An Introduction to Algebraic Multigrid

Algebraic multigrid (AMG) solves linear systems based on multigrid principles, but in a way that depends only on the coefficients in the underlying matrix.

Multigrid methods are called *scalable* or *optimal* because they can solve a linear system with N unknowns with only $O(N)$ work. Because this work can be effectively distributed across a parallel machine, multigrid methods can 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 through two complementary processes: *smoothing* and *coarse-grid correction*. Smoothing involves the application of a smoother (also called a *relaxation method*). Coarse-grid correction involves transferring information to a coarser grid through *restriction*, solving a coarse-grid system of equations, and then transferring the solution back to the fine grid through *interpolation* (also called *prolongation*). In the classical geometric multigrid setting (see Irad Yavneh's article on p. 12), smoothing reduces (oscillatory) high-frequency error, whereas coarse-grid correction eliminates (smooth) low-frequency error. Although this geometric interpretation of multigrid was critical to the method's early development and still plays an important role in simulation, geometric techniques fall short for some problem classes (see the "Operator-Dependent Interpolation and Algebraic Multigrid" sidebar).

Algebraic multigrid is a method for solving linear systems based on multigrid principles, but requires no explicit knowledge of the problem geometry.[1–3] AMG determines coarse grids, intergrid transfer operators, and coarse-grid equations based solely on the matrix entries. Since the method's introduction, researchers have developed numerous AMG algorithms with different robustness and efficiency properties that target a variety of problem classes. This article introduces AMG methods, beginning with a description of the classical algorithm of Achi Brandt, Steve McCormick, John Ruge, and Klaus Stüben,[1–3] and then moves on to some more recent advances and theoretical developments. More thorough introductions to AMG[4,5] and an overview of parallel AMG methods[6] are available elsewhere.

AMG Basics

Before getting into the specifics of particular AMG algorithms, it's important to set the stage by introducing the method's basic ingredients. Because AMG is a matrix-based method, let us start with a purely linear algebra viewpoint.

We're interested in solving the linear system

1521-9615/06/$20.00 © 2006 IEEE
Copublished by the IEEE CS and the AIP

ROBERT D. FALGOUT
Lawrence Livermore National Laboratory

Operator-Dependent Interpolation and Algebraic Multigrid

Researchers introduced operator-dependent interpolation in 1981 to solve diffusion problems with large jumps in coefficients.[1] Unlike geometric interpolation techniques, operator-dependent interpolation accounts for 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. Figure A shows the two problems, together with the ith discrete equations for their differential terms.

Linear interpolation works well for the problem in Figure A1, but not for the problem in Figure A2 if k has large jumps. However, as Figure B shows, we can derive an appropriate interpolation for the problem in Figure A2 by noticing that it's equivalent to the problem in Figure A1 if the grid spacing satisfies $h_{i-1/2} = h/k_{i-1/2}$.

Figure B shows that geometric information alone isn't 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 Figure A to the ith matrix equation, $(\mathbf{Au})_i = a_{i,i-1}u_{i-1} + a_{i,i}u_i + a_{i,i+1}u_{i+1}$, we can write both linear and operator-dependent interpolation entirely in terms of matrix coefficients:

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

Reference

1. R.E. Alcouffe et al., "The Multi-Grid Method for the Diffusion Equation with Strongly Discontinuous Coefficients," *SIAM J. Scientific and Statistical Computing*, vol. 2, no. 4, 1981, pp. 430–454.

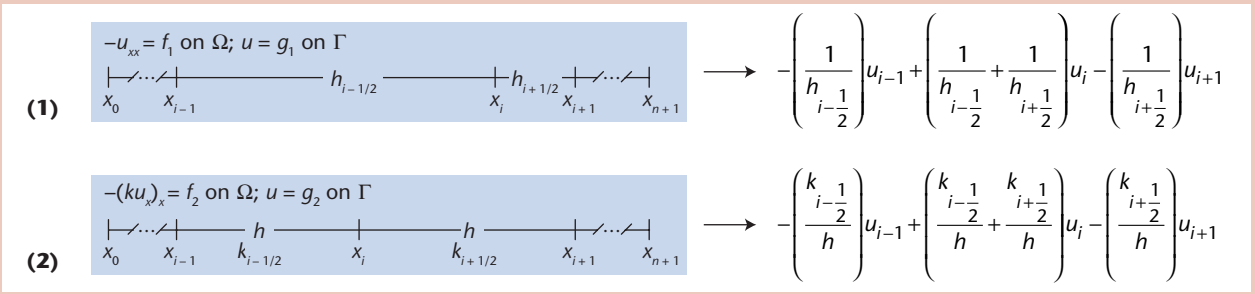

Figure A. One-dimensional problems discretized with piecewise-linear finite elements. (1) A constant-coefficient problem on a variably spaced grid and (2) a variable-coefficient problem on a uniform grid.

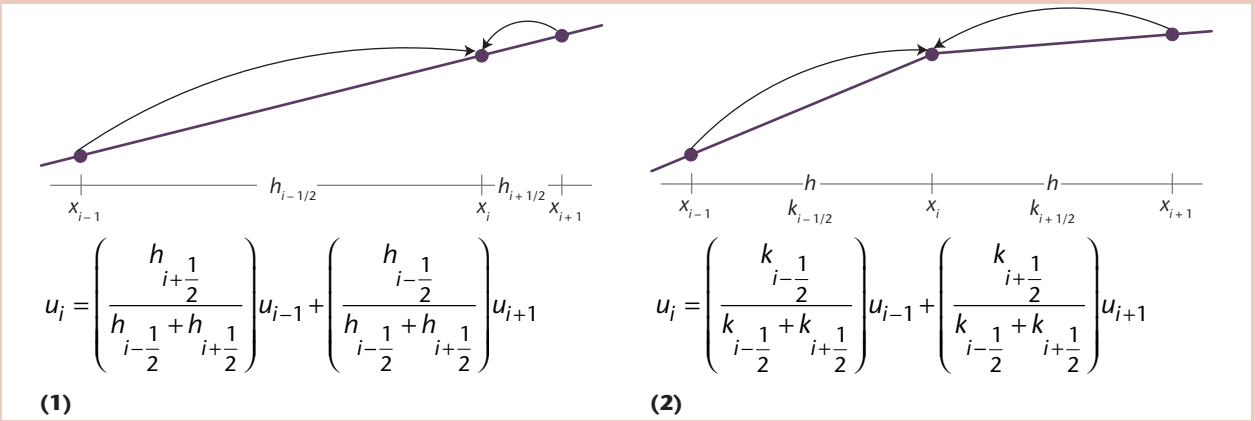

Figure B. Interpolation for the problems in Figure A. (1) Linear interpolation for Figure A1 and (2) operator-dependent interpolation for Figure A2.

$$\mathbf{Au} = \mathbf{f}, \tag{1}$$

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

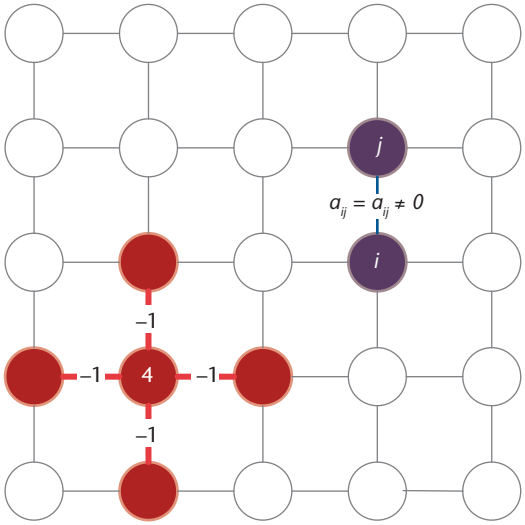

Figure 1. Matrix adjacency graph. This matrix adjacency graph is for a five-point discretization of the Laplace equation on a 5 × 5 uniform grid. The edges between points *i* and *j* correspond to nonzero entries a_{ij} in the matrix. The discretization stencil (bottom left) is just a row in the matrix.

between fine and coarse grids, which are denoted in linear algebra terms simply as the vector space $\Re^n$ and the lower-dimensional (coarse) vector space $\Re^{n_c}$. Interpolation (prolongation) maps the coarse grid to the fine grid and is just the $n \times n_c$ matrix $\mathbf{P}$: $\Re^{n_c} \rightarrow \Re^n$. Restriction maps the fine grid to the coarse grid and is the transpose of interpolation ($\mathbf{P}^T$) in this article. The two-grid method for solving Equation 1 is then defined as follows:

Do ν_1 smoothing steps on $\mathbf{Au} = \mathbf{f}$. (2a)

Compute residual $\mathbf{r} = \mathbf{f} - \mathbf{Au} = \mathbf{Ae}$. (2b)

Solve $\mathbf{A}_c\mathbf{e}_c = \mathbf{P}^T\mathbf{r}$. (2c)

Correct $\mathbf{u} \leftarrow \mathbf{u} + \mathbf{Pe}_c$. (2d)

Do ν_2 smoothing steps on $\mathbf{Au} = \mathbf{f}$. (2e)

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

Adding Geometry to the Discussion

Although AMG aims to solve matrix equations using multigrid principles, it's difficult to intuitively understand the method from a purely matrix viewpoint. Because many of the problems we want to solve come from discretized partial differential equations (PDEs), one of the best approaches for visualizing AMG is to relate the matrix equations back to an underlying PDE. I rely heavily on this technique in this article. In fact, my descriptions use PDEs on 2D structured grids almost exclusively. Although the illustrations contain geometry, remember that AMG doesn't actually use any geometric information.

The matrix adjacency graph 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 matrix $\mathbf{A}$ (see Figure 1). The grid in AMG is simply the set of vertices in the graph—that is, grid point *i* is just vertex *i*. If the linear system comes from a PDE's discretization, we can draw the grid points in their actual geometric locations along with the associated graph. Figure 1 illustrates this 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. An error not eliminated by the smoother is called a *smooth error*, and must be handled by coarse-grid correction (recall the algorithm in Equation 2). In the classical geometric multigrid setting, a smooth error is smooth in the usual geometric sense. In the AMG setting, however, a smooth error might be geometrically oscillatory. Researchers in AMG often use the term "algebraically smooth" to clarify the distinction.

Consider the following simple 2D example, discretized by finite elements on a uniform mesh:

$$\begin{aligned} -au_{xx} - bu_{yy} &= f \quad \text{on } \Omega = [0,1] \times [0,1] \\ u &= g \quad \text{on } \Gamma \\ a &= b \quad \text{if } x < 1/2 \\ a &>> b \quad \text{if } x \geq 1/2. \end{aligned} \tag{3}$$

Figures 2a, 2b, and 2c show the error after seven Gauss-Seidel iterations. The error is geometrically smooth in both the *x* and *y* directions in the left half of the plane, where the problem is isotropic, but it's geometrically oscillatory in the *y* direction in the

right half of the plane, where the problem is anisotropic. As Figure 2d illustrates, AMG coarsens in directions of geometric smoothness. That is, in the plane's left half, the grid is coarsened in both directions (*full coarsening*), but in the right half, the grid is coarsened only in the x direction (*semicoarsening*). AMG's ability to "follow the physics" during coarse-grid correction is another of its advantages over geometric approaches.

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

Consider, for example, the PDE in Equation 3. Any linear function u is in the kernel of the differential operator because 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 Figure 2 shows. This example underscores the distinction between smooth error (error 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, whereas the near-null space contains only geometrically smooth functions. For applications in which the near-null space contains geometrically oscillatory functions (such as electromagnetics), the approach of coarsening in directions of geometric smoothness isn't sufficient.

So, how can we use knowledge of the near-null space to design a real AMG algorithm? Researchers have used many approaches, some of which I describe next.

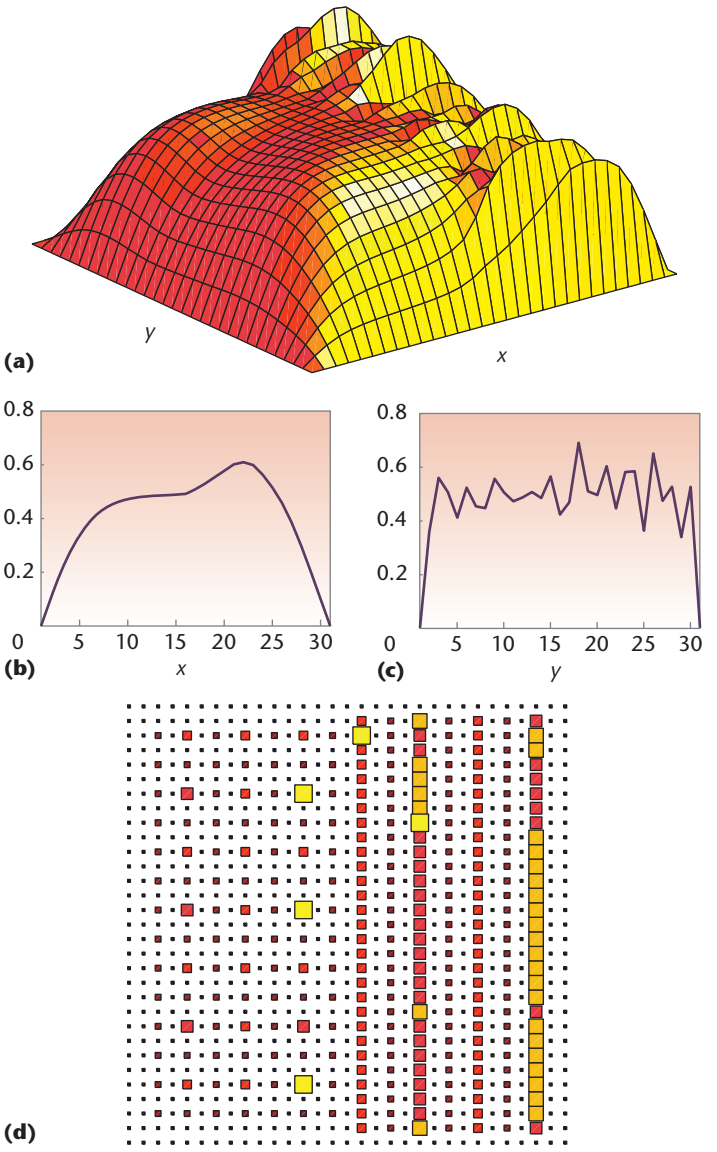

Figure 2. Algebraic smoothness. (a) Error for the problem in Equation 3 after seven sweeps of Gauss-Seidel. (b) The error is geometrically smooth in the *x* direction, (c) but oscillatory in the *y* direction in the plane's right half. (d) In this example, AMG coarsens the grid in directions of geometric smoothness. Larger squares correspond to coarser grids.

Classical AMG

The classical AMG algorithm (C-AMG) is based on the assumption that geometrically smooth functions are in the near-null space of **A**. Because AMG knows nothing about the problem's geometry, we need to characterize this assumption in some algebraic way.

To simplify the discussion, assume that we've scaled **A** so that its largest eigenvalue equals 1, and let **e** be a small normalized eigenmode of **A** (that is, $\|\mathbf{e}\| = 1$). Multiplying the eigenvalue equation $\mathbf{Ae} = \lambda\mathbf{e}$ by $\mathbf{e}^T$, we see that a small eigenmode satisfies

$$\mathbf{e}^T\mathbf{Ae} = \lambda \ll 1. \quad (4)$$

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

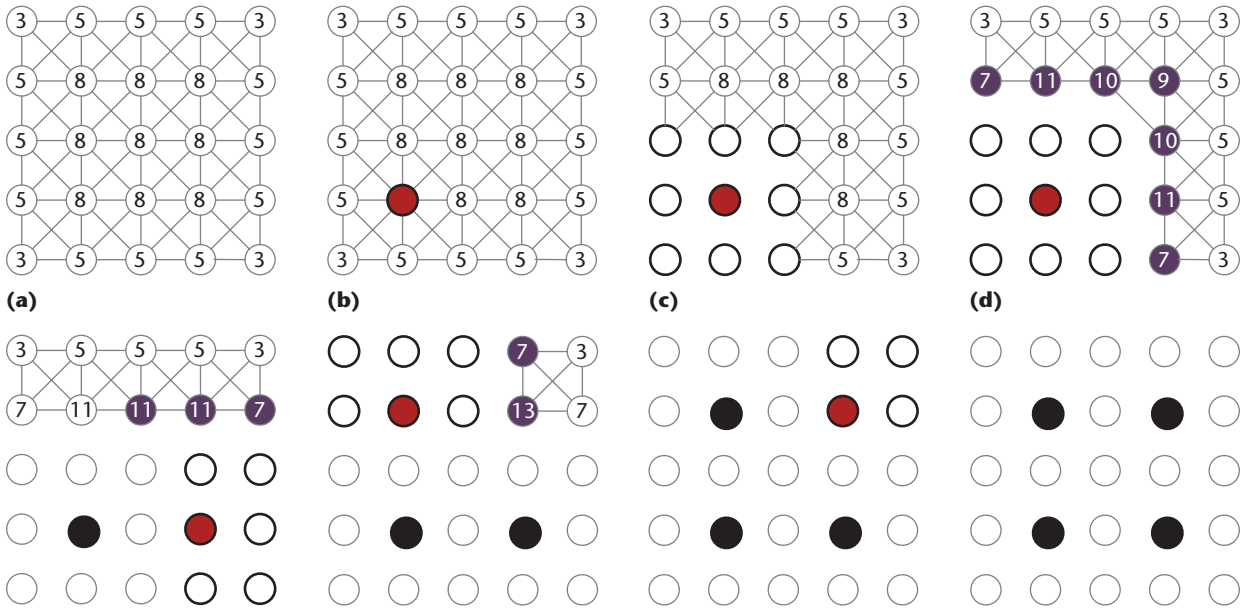

Figure 3. The first pass of the C-AMG coarsening algorithm for a nine-point discretization stencil. (a) The algorithm assigns to the nodes of the strength matrix graph a weight equal to the number of off-diagonal connections. (b) A point with maximal weight is chosen as a *C*-point. (c) The algorithm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

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

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

C-AMG smoothness heuristic. Smooth error varies slowly in the direction of relatively large (negative) coefficients of the matrix.

This heuristic gives us an algebraic way to track geometrically smooth errors, 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 $0 < \theta \leq 1$, *variable* u_i *strongly depends on variable* u_j *if*

$$-a_{ij} \geq \theta \max_{k \neq i} \{-a_{ik}\}.$$

In other words, C-AMG measures strength of connection relative to the largest off-diagonal entry in a row. In practice, C-AMG considers positive off-diagonal entries to be weak connections. It is interesting to note that, with this definition of strength, a point i could strongly depend on j while point j only weakly depends on i, even though $\mathbf{A}$ is symmetric. Some algorithms use alternative symmetric definitions. For simplicity, I assume that strength is symmetric.

Choosing the Coarse Grid

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

1. Define a strength matrix, $\mathbf{A}_s$, by deleting weak connections in $\mathbf{A}$.
2. *First pass*: choose an independent set of fine-grid points based on the graph of $\mathbf{A}_s$.
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

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

Because all the off-diagonal coefficients are –1s, the connections in the matrix are all strong connections, regardless of the parameter choice θ. Hence, $\mathbf{A}_s$ and $\mathbf{A}$ are the same.

The original C-AMG interpolation scheme (which I describe next) 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've largely abandoned it in favor of other approaches for defining interpolation.

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

Another area of active research is parallel coarsening algorithms. The algorithm in Figure 3 is obviously inherently sequential. Unfortunately, most parallel coarsening algorithms increase computational costs and often degrade convergence.

Defining Interpolation

We again use the fact that smooth error $\mathbf{e}$ is characterized by small eigenmodes. Because the residual $\mathbf{r} = \mathbf{Ae}$, we have similarly to Equation 4 that

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

In other words, smooth error is also characterized by small residuals. To derive interpolation in C-AMG, we take this to its extreme and assume that $r_i = (\mathbf{Ae})_i = 0$.

If we rewrite this equation at an *F*-point i in terms of the coefficients of $\mathbf{A}$, 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:

C_i: *C*-points strongly connected to i.

F_i^s: *F*-points strongly connected to i.

N_i^w: all points weakly connected to i.

C_i is the set of *interpolatory points*—the points from which *F*-point i will interpolate. The trick to deriving interpolation is to rewrite the e_j in the last two terms of Equation 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." Figures 4 and 5 illustrate the process for two finite element stencils.

The stencil in Figure 5 helps illustrate one of the potential problems with C-AMG's strength of connection definition. 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 the size of the off-diagonal entries doesn't reflect this strong anisotropy. In fact, any value of the strength threshold θ that is less than or equal to 0.25 will turn the corner couplings into strong connections. The resulting interpolation has six interpolatory points instead of two and degrades C-AMG convergence.

Numerical Performance

To illustrate C-AMG's numerical performance, consider again the example problem in Equation 3 and Figure 2. Table 1 shows single-processor results on a 2.66-GHz Intel Xeon workstation. The figure shows the coarse grids for the 31×31 problem. The convergence factors are almost uniform across problem sizes, 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. The grid and operator complexities stay nicely bounded for this problem (growth in operator complexity is often an issue for AMG, especially in parallel).

In practice, it's 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 eight or nine iterations for all problem sizes. A more extensive set of numerical experiments is available elsewhere.[5,7]

Figure 6 shows scaling results (courtesy of Ulrike

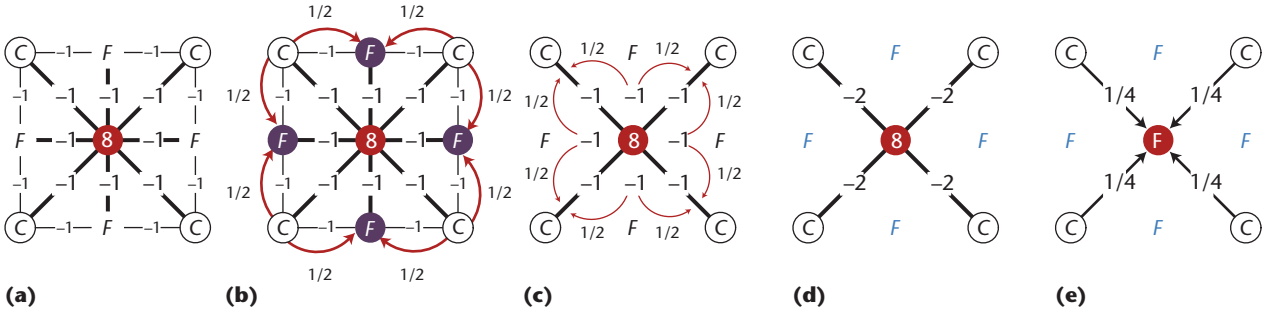

Figure 4. Derivation of C-AMG interpolation for (a) the standard nine-point finite element stencil. (b) We assume that strongly connected *F*-points are interpolated from neighboring interpolatory points. We choose the weights (all 1/2) based on the underlying matrix entries such that the constant function is interpolated exactly. (c) We redistribute the strong *F* connections according to the interpolation weights in the previous step. The redistribution produces (d) a collapsed stencil, which leads directly to (e) the interpolation rule.

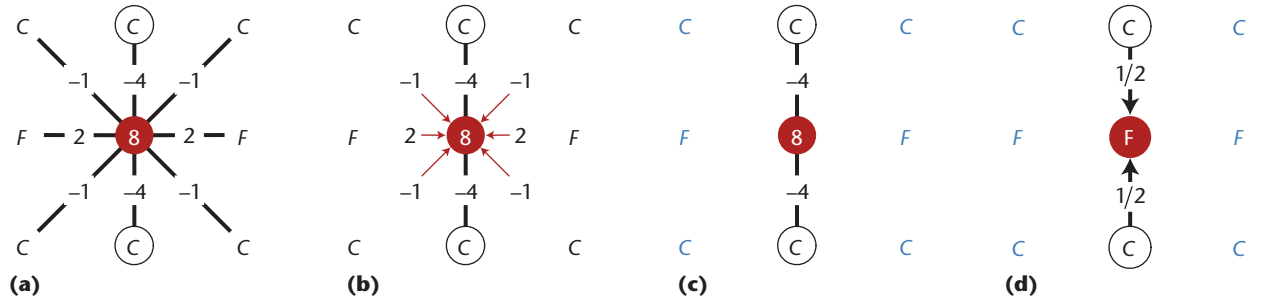

Figure 5. Derivation of C-AMG interpolation for (a) an anisotropic nine-point finite element stencil. (b) We add weak coefficients to the diagonal to produce (c) the collapsed stencil, which leads directly to (d) the interpolation rule.

Yang at Lawrence Livermore National Laboratory) for a parallel variant of C-AMG and illustrates 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 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 parallel algorithm's nonsequential nature 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 using long-range interpolation with no second pass. Nonoptimal methods such as CG would be orders of magnitude slower than any of the curves in the figure.

Other AMG Algorithms

Although the classical C-AMG method works remarkably well for a wide variety of problems, some of the assumptions made in its derivation limit its applicability. Many other AMG algorithms extend AMG's applicability to new classes of problems. I mention a few of them here.

Researchers first introduced the AMGe approach (the "e" stands for element) as a means of improving AMG's robustness for finite element problems.[8] It differs from standard AMG by requiring access to local element stiffness matrices (in addition to the assembled global stiffness matrix). AMGe uses these stiffness matrices to construct effective interpolation operators. Another variant is element-free AMGe,[9] which constructs its own local stiffness matrices directly from the system matrix. You can view the interpolation for this method as a generalization of the C-AMG interpolation. These AMGe methods are robust for such difficult problems as non-grid-aligned anisotropic diffusion and thin-body elasticity. However, they can suffer from generally expensive setup costs, both in time and memory, because they require generating coarse element matrices on all levels during the setup.

The smoothed aggregation (SA) method[10,11] is a

Table 1. C-AMG results for the example problem in Equation 3 for different grid sizes.

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

Note: *The code used strength threshold $\theta = 0.4$ with $\nu_1 = \nu_2 = 1$ step of C-F Gauss-Seidel, and iterated until the relative residual was 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.*

highly successful AMG method that is robust and efficient over a wide variety of problems. One of SA's most interesting aspects is its approach to defining interpolation. In all the methods I've discussed so far, interpolation is viewed (and constructed) as rows of **P**—that is, as being "to point *i*." However, if $\mathbf{p}_j$ are **P**'s columns, we can write interpolation of a coarse-grid vector $\mathbf{e}_c$ as $\mathbf{P}\mathbf{e}_c = \Sigma_j e_{c,j}\mathbf{p}_j$. 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. Consider the case in which we have a single near-null space component **x** (for example, the constant function). The SA algorithm first partitions the grid by aggregating grid points into small disjoint sets. It then builds a preliminary interpolation operator called the *tentative prolongator* such that the nonzeroes of column *j* are the values of **x** in aggregate *j*. Lastly, it applies a smoother 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. AMGe uses the stiffness matrices, whereas SA explicitly requires the near-null space components. However, the recently developed *adaptive AMG* methods do not require this additional information.[12,13] Adaptive AMG "uses the method to improve the method," and exhibits the optimal convergence properties of multigrid without requiring a priori knowledge of the near-null space. Instead, these methods automatically discover problematic components and adjust for them (that is, they adapt). The basic adaptive algorithm is as follows:

Initialize the method, **E**. (8a)

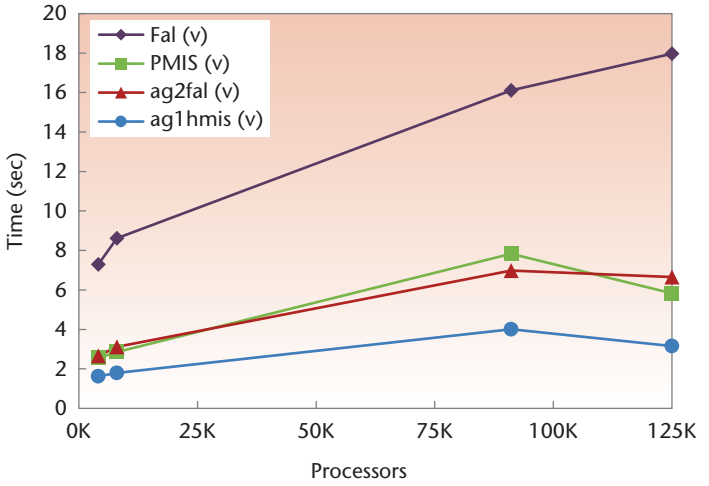

Figure 6. AMG scaling results on BlueGene/L for four parallel coarsening schemes. The problem is a 3D Laplacian discretized with seven-point finite differences such that each processor has a 25 × 25 × 25 piece of the grid. The largest problem has nearly 2 billion unknowns distributed across 125,000 processors.

Apply **E** to **Ax** = 0. (8b)

If fast, use **E** to solve **Au** = **f**. (8c)

Else, use **x** to update **E** and go to Step 8b. (8d)

The **x** resulting from Step 8b is called a *prototype*. It represents an error that isn't damped well by method **E**. The key is Step 8d, the adaptive step. Understanding how to use information in **x** to update **E** is one of the main research issues for these algorithms.

Theory and New Developments

Theory guides most AMG algorithm development.

A major component of the underlying theory of AMGe algorithms[8] is the *weak approximation* property that, if satisfied by interpolation, implies convergence of the two-grid algorithm in Equation 2. This approximation property relates the accuracy of interpolation to the spectrum of the system matrix—namely, that eigenmodes with small associated eigenvalues must be interpolated well. However, the theory is limited to simple pointwise smoothers and a particular type of coarse grid.

Recently, Panayot Vassilevski and I introduced a new theory that allows for general smoothing processes and coarse grids (for example, vertex-, cell-, and agglomeration-based), encompassing a broader class of problems and algorithms than previous theories.[14] The motivation for developing the new theory was Maxwell's equations, for which pointwise smoothers are inadequate and nonstandard coarse grids are often more appropriate. The following two theorems summarize our main convergence results.[14]

Theorem 1: $\|\mathbf{E}\|_{\mathbf{A}}^2 \le 1 - \frac{1}{K}$, where

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

Theorem 2: $K \le \eta K_*$, where

$$\eta = \|\mathbf{PR}\|_{\mathbf{A}}; K_* = \inf_{\mathbf{P}} \sup_{\mathbf{e}} \frac{\|(\mathbf{I} - \mathbf{PR})\mathbf{e}\|_{\tilde{\mathbf{M}}}^2}{\|\mathbf{e}\|_{\mathbf{A}}^2}.$$

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

Theorem 1 gives conditions that $\mathbf{P}$ must satisfy to achieve a fast converging multigrid method. To make K small, either $\mathbf{P}$ must accurately interpolate small eigenmodes (because the denominator is small for these eigenmodes), or the smoother must handle them.

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

To ensure that K_* is bounded in practice, we can use compatible relaxation (CR). Brandt introduced CR as a modified relaxation scheme that keeps the coarse-grid variables invariant.[15] He stated that CR's convergence rate is a general measure for the quality of the set of coarse variables. Vassilevski and I proved that fast convergence of CR implies a small K_* (a good coarse grid). Based on this work, we developed an algorithm for selecting coarse grids.[16] To date, we've considered only the case in which the coarse grid is chosen as a subset of the fine grid variables. This is the classic C-AMG approach. The algorithm doesn't use fragile notions of strength of connection, and it naturally complements the smoother used in the AMG method. Oren Livne developed a similar method.[17]

Most recently, Vassilevski, Ludmil Zikatanov, and I have developed a new sharp theory that gives necessary and sufficient conditions for two-grid convergence and provides additional insight for the development of AMG methods.[18] The sharp theory is similar in form to Theorem 1, and we've begun to use this similarity to develop more predictive CR methods (better predictors of AMG convergence) that might play an important role in future adaptive AMG methods. CiSE

Acknowledgments

This work was performed under the auspices of the US Department of Energy by University of California Lawrence Livermore National Laboratory under contract W-7405-Eng-48.

References

1. A. Brandt, S.F. McCormick, and J.W. Ruge, "Algebraic Multigrid (AMG) for Sparse Matrix Equations," *Sparsity and Its Applications*, D.J. Evans, ed., Cambridge Univ. Press, 1984, pp. 257–284.
2. A. Brandt, "Algebraic Multigrid Theory: The Symmetric Case," *Applied Mathematics and Computation*, vol. 19, nos. 1–4, 1986, pp. 23–56.
3. J.W. Ruge and K. Stüben, "Algebraic Multigrid (AMG)," *Multigrid Methods*, S.F. McCormick, ed., SIAM Press, 1987, pp. 73–130.
4. W.L. Briggs, V.E. Henson, and S.F. McCormick, *A Multigrid Tutorial*, 2nd ed., SIAM Press, 2000.
5. U. Trottenberg, C.W. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, 2001.
6. U.M. Yang, "Parallel Algebraic Multigrid Methods—High Performance Preconditioners," *Numerical Solution of Partial Differential Equations on Parallel Computers*, LNCS 51, A.M. Bruaset and A. Tveito, eds., Springer-Verlag, 2006, pp. 209–233.

7. A.J. Cleary et al., "Robustness and Scalability of Algebraic Multigrid," *SIAM J. Scientific Computing*, vol. 21, no. 5, 2000, pp. 1886–1908.
8. M. Brezina et al., "Algebraic Multigrid Based on Element Interpolation (AMGe)," *SIAM J. Scientific Computing*, vol. 22, no. 5, 2000, pp. 1570–1592.
9. V.E. Henson and P.S. Vassilevski, "Element-Free AMGe: General Algorithms for Computing Interpolation Weights in AMG," *SIAM J. Scientific Computing*, vol. 23, no. 2, 2001, pp. 629–650.
10. P. Vaněk, "Acceleration of Convergence of a Two-Level Algorithm by Smoothing Transfer Operator," *Applications of Mathematics*, vol. 37, 1992, pp. 265–274.
11. P. Vaněk, J. Mandel, and M. Brezina, "Algebraic Multigrid by Smoothed Aggregation for Second and Fourth Order Elliptic Problems," *Computing*, vol. 56, 1996, pp. 179–196.
12. M. Brezina et al., "Adaptive Smoothed Aggregation (αSA) Multigrid," *SIAM Rev.*, vol. 47, no. 2, 2005, pp. 317–346.
13. M. Brezina et al., "Adaptive Algebraic Multigrid," *SIAM J. Scientific Computing*, vol. 27, no. 4, 2006, pp. 1261–1286.
14. R.D. Falgout and P.S. Vassilevski, "On Generalizing the AMG Framework," *SIAM J. Numerical Analysis*, vol. 42, no. 4, 2004, pp. 1669–1693.
15. A. Brandt, "General Highly Accurate Algebraic Coarsening," *Electronic Trans. Numerical Analysis*, vol. 10, 2000, pp. 1–20.
16. J.J. Brannick and R.D. Falgout, "Compatible Relaxation and Coarsening in Algebraic Multigrid," in preparation.
17. O.E. Livne, "Coarsening by Compatible Relaxation," *Numerical Linear Algebra with Applications*, vol. 11, no. 2–3, 2004, pp. 205–227.
18. R.D. Falgout, P.S. Vassilevski, and L.T. Zikatanov, "On Two-Grid Convergence Estimates," *Numerical Linear Algebra with Applications*, vol. 12, nos. 5–6, 2005, pp. 471–494.

Robert D. Falgout *is a computational mathematician at the Center for Applied Scientific Computing (CASC) at the Lawrence Livermore Laboratory; he's also the project leader for the Scalable Linear Solvers Project and its associated software effort, hypre. His work focuses on the development of multilevel methods, as well as the software design and parallel computing issues related to delivering these methods to the scientific simulation community. Falgout has a PhD in applied mathematics from the University of Virginia. He is also on the editorial board for the* SIAM Journal on Scientific Computing *and the* Journal on Numerical Linear Algebra with Applications. *Contact him at rfalgout@llnl.gov.*