation takes into account coefficients in the discrete operator. To illustrate this idea and see the connection to AMG, consider two simple 1D problems (on domain Ω with boundary Γ) discretized with piecewise-linear finite elements. The two problems, together with the i^{th} discrete equations for their differential terms, are as follows:

(a) $-u_{xx} = f_a$ on Ω; $u = g_a$ on Γ



$$\Rightarrow -\left(\frac{1}{h_{i-\frac{1}{2}}}\right) u_{i-1} + \left(\frac{1}{h_{i-\frac{1}{2}}} + \frac{1}{h_{i+\frac{1}{2}}}\right) u_i - \left(\frac{1}{h_{i+\frac{1}{2}}}\right) u_{i+1}$$

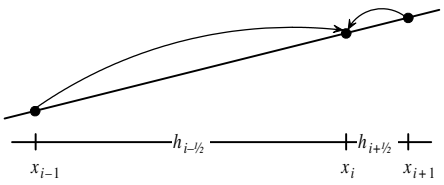
(b) $-(ku_x)_x = f_b$ on Ω; $u = g_b$ on Γ



$$\Rightarrow -\left(\frac{k_{i-\frac{1}{2}}}{h}\right) u_{i-1} + \left(\frac{k_{i-\frac{1}{2}}}{h} + \frac{k_{i+\frac{1}{2}}}{h}\right) u_i - \left(\frac{k_{i+\frac{1}{2}}}{h}\right) u_{i+1}$$

It is well known that linear interpolation works well for problem (a), but not for problem (b) if k has large jumps. However, we can derive an appropriate interpolation for problem (b) by noticing that it is equivalent to problem (a) if the grid spacing satisfies $h_{i-\frac{1}{2}} = h/k_{i-\frac{1}{2}}$:

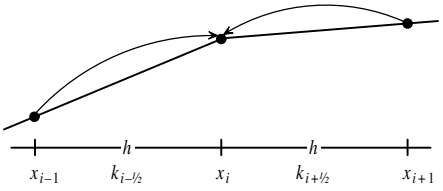
linear interpolation for (a)



$$u_i = \left(\frac{h_{i+\frac{1}{2}}}{h_{i-\frac{1}{2}}+h_{i+\frac{1}{2}}}\right) u_{i-1} + \left(\frac{h_{i-\frac{1}{2}}}{h_{i-\frac{1}{2}}+h_{i+\frac{1}{2}}}\right) u_{i+1}$$

=

operator-dependent interpolation for (b)



$$u_i = \left(\frac{k_{i-\frac{1}{2}}}{k_{i-\frac{1}{2}}+k_{i+\frac{1}{2}}}\right) u_{i-1} + \left(\frac{k_{i+\frac{1}{2}}}{k_{i-\frac{1}{2}}+k_{i+\frac{1}{2}}}\right) u_{i+1}$$

From this example, we see that geometric information alone is not enough to solve some classes of problems. AMG takes this idea to the extreme by ignoring geometric information altogether. If we compare the discrete equations in (a) and (b) to the i^{th} matrix equation, $(A\mathbf{u})_i = a_{i,i-1}u_{i-1} + a_{i,i}u_i + a_{i,i+1}u_{i+1}$, we see that both linear and operator-dependent interpolation can be written entirely in terms of matrix coefficients:

$$u_i = \left(-\frac{a_{i,i-1}}{a_{i,i}}\right) u_{i-1} + \left(-\frac{a_{i,i+1}}{a_{i,i}}\right) u_{i+1}.$$

A is symmetric positive definite (SPD). Recall that the two main components of multigrid are smoothing and coarse-grid correction. Coarse-grid correction involves operators that transfer information between fine and coarse "grids", which are denoted in linear algebra terms simply as the vector space $\mathbb{R}^n$ and the lower-dimensional (coarse) vector space $\mathbb{R}^{n_c}$. Interpolation (prolongation) maps the coarse grid to the fine grid and is just the $n \times n_c$ matrix $P : \mathbb{R}^{n_c} \to \mathbb{R}^n$. Restriction maps the fine grid to the coarse grid, and is the transpose of interpolation (P^T) in this paper. The two-grid method for solving (1) is then defined as follows:

$$\text{Do } \nu_1 \text{ smoothing steps on } A\mathbf{u} = \mathbf{f}. \tag{2a}$$
$$\text{Compute residual } \mathbf{r} = \mathbf{f} - A\mathbf{u} = A\mathbf{e}. \tag{2b}$$
$$\text{Solve } A_c\mathbf{e}_c = P^T\mathbf{r}. \tag{2c}$$
$$\text{Correct } \mathbf{u} \leftarrow \mathbf{u} + P\mathbf{e}_c. \tag{2d}$$
$$\text{Do } \nu_2 \text{ smoothing steps on } A\mathbf{u} = \mathbf{f}. \tag{2e}$$

In (2b), $\mathbf{e}$ is called the *error* and is the difference between the exact solution and the current iterate, i.e., $\mathbf{e} = A^{-1}\mathbf{f} - \mathbf{u}$. In (2c), we solve for a coarse approximation, $\mathbf{e}_c$, to this error. In practice, the coarse system in (2c) is solved by recursively re-applying algorithm (2), yielding a hierarchy of coarse grids, transfer operators, and coarse-grid systems. Because AMG is based only on the matrix A, there are few options for defining the coarse system A_c. The most common approach is to use the Galerkin operator, $A_c = P^T AP$, which has the nice property that it minimizes the error after correction (in the energy norm).

Adding geometry to the discussion

Although the goal of AMG is to solve matrix equations using multigrid principles, it is difficult to get an intuitive understanding of the method from a purely matrix point-of-view. Since many of the problems we are interested in solving come from discretized partial differential equations (PDE's), one of the best approaches for visualizing AMG is to relate the matrix equations back to an underlying PDE. We rely heavily on this technique in the paper. In fact, our descriptions almost exclusively use PDE's on two-dimensional structured grids. It is important to remember that, although our illustrations contain geometry, AMG does not actually use any geometric information.

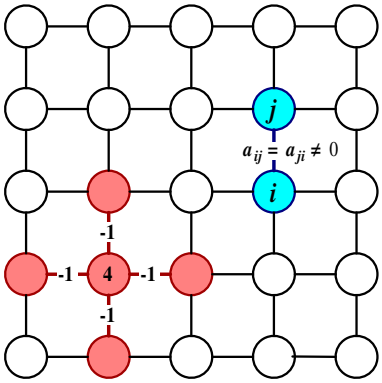


Figure 1: Illustration of the matrix adjacency graph for a 5-point discretization of the Laplace equation on a 5×5 uniform grid. The edges between points i and j correspond to nonzero entries a_{ij} in the matrix. The discretization stencil (bottom left) is just a row in the matrix.

The adjacency graph of the matrix plays an important role in AMG. The graph has a directed edge from vertex i to vertex j for every nonzero entry a_{ij} in the matrix A (see Figure 1). The *grid* in AMG is simply the set of vertices in the graph, i.e., grid point i is just vertex i. If the linear system comes from the discretization of a PDE, then we can draw the grid points in their actual geometric locations along with the associated graph. We illustrate this in Figure 1 for a simple 2D Laplacian problem.

Algebraic smoothness

In AMG, the smoother is generally fixed to be a simple pointwise method such as Gauss-Seidel. Error not eliminated by the smoother is called *smooth error*, and must be handled by coarse-grid correction (recall algorithm (2)). In the classical geometric multigrid setting, smooth error is smooth in the usual geometric sense. In the AMG setting, however, smooth error may actually be geometrically oscillatory. We often use the term *algebraically smooth* to be clear about the distinction.

To see this, consider the following simple 2D exam-

ple, discretized by finite elements on a uniform mesh:

$$\begin{aligned} -au_{xx} - bu_{yy} &= f \quad \text{on } \Omega \\ u &= g \quad \text{on } \Gamma \end{aligned} \tag{3}$$

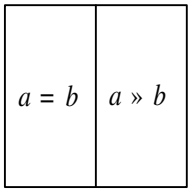


Figure 2 shows the error after 7 Gauss-Seidel iterations. We see that the error is geometrically smooth in both the x and y directions in the left-half plane where the problem is isotropic, but it is geometrically oscillatory in the y direction in the right-half plane where the problem is anisotropic. As illustrated in the figure, AMG coarsens in directions of geometric smoothness. That is, in the left-half plane, the grid is coarsened in both directions (so-called full coarsening), but in the right-half plane, the grid is coarsened only in the x direction (so-called semicoarsening). This ability of AMG to "follow the physics" during coarse-grid correction is another advantage it has over geometric approaches.

The key to designing an effective AMG algorithm is to have a good characterization of smooth error. In general, smooth error corresponds to eigenvectors of A with small associated eigenvalues (we call these *small eigenmodes* for short). In other words, smoothing damps large eigenmodes, leaving coarse-grid correction to eliminate the remaining small eigenmodes of A. As we see later in the paper when we discuss AMG theory, the smaller the eigenmode, the more effective must be coarse-grid correction. This makes the smallest of the eigenmodes, called the *near null space* or *near kernel* of A, particularly important in the design of AMG algorithms.

As an example, we again consider problem (3). Any linear function u is in the kernel of the differential operator since both u_{xx} and u_{yy} are zero. The same is true for the discrete operator A (away from boundaries). That is, the near null space of A for this problem consists of any vector that is almost linear when plotted on the grid. Hence, it makes perfect sense for AMG to coarsen in directions of geometric smoothness, as shown in Figure 2. The example underscores the distinction between smooth error (er-

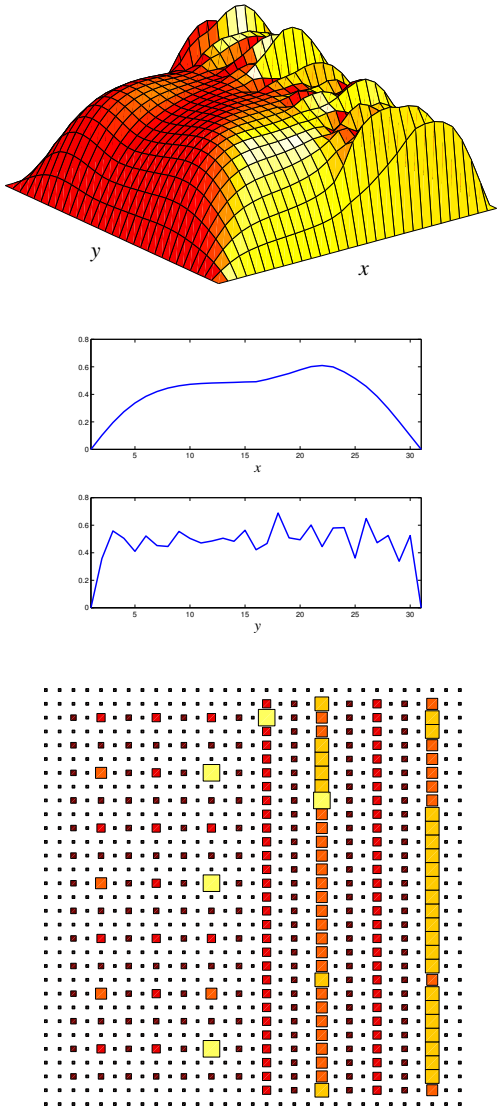


Figure 2: Smooth error for problem (3) after 7 sweeps of Gauss-Seidel (top image). The error is geometrically smooth in the x direction (2nd image), but oscillatory in the y direction in the right-half plane (3rd image). For this problem, AMG coarsens the grid in directions of geometric smoothness (bottom image). Larger squares correspond to coarser grids.

ror not eliminated by the smoother) and the near null space (the smallest eigenmodes of A). In the example, smooth error consists of functions that are geometrically both smooth and oscillatory, while the near null space contains only geometrically smooth functions. For applications where the near null space contains geometrically oscillatory functions (such as electromagnetics), the approach of coarsening in directions of geometric smoothness is not sufficient.

So, how can we use knowledge of the near null space to design a real AMG algorithm? There are many approaches that researchers have used, and we describe some of them in this paper. But, first, we start with the classical AMG algorithm of Brandt, McCormick, Ruge, and Stüben [2, 3, 4].

Classical AMG

As in the example of the previous section, the classical AMG algorithm (C-AMG) is based on the assumption that geometrically smooth functions are in the near null space of A. Since AMG knows nothing about the geometry of the problem, we need to characterize this in some algebraic way.

To simplify the discussion, assume that A has been scaled so that its largest eigenvalue equals 1, and let $\mathbf{e}$ be a small normalized eigenmode of A (i.e., $\|\mathbf{e}\| = 1$). Multiplying the eigenvalue equation $A\mathbf{e} = \lambda\mathbf{e}$ by $\mathbf{e}^T$, we see that a small eigenmode satisfies

$$\lambda = \mathbf{e}^T A\mathbf{e} \ll 1. \tag{4}$$

Since the constant function is geometrically smooth, from our assumption that geometrically smooth functions are in the near null space of A, it is reasonable to assume that A has row sum zero. Then, we can expand $\mathbf{e}^T A\mathbf{e}$ to arrive at

$$\mathbf{e}^T A\mathbf{e} = \sum_{i<j}(-a_{ij})(e_i - e_j)^2 \ll 1. \tag{5}$$

If $-a_{ij} > 0$, then this equation leads us to one of the main heuristics in C-AMG:

> **C-AMG Heuristic:** Smooth error varies slowly in the direction of relatively large (negative) coefficients of the matrix.

The C-AMG heuristic gives us an algebraic way to track geometrically smooth error, but we need to be more specific about what it means to be a "large" coefficient. This leads us to another major concept in the algorithm:

> **Strength of Connection:** Given a threshold $0 < \theta \leq 1$, we say that variable u_i *strongly depends* on variable u_j if
>
> $$-a_{ij} \geq \theta \max_{k \neq i}\{-a_{ik}\}.$$

In other words, strength of connection is measured relative to the largest off-diagonal entry in a row. Note that, in practice, positive off-diagonal connections are considered to be *weak* connections and are ignored in the above. Also note that, with this definition of strength, it is possible that a point i strongly depends on j, but point j only weakly depends on i, even though A is symmetric. There are alternative symmetric definitions used in some algorithms. For simplicity, we assume that strength is symmetric.

Choosing the coarse grid

In C-AMG, the coarse grid is a subset of the fine grid. Points are chosen such that the grid is coarsened in directions of strong connections of the matrix. The procedure for doing this is actually quite simple. In a nutshell, the algorithm consists of three main steps:

1. Define a *strength matrix*, A_s, by deleting weak connections in A;
2. **First pass:** Choose an independent set of fine-grid points based on the graph of A_s;
3. **Second pass:** Choose additional points if needed to satisfy interpolation requirements.

The coarsening procedure partitions the grid into C-points (points on the coarse grid) and F-points (points not on the coarse grid). Figure 3 illustrates the first pass of the algorithm for a 2D Laplacian problem discretized with finite elements on a uniform mesh. The discretization stencil is given by

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

Since all of the off-diagonal coefficients are -1's, the connections in the matrix are all strong connections,

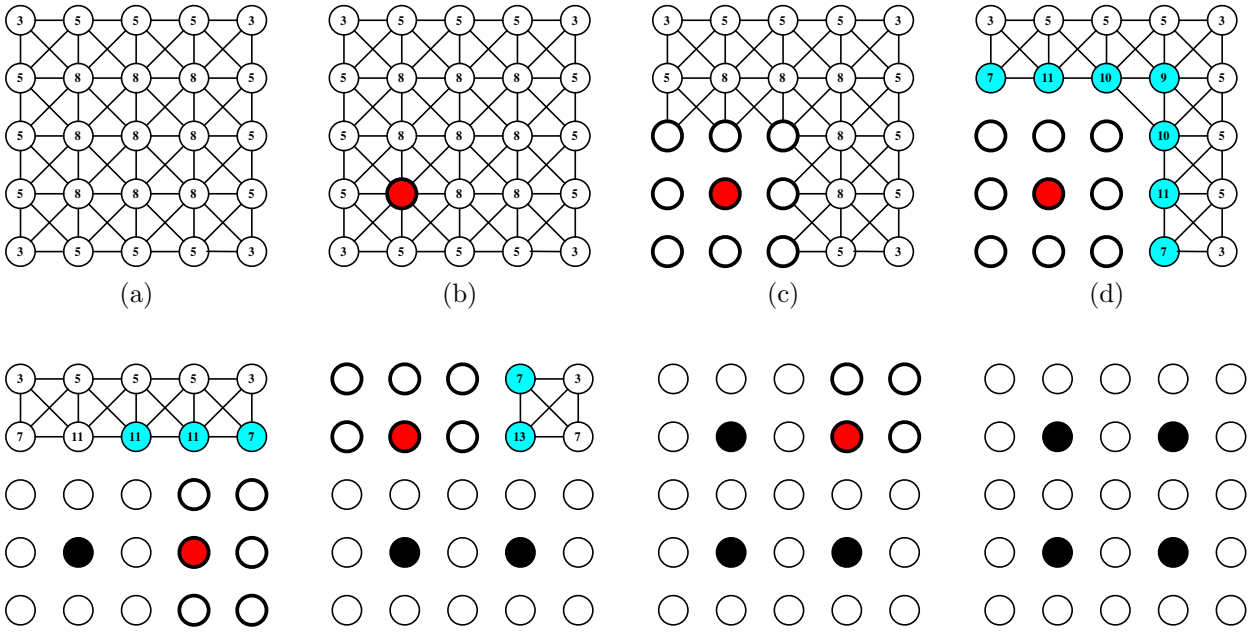


Figure 3: Illustration of the first pass of the C-AMG coarsening algorithm for a 9-point discretization stencil. (a) The nodes of the graph of the strength matrix are assigned a weight equal to the number of off-diagonal connections. (b) A point with maximal weight is chosen as a C-point. (c) The neighbors of the new C-point are set to be F-points. (d) For each new F-point, the weights of its neighbors are increased by one to make them more likely to be chosen next. The algorithm continues in this way until all points are either C- or F-points.

regardless of the parameter choice θ. Hence, A_s and A are the same.

The original C-AMG interpolation scheme (described below) requires each pair of strongly-connected F-points to be strongly connected to a common C-point. The second pass of the coarsening algorithm searches for F-point pairs that do not satisfy this requirement, and changes one of them to a C-point. Researchers later found that the second pass leads to high computational costs, and they have largely abandoned it in favor of other approaches for defining interpolation.

As we saw in the example of Figure 2, the C-AMG coarsening algorithm is able to produce standard fully coarsened and semicoarsened grids, and combinations thereof. The strength matrix is the key to making this happen, but as we see later, it can sometimes be sensitive to the choice of strength parameter θ. Some researchers today are exploring more reliable definitions of strength, while others are exploring completely different coarsening approaches based on so-called *compatible relaxation* that avoid defining strength altogether.

Another area of active research is *parallel* coarsening algorithms. It is easy to see that the algorithm in Figure 3 is inherently sequential. Unfortunately, most parallel coarsening algorithms lead to increased computational costs and often degrade convergence.

Defining interpolation

We again use the fact that smooth error $\mathbf{e}$ is characterized by small eigenmodes. Since the residual $\mathbf{r} = A\mathbf{e}$, we have from (4) that

$$\lambda^2 = \mathbf{e}^T A^2 \mathbf{e} = \mathbf{r}^T \mathbf{r} \ll 1. \tag{6}$$

In other words, smooth error is also characterized by small residuals. To derive interpolation in C-AMG,

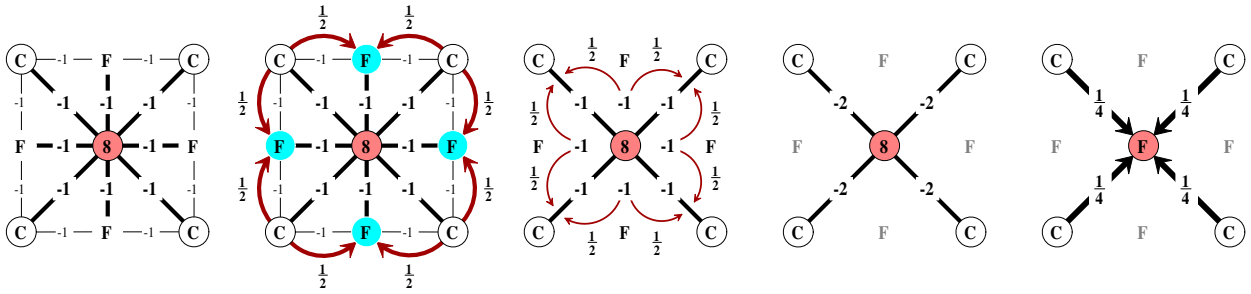


Figure 4: Derivation of C-AMG interpolation for the standard 9-point finite element stencil. In the second image, we assume that strongly-connected F-points are interpolated from neighboring interpolatory points. The weights (all 1/2) are chosen based on the underlying matrix entries such that the constant function is interpolated exactly. In the third image, we "redistribute" the strong F connections according to the interpolation weights in the previous step. This produces the "collapsed stencil" in the fourth image, which leads directly to the interpolation rule in the last image.

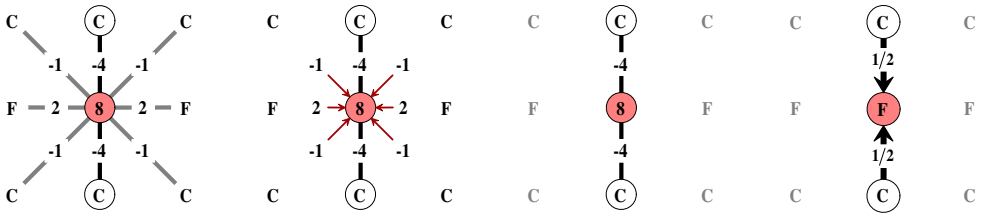


Figure 5: Derivation of C-AMG interpolation for an anisotropic 9-point finite element stencil. In the second image, weak coefficients are added to the diagonal to produce the "collapsed stencil" in the third image, which leads directly to the interpolation rule in the last image

we take this to its extreme and assume that

$$r_i = (A\mathbf{e})_i = 0.$$

If we rewrite this equation at an F-point i in terms of the coefficients of A, some regrouping leads to

$$a_{ii}e_i = -\sum_{j\in C_i} a_{ij}e_j - \sum_{j\in F_i^s} a_{ij}e_j - \sum_{j\in N_i^w} a_{ij}e_j, \quad (7)$$

where the sets C_i, F_i^s, and N_i^w are defined as follows:

$$\begin{array}{lcl} C_i & : & C\text{-points strongly connected to } i \\ F_i^s & : & F\text{-points strongly connected to } i \\ N_i^w & : & \text{all points weakly connected to } i. \end{array} \quad (8)$$

The set C_i is the set of *interpolatory points*. That is, these are the points that F-point i will interpolate from. The trick to deriving interpolation is to rewrite the e_j in the last two terms of (7) in terms of either the interpolatory points in C_i or the F-point i. This produces an equation that involves only the F-point and its interpolatory points, which we can use directly to define interpolation. This process is sometimes referred to as "collapsing the stencil", and it is illustrated in Figures 4 and 5 for two finite element stencils.

The stencil in Figure 5 helps to illustrate one of the potential problems with the strength of connection definition used in C-AMG. The stencil comes from a quadrilateral finite element discretization of the Laplacian on a mesh that is highly stretched in the x direction. The resulting problem is strongly anisotropic in the y direction, yet this strong anisotropy is not reflected in the size of the off-diagonal entries. In fact, any value of the strength

Fine Grid	Iterations	Convergence factor	Coarse grids	Grid complexity	Operator complexity	Setup time	Solve time
31×31	9	0.19	5	1.6	1.7	-	-
61×61	10	0.23	6	1.6	1.6	0.01	0.02
121×121	9	0.23	8	1.6	1.7	0.05	0.07
241×241	9	0.23	9	1.6	1.7	0.25	0.32
481×481	9	0.23	12	1.7	1.7	1.02	1.27
961×961	11	0.29	13	1.7	1.7	4.42	6.28

Table 1: C-AMG results for example problem (3) for different grid sizes with strength threshold $\theta = 0.4$, and with $\nu_1 = \nu_2 = 1$ smoothing steps of C-F Gauss-Seidel. Iterations were done until the relative residual was reduced below 10^{-9}. *Grid complexity* is the total number of grid points on all grids divided by the number of grid points on the fine grid. *Operator complexity* is the total number of nonzeroes in the linear operators on all grids divided by the number of nonzeroes in the fine grid operator. *Setup time* is the time required to choose coarse grids and build interpolation, restriction, and coarse-grid operators.

threshold θ that is less than or equal to 0.25 will turn the corner couplings into strong connections. The resulting interpolation has 6 interpolatory points instead of 2, and degrades the convergence of C-AMG.

Numerical performance

To give an illustration of the numerical performance of C-AMG, consider again the example problem in (3) and Figure 2. Table 1 shows single-processor results on an Intel Pentium workstation. The coarse grids for the 31×31 problem are shown in the figure. We see that the convergence factors are almost uniform independent of problem size, the growth in both setup and solve time is essentially linear with problem size, and the number of grid levels grows logarithmically with problem size. These are expected characteristics of multigrid methods. We also see that the grid and operator complexities stay nicely bounded for this problem (growth in operator complexity is often an issue for AMG, especially in parallel). Note that in practice, it is usually better to use AMG as a preconditioner for a Krylov method such as conjugate gradients (CG). To precondition CG, we first must ensure that the AMG cycle is symmetric. If we do that for this problem by using C-F Jacobi, the resulting AMG-CG method takes 8 or 9 iterations for all problem sizes. See [6, 8] for a more extensive set of numerical experiments.

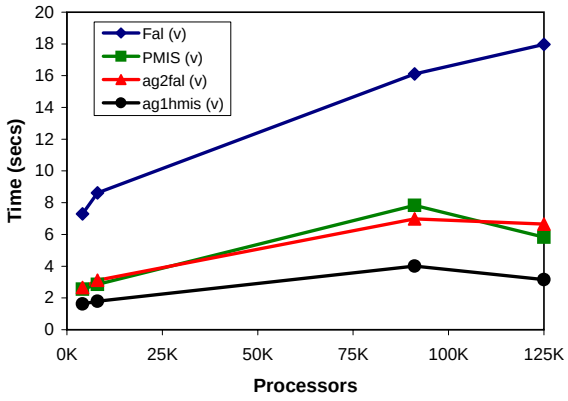


Figure 6: AMG scaling results on BlueGene/L for four different parallel coarsening schemes. The problem is a 3D Laplacian discretized with 7-point finite differences such that each processor has a $25 \times 25 \times 25$ piece of the grid. The largest problem has nearly 2 billion unknowns distributed across 125K processors.

In Figure 6, we show scaling results[1] for a parallel variant of C-AMG, and illustrate the dramatic effect that coarsening algorithms can have on parallel performance at large processor counts. The algorithm indicated by diamonds is the most similar to C-AMG. Here, each processor uses the C-AMG

[1] Courtesy of Ulrike Yang at Lawrence Livermore National Laboratory.

coarsening algorithm on the interior of its piece of the grid, then does something special to complete the coarsening along processor boundaries. The second pass and the non-sequential nature of the parallel algorithm conspire to increase complexities and slow parallel performance. The algorithm indicated by circles achieves decent scaling results by controlling complexity through more aggressive coarsening and the use of long-range interpolation with no second pass. Note that non-optimal methods like CG would be orders of magnitude slower than any of the curves in the figure.

Other AMG Algorithms

Although the classical C-AMG method of the previous section works remarkably well for a wide variety of problems, some of the assumptions made in its derivation limit its applicability. There have been many other AMG algorithms that have been developed to extend the applicability of AMG to new classes of problems. We mention a few of them here.

The AMGe approach ("e" stands for "element") was first introduced in [9] as a means of improving the robustness of AMG for finite element problems. It differs from standard AMG by requiring access to local element stiffness matrices (in addition to the assembled global stiffness matrix). These stiffness matrices are used to construct effective interpolation operators. Another variant of AMGe is element-free AMGe [10], which constructs its own local stiffness matrices directly from the system matrix. The interpolation for this method can be viewed as a generalization of the C-AMG interpolation described earlier. These AMGe methods have been proven to be robust for such difficult problems as non-grid-aligned anisotropic diffusion and thin-body elasticity. They can suffer though from generally expensive setup costs, both in time and memory, since they require generating coarse element matrices on all levels during the setup.

The smoothed aggregation (SA) method [11, 12] is a highly successful AMG method that has been shown to be robust and efficient over a wide variety of problems. One of the most interesting aspects of SA is its approach for defining interpolation. In all of the methods we have discussed so far, interpolation is viewed (and constructed) as rows of P, i.e., we think of interpolation as being "to point i". However, if $\mathbf{p}_j$ are the columns of P, then interpolation of a coarse-grid vector $\mathbf{e}_c$ can be written as

$$P\mathbf{e}_c = \sum_j e_{c,j}\mathbf{p}_j.$$

That is, we can think of interpolation as being a linear combination of basis functions $\mathbf{p}_j$. The SA algorithm takes this view and builds a set of sparse (local) basis functions from a given small set of near null space components. To see how this works, consider the case where we have a single near null space component $\mathbf{x}$ (e.g., the constant function). The SA algorithm first partitions the grid by "aggregating" grid points into small disjoint sets. It then builds the so-called tentative prolongator (interpolation) so that the nonzeroes of column j are the values of $\mathbf{x}$ in aggregate j. Lastly, a smoother is applied to the tentative prolongator to produce the final interpolation operator.

In general, AMG methods (all multigrid methods, actually) must use some additional information characterizing the near null space to be successful. For AMGe, the stiffness matrices are used, while in smoothed aggregation, the near null space components are explicitly required. The recently developed *adaptive AMG* methods are the first methods that do not require this additional information [13, 14]. These methods employ the idea of "using the method to improve the method," and they exhibit the optimal convergence properties of multigrid without requiring a priori knowledge of the near null space. Instead, they automatically "discover" these problematic components and make adjustments for them (i.e., they adapt). The basic adaptive algorithm is as follows:

Initialize the method, E (9a)

Apply E to $A\mathbf{x} = 0$ (9b)

If fast, use E to solve $A\mathbf{u} = \mathbf{f}$ (9c)

Else, use $\mathbf{x}$ to update E and go to (9b) (9d)

The $\mathbf{x}$ that results from step (9b) is called a *prototype*. It is a representative of error that is not damped well by the method E. The key step is (9d), the adaptive step. Understanding how to use information in $\mathbf{x}$ to update E is one of the main research issues for these algorithms.

Theory and New Developments

Most AMG algorithm development is guided by theory. In the AMGe algorithms above, the underlying theory was outlined in [9]. A major component of this theory is the so-called *weak approximation property* that, if satisfied by interpolation, implies convergence of the two-grid algorithm in (2). This approximation property relates the accuracy of interpolation to the spectrum of the system matrix: namely, that eigenmodes with small associated eigenvalue must be interpolated well. The weakness of this theory is that it is limited to simple pointwise smoothers and a particular type of coarse grid.

Recently, we introduced a new theory [15] that allows for general smoothing processes and coarse grids (e.g., vertex-based, cell-based, and agglomeration-based), encompassing a much broader class of problems and algorithms than before. The motivation for developing the new theory was Maxwell's equations, for which pointwise smoothers are not adequate and non-standard coarse grids are often more appropriate. The next two theorems summarize the main convergence results in [15].

Theorem 1 $\|E\|_A^2 \leq 1 - \frac{1}{K}$, *where*

$$K = \sup_{\mathbf{e}} \frac{\|(I - PR)\mathbf{e}\|_{\widetilde{M}}^2}{\|\mathbf{e}\|_A^2}.$$

Theorem 2 $K \leq \eta K_\star$, *where*

$$\eta = \|PR\|_A; \qquad K_\star = \inf_P \sup_{\mathbf{e}} \frac{\|(I - PR)\mathbf{e}\|_{\widetilde{M}}^2}{\|\mathbf{e}\|_A^2}.$$

In the theorems, E is the multigrid operator, $\widetilde{M}$ is an operator derived from the smoother, P is interpolation, and R is a restriction-like operator such that $RP = I$ (so that PR is a projection onto range(P)). We think of R as defining the *coarse-grid variables*, i.e., $\mathbf{u}_c = R\mathbf{u}$.

Theorem 1 gives conditions that P must satisfy in order to achieve a fast converging multigrid method. We can see that to make K small, then small eigenmodes must either be interpolated well by P (since the denominator is small for these eigenmodes) or they must be handled by the smoother.

Theorem 2 bounds K by two new constants, η and $K_\star$. The significance of this theorem is that it separates the construction of P into its natural two components: coarse-grid selection and definition of P's coefficients. The constant $K_\star$ is the K in the first theorem for the "best" P possible. Hence, $K_\star$ measures the quality of the coarse grid in some sense, because if it is small, we know there exists an interpolation operator that gives good AMG convergence. Once we have a coarse grid, the expression for η gives us guidance on how to define the coefficients of P in a way that is independent of the relaxation process.

To ensure that $K_\star$ is bounded in practice, we can use compatible relaxation (CR). The notion of compatible relaxation was introduced by Brandt in [16] as a modified relaxation scheme that keeps the coarse-grid variables invariant. Brandt stated that the convergence rate of CR is a general measure for the quality of the set of coarse variables. In [15], we proved that fast convergence of CR implies a small $K_\star$ (a good coarse grid). Based on this work, we developed an algorithm for selecting coarse grids [17]. To date, we have considered only the case where the coarse grid is chosen as a subset of the fine grid variables. This is the classic C-AMG approach. The algorithm does not use fragile notions of strength-of-connection, and it naturally complements the smoother used in the AMG method. A similar method was developed in [18].

Most recently, we have developed a new sharp theory [19] that gives necessary and sufficient conditions for two-grid convergence and provides additional insight for the development of AMG methods. The sharp theory is very similar in form to Theorem 1, and we have begun to use this similarity to develop more predictive CR methods (better predictors of AMG convergence) that may play an important role in future adaptive AMG methods.

References

[1] R. E. Alcouffe, A. Brandt, J. E. Dendy, and J. W. Painter. The multi–grid method for the diffusion equation with strongly discontinuous coefficients. *SIAM J. Sci. Stat. Comput.*, 2:430–454, 1981.

[2] A. Brandt, S. F. McCormick, and J. W. Ruge. Algebraic multigrid (AMG) for sparse matrix equations.

In D. J. Evans, editor, *Sparsity and Its Applications*. Cambridge University Press, Cambridge, 1984.

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

[4] J. W. Ruge and K. Stüben. Algebraic multigrid (AMG). In S. F. McCormick, editor, *Multigrid Methods*, volume 3 of *Frontiers in Applied Mathematics*, pages 73–130. SIAM, Philadelphia, PA, 1987.

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

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

[7] U. M. Yang. Parallel algebraic multigrid methods - high performance preconditioners. In A. M. Bruaset and A. Tveito, editors, *Numerical Solution of Partial Differential Equations on Parallel Computers*, volume 51 of *Lecture Notes in Computational Science and Engineering*, pages 209–233. Sprinter-Verlag, 2006.

[8] A. J. Cleary, R. D. Falgout, V. E. Henson, J. E. Jones, T. A. Manteuffel, S. F. McCormick, G. N. Miranda, and J. W. Ruge. Robustness and scalability of algebraic multigrid. *SIAM J. Sci. Comput.*, 21(5):1886–1908, 2000.

[9] 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(5):1570–1592, 2000.

[10] V. E. Henson and P. S. Vassilevski. Element-free AMGe: general algorithms for computing interpolation weights in AMG. *SIAM J. Sci. Comput.*, 23(2):629–650, 2001.

[11] P. Vaněk. Acceleration of convergence of a two-level algorithm by smoothing transfer operator. *Applications of Mathematics*, 37:265–274, 1992.

[12] P. Vaněk, J. Mandel, and M. Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, 56:179–196, 1996.

[13] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge. Adaptive smoothed aggregation (αSA) multigrid. *SIAM Review: SIGEST*, 47(2):317–346, 2005.

[14] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge. Adaptive algebraic multigrid. *SIAM J. Sci. Comput.*, 27(4):1261–1286, 2006.

[15] R. D. Falgout and P. S. Vassilevski. On generalizing the AMG framework. *SIAM J. Numer. Anal.*, 42(4):1669–1693, 2004.

[16] A. Brandt. General highly accurate algebraic coarsening. *Electronic Transactions on Numerical Analysis*, 10:1–20, 2000.

[17] J. J. Brannick and R. D. Falgout. Compatible relaxation and coarsening in algebraic multigrid. *SIAM J. Sci. Comput.*, in preparation.

[18] O. E. Livne. Coarsening by compatible relaxation. *Num. Lin. Alg. Appl.*, 11(2–3):205–227, 2004. Special issue on Multigrid Methods.

[19] R. D. Falgout, P. S. Vassilevski, and L. T. Zikatanov. On two-grid convergence estimates. *Numer. Linear Algebra Appl.*, 12(5–6):471–494, 2005.