# ALGEBRAIC MULTIGRID DOMAIN AND RANGE DECOMPOSITION (AMG-DD/AMG-RD)*

R. BANK† R. D. FALGOUT‡ T. JONES §T. A. MANTEUFFEL§ S. F. MCCORMICK§ J. W. RUGE§

**Abstract.**
In modern large-scale supercomputing applications, Algebraic MultiGrid (AMG) is a leading choice for solving matrix equations. However, the high cost of communication relative to that of computation is a concern for the scalability of traditional implementations of AMG on emerging architectures. This paper introduces two new algebraic multilevel algorithms, Algebraic MultiGrid Domain Decomposition (AMG-DD) and Algebraic MultiGrid Range Decomposition (AMG-RD), that replace traditional AMG V-cycles with a fully overlapping domain decomposition approach. While the methods introduced here are similar in spirit to the *geometric* methods developed by Brandt and Diskin [1], Mitchell [2], and Bank and Holst [3], they differ primarily in that they are purely *algebraic*: AMG-RD and AMG-DD trade communication for computation by forming global composite "grids" based only on the matrix, not the geometry. (As is the usual AMG convention, "grids" here should only be taken in the algebraic sense, regardless of whether or not it corresponds to any geometry.) Another important distinguishing feature of AMG-RD and AMG-DD is their novel residual communication process that enables effective parallel computation on composite grids, avoiding the all-to-all communication costs of the geometric methods. The main purpose of this paper is to study the potential of these two algebraic methods as possible alternatives to existing AMG approaches for future parallel machines. To this end, this paper develops some theoretical properties of these methods and reports on serial numerical tests of their convergence properties over a spectrum of problem parameters.

**Key words.** iterative methods, multigrid, algebraic multigrid, parallel, scalability, domain decomposition

**1. Introduction.** Multigrid methods are often well-suited for large-scale scientific computing problems because they offer the possibility of an $O(N)$ method for solving equations of the form $A_0 u_0 = f_0$, where $A_0$ is an $N \times N$ matrix. However, the challenge for multigrid and other matrix equation solvers on large parallel machines is that performance can suffer from the high cost of communication relative to that of computation. While Algebraic MultiGrid (AMG [4, 5]) solvers scale nearly optimally, they too are increasingly affected by relative communication costs as the number of processors increase. Indeed, all known parallel multigrid algorithms experience $O(log(N))$ communication costs for the main computation in each V-Cycle. Previous *geometric* multilevel approaches developed by [1, 2, 6, 3, 7, 8, 9, 10, 11] aimed to reduce the constant in this $O(log(N))$ cost by trading communication for computation using redundant processing on overlapping grids. Inspired by these efforts, the Algebraic MultiGrid Domain (AMG-DD) and Range Decomposition (AMG-RD) methods introduced here attempt to achieve the same goal by tasking possibly otherwise idle processors to perform redundant computations via a domain decomposition approach;

*Submitted to SIAM Journal of Scientific Computing, June 30, 2014. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-JRNL-666751).

†Department of Mathematics, University of California at San Diego, La Jolla, CA 92093. *email:* rbank@ucsd.edu

‡Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, P.O. Box 808, L-561, Livermore, CA 94551. *email:* rfalgout@llnl.gov

§Department of Applied Mathematics, University of Colorado at Boulder, Boulder, CO 80309–0526 *email:* tobias.jones@colorado.edu, tmanteuf@colorado.edu, stevem@colorado.edu, jruge@colorado.edu

the basic idea is to use subdomains that fully overlap at coarse scales. Our departure from the previous geometric approaches is to exploit the benefits of domain decomposition in a purely *algebraic* AMG setting. AMG-DD and AMG-RD first assume that the setup for an effective AMG implementation has been formed [4, 5]. The two methods then use the *global* AMG hierarchical constructs (coarse grids, coarse-grid operators, and intergrid transfer operators) to create a composite grid for each processor. (Our use of geometric terms such as "grid" and "domain" should be taken in an algebraic sense, as is common in the AMG literature. While these objects may in many cases have an underlying geometry, we make no such assumptions here. No aspects of the methods that we discuss use any notion of, or depend upon, an underlying geometry– the constructs are all purely algebraic. Nevertheless, we use the geometric convention here because it makes for a clearer discussion.) Each composite grid consists of the original grid in and about the subdomain owned by its associated processor and of grids that are increasingly coarse as they extend away from the processor subdomain to the boundary. These composite grids are formed algebraically and directly from the hierarchical constructs determined in the traditional AMG setup phase. In this way, AMG-DD and AMG-RD can be thought of as globally overlapping domain decomposition methods that have reduced communication per cycle, since they do not require communication on each grid level as standard AMG methods do. Moreover, the new process introduced here provides an efficient communication phase between each cycle, in contrast to the expensive all-to-all communication processes of the previous geometric approaches. The composite grids thus provide a means for maintaining effective communication between processors that controls cost and maintains optimal convergence rates.

The main focus of this paper is to study the potential of these AMG-DD and AMG-RD as possible alternatives to existing AMG approaches for solving large-scale matrix equations on advanced parallel machines. To this end, this paper develops some theoretical properties of these methods and reports on serial numerical tests of their convergence properties over a spectrum of parameters on a model problem. Also included are some heuristics and a parameter study based on a performance model, both of which are designed to anticipate the potential of AMG-DD and AMG-RD for use on emerging parallel architectures.

Since AMG-DD and AMG-RD are constructed on top of an existing AMG hierarchy, the cost of the setup, as in all AMG algorithms, needs to be addressed. In a purely serial setting, setup costs for AMG-DD and AMG-RD can be up to about twice that of standard AMG due to the redundant calculation that we use. However, in parallel, owing to the construction of the algorithm from existing operators, the increased setup cost can be reduced essentially to the cost of communicating the components of the operators needed to each processor. This communication can be done using the same pattern as the residual communication and, as such, is bounded in cost by that of performing one extra V-Cycle.

We begin by outlining a traditional AMG implementation and setup, and then specifying how AMG-DD and AMG-RD are constructed. While a wide range of parameters can be used to create the solvers, in the interest of space, we explore just a few choices and how they affect convergence and scalability of the algorithms. We show here that AMG-DD and AMG-RD are algebraic duals of each other in the energy inner product, and we establish convergence of these methods in a two-level setting. (This paper focuses mainly on AMG-DD, but AMG-RD is included because it may be more convenient for some applications and because the two methods together can

be used in sequence to provide a self-adjoint scheme.)

To provide a framework for building models and estimates for scalability, the results in this paper are restricted to determining the effects of the algorithms on the model 2D Poisson equation. We present several numerical results from which we can form models of how parallel AMG and AMG-DD are expected to scale. These results suggest that, for current parallel architectures, the trading of communication for computation that AMG-DD achieves puts it essentially at break even with AMG in terms of parallel efficiency. This parity comes perhaps more from the remarkable parallel efficiency of advanced AMG solvers [12] than any lack of efficiency of the methods developed here. Nevertheless, in addition to their fault tolerance advantages that result from the redundancy of the overlap, AMG-DD (and AMG-RD) can be expected to outperform AMG if communication becomes more costly relative to computation in future hardware.

**2. AMG.** This section gives a very brief overview of the methodology to set the scene for the introduction of our new algorithms. AMG consists of two phases: setup and solution. The setup phase is a one-time cost that is based on the matrix $A_0$, and could be used for multiple right-hand sides. Letting $\Omega_0 = \{i\}$ be the set of indices on the fine grid, then $A_0, u_0, f_0$ are represented on $\Omega_0$ and the setup phase consists of forming a set of coarser grids, $\Omega_1, \ldots, \Omega_L$, with associated operators, $A_1, \ldots, A_L$. The setup also defines the operators $P_0, \ldots, P_{L-1}$ such that $P_i : G_{i+1} \to G_i$, where $G_i$ is the function space associated with $\Omega_i$. The standard Galerkin condition can then be used to form the coarse-grid operators: $A_{i+1} = P_i^T A_i P_i$, where superscript $T$ denotes matrix transpose. The details of how to form these operators are covered in [4, 5]. The solve phase then consists of cycles, and utilizes the complementary processes of relaxation and coarse-grid correction to iterate and improve the solution.

**Algorithm 2.1** $u = \text{VCycle}(\nu_1, \nu_2, u_i, f_i)$ for solving $A_i u_i = f_i$ , where $L$ is the coarsest grid

**if** $i == L$ **then**
  Solve $A_L u_L = f_L$.
**else**
  Relax $\nu_1$ times on $A_i u_i = f_i$.
  Set $f_{i+1} = P_i^T (f_i - A_i u_i)$.
  Call $u_{i+1} \leftarrow \text{VCycle}(\nu_1, \nu_2, 0, f_{i+1})$.
  Update $u_i \leftarrow u_i + P_i u_{i+1}$.
  Relax $\nu_2$ times on $A_i u_i = f_i$.
**end if**
**return** $u_i$

## 3. AMG-DD/AMG-RD.

**3.1. Overview of Algorithms.** The goal of AMG-DD/AMG-RD is to replace the V-cycles used in standard AMG with domain decomposition cycles that require less communication. The assumption is that the setup phase of the AMG process has been completed, forming $A_i, P_i, \Omega_i$. Thus, a traditional AMG setup process has already formed a partitioning of the original fine grid, $\Omega_0$, across all the processors, $p = 1, 2, \ldots, N_p$.

Let $\mathcal{D}_0^p$ denote the indices of $\Omega_0$ assigned to processor $p$. (When it is clear which processor we are referring to, we drop the superscript and write just $\mathcal{D}_0$, and similarly

for other notation.) After the AMG setup is complete, each processor owns the region of the fine grid it is assigned, as well as any of the points in that region that are repeated on any of the coarser grids ($\mathcal{D}_0 \cap \Omega_k$, $k = 1, \ldots, L$).

To understand the AMG-DD algorithm, we first consider the ideal setting in which each processor separately solves the original problem, $A_0 u_0 = f_0$, on the original fine grid, $\Omega_0$. Each processor is assigned its region, $\mathcal{D}_0$. Now letting each processor solve the original problem $A_0 u_p = f_0$, the final solution $u$ would be given exactly by

$$u = \sum_p Q_p u_p. \tag{3.1}$$

Here, $\{Q_p\}_{p=1}^{N_p}$ is the partition of unity defined by letting $Q_p$ be the identity on $\mathcal{D}_0$ and 0 otherwise. (This is the definition used throughout the paper.) It is generally absurd, however, for each processor to solve the original global equations, so each instead solves a much smaller problem, $A_c^p u_c^p = f_c^p$, that is fine only near the processor's domain and increasingly coarse as it extends away to the boundary. More precisely, processor $p$ iterates on residual equations by applying V-Cycles to

$$A_c^p u_c^p = r_c^p \equiv (P_c^p)^T (f_0 - A_0 u_0), \tag{3.2}$$

where $P_c^p$ is an interpolation operator from composite-grid to fine-grid functions. The global approximation is then corrected according to

$$u_0^{(n+1)} = u_0^{(n)} + \sum_p Q_p P_c^p u_c^p. \tag{3.3}$$

Notice that, in the ideal case, the composite grid is just the fine grid, so one iteration of AMG-DD is equivalent to traditional V-Cycles if the composite problems are solved by V-Cycles. The questions that remain are how the inaccuracies introduced by the composite grids are accumulated in forming the approximate solution of the original problem, and how much useful work can be done on each processor before a new residual must be calculated.

The dual of this process is AMG-RD, which decomposes the right-hand side (as opposed to the solution), and then reconstructs the approximation as the sum of the global processor components as opposed to local processor pieces. So, in AMG-RD, each processor solves a problem of the form

$$A_c^p u_c^p = (P_c^p)^T Q_p (f_0 - A_0 u_0)$$

and the approximate solution, $u$, is then formed according to

$$u_0^{(n+1)} = u_0^{(n)} + \sum_p P_c^p u_c^p.$$

Notice that if each processor is assigned the entire fine-grid problem, then, under the same assumptions as in the ideal AMG-DD setting, this could also correspond exactly to standard V-Cycles. However, we once again need to verify how well the composite problems combine to represent the original problem.

**3.2. Composite-Grid Creation.** A traditional V-Cycle requires each processor to obtain a solution for its own region, $\mathcal{D}_0^p$. This is accomplished by communicating on each grid of the V-Cycle with neighboring processors. The AMG-DD algorithm

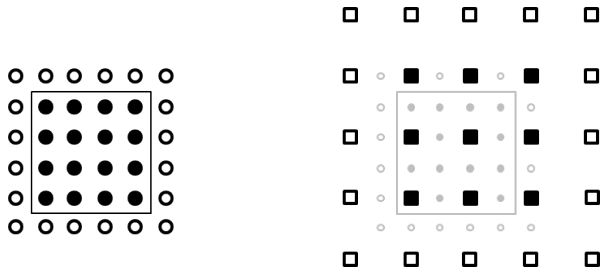

Fig. 3.1: Two-dimensional example of sets used to construct the composite grid for $\eta = 1$. Left: $\mathcal{D}_k^p$ (black circles), $\mathcal{D}_{k,\eta}^p$ (all circles). Right: $\mathcal{D}_{k+1}^p$ (black squares), $\mathcal{D}_{k+1,\eta}^p$ (all squares). The left sets are shown again on the right for reference.

instead uses global composite grids to approximate the solution on $\mathcal{D}_0^p$. A composite grid is best understood by examining how it is created.

Define distance, $dist(x, y)$, between two points $x, y \in \Omega_k$ in the usual way as the length of the shortest path connecting $x$ and $y$ in the graph of $A_k$. Similarly, define $dist(x, W)$ to be the minimal distance between point $x$ and the set of points $W$. The finest level of the composite grid, $\Omega_c^p$, associated with processor $p$ consists of the points that belong to $\mathcal{D}_0^p$ and all points that are within distance $\eta_0$ from $\mathcal{D}_0^p$:

$$\mathcal{D}_{0,\eta}^p = \{x \in \Omega_0 \,|\, dist(x, \mathcal{D}_0^p) \leq \eta_0\}\,.$$

Now, let $\mathcal{D}_1^p$ denote the points in $\mathcal{D}_{0,\eta}^p$ that are repeated on the first coarse grid, $\Omega_1$. Adding all the points within distance $\eta_1$ on grid $\Omega_1$, we obtain the set

$$\mathcal{D}_{1,\eta}^p = \{x \in \Omega_1 \,|\, dist(x, \mathcal{D}_1^p) \leq \eta_1\}\,.$$

Proceeding recursively in this fashion for all grids, $\Omega_0, \Omega_1, \ldots, \Omega_L$, we obtain the final composite grid

$$\Omega_c^p = \bigcup_{i=0}^{L} \mathcal{D}_{i,\eta}^p.$$

Figure 3.1 provides an illustration of the sets $\mathcal{D}_k^p$ and $\mathcal{D}_{k,\eta}^p$ in a simple 2D setting. Notice that, for any point in the composite grid, it is possible to trace a path from that point through the graph of the matrix to a point in $\mathcal{D}_0^p$. This is illustrated (in reverse order) by the path from $x_0 \in \mathcal{D}_0^p$ to $x_{k,\eta} \in \mathcal{D}_{k,\eta}^p \subseteq \Omega_c^p$ that arises naturally from the construction of the composite grid:

$$x_0 \to x_{0,\eta} \downarrow x_1 \to x_{1,\eta} \downarrow x_2 \to x_{2,\eta} \downarrow x_3 \to \ldots \downarrow x_k \to x_{k,\eta}. \tag{3.4}$$

Here, a right arrow signifies a path through the graph on the current grid, and a down arrow signifies a transition to a repeated point on a coarser grid.

Notice that this creation process constructs a composite grid, $\Omega_c^p$, consisting of the original region of the fine grid assigned to it and a mesh outside of this region that becomes increasingly coarse further away from the region, until it reaches the boundaries on some coarsest grid, $i \leq L$. Notice also that we can select $\eta_L$ to ensure that the composite grid reaches all boundaries on the coarsest grid, which is what we have done in all of the tests we report on below.

**3.3. Composite-Grid Operators.** Recall that we are forming the coarse grids from the original grid hierarchy that AMG created for us, so a natural fit is to use parts of the already created operators. In the previous section, we defined a composite grid, $\Omega_c$, and now we would like to form an operator from the associated function space, $G_c$, to the fine-grid function space, $G_0$, on the global fine grid.

For ease of discussion, first assume that there are only two grids, $\Omega_0$ and $\Omega_1$, in the original hierarchy, and assume we reorder the grids together with the corresponding composite grid $\Omega_{c1}^p$ as follows:

$$\begin{aligned}\Omega_0 &= \{\mathcal{D}_{0,\eta}^p \quad\quad , \ \Omega_0 - \mathcal{D}_{0,\eta}^p\},\\ \Omega_1 &= \{\mathcal{D}_{0,\eta}^p \cap \Omega_1, \ \Omega_1 - \mathcal{D}_1^p \ \},\\ \Omega_{c1}^p &= \{\mathcal{D}_{0,\eta}^p \quad\quad , \ \Omega_1 - \mathcal{D}_1^p \ \}.\end{aligned}$$

Notice that the individual sets that make up $\Omega_1$ are coarsened versions of the sets in $\Omega_0$, and the composite grid is a combination of sets from both. Rewriting the prolongation operator, $P_0$, based on this reordering gives

$$P_0 = \begin{bmatrix} * & * \\ P_{10} & P_{11} \end{bmatrix}.$$

From this, we define the composite prolongation operator, $P_{c1}$, that interpolates from grid $\Omega_{c1}^p$ to $\Omega_0$ by

$$P_{c1}^p = \begin{bmatrix} I & 0 \\ \hat{P}_{10} & P_{11} \end{bmatrix},$$

where $\hat{P}_{10}$ has had zero columns added to match the size of the identity block above it. Suppose now that there are three grids in the hierarchy, $\Omega_0, \Omega_1$, and $\Omega_2$, and that we reorder the grids $\Omega_1$ and $\Omega_2$, together with the corresponding composite grid, $\Omega_{c2}^p$, as follows:

$$\begin{aligned}\Omega_1 &= \{\mathcal{D}_{0,\eta}^p \cap \Omega_1, \ (\mathcal{D}_{1,\eta}^p - \mathcal{D}_1^p) \quad\quad , \ \Omega_1 - \mathcal{D}_{1,\eta}^p\},\\ \Omega_2 &= \{\mathcal{D}_{0,\eta}^p \cap \Omega_2, \ (\mathcal{D}_{1,\eta}^p - \mathcal{D}_1^p) \cap \Omega_2, \ \Omega_2 - \mathcal{D}_2^p \ \},\\ \Omega_{c2}^p &= \{\mathcal{D}_{0,\eta}^p \quad\quad , \ (\mathcal{D}_{1,\eta}^p - \mathcal{D}_1^p) \quad\quad , \ \Omega_2 - \mathcal{D}_2^p \ \}.\end{aligned}$$

Rewriting the prolongation operator, $P_1$, based on this reordering gives

$$P_1 = \begin{bmatrix} * & * & * \\ * & * & * \\ * & P_{21} & P_{22} \end{bmatrix}.$$

Then the composite operator takes the form

$$P_{c2}^p = \begin{bmatrix} I & 0 \\ \hat{P}_{10} & P_{11} \begin{bmatrix} I & 0 \\ \hat{P}_{21} & P_{22} \end{bmatrix} \end{bmatrix}.$$

For four grids, the operator takes the form

$$P_{c3}^p = \begin{bmatrix} I & 0 \\ \hat{P}_{10} & P_{11} \begin{bmatrix} I & 0 \\ \hat{P}_{21} & P_{22} \begin{bmatrix} I & 0 \\ \hat{P}_{32} & P_{33} \end{bmatrix} \end{bmatrix} \end{bmatrix}.$$

Continuing in this way to the coarsest grid that covers the whole domain, $\Omega_L$, we can define the prolongation operator from the composite function space to the global finest function space, and from this we define the composite operator $A_c^p = (P_c^p)^T A_0 P_c^p$ for each processor [11]. In practice, we can use an FAC type solver [13], which means that we would not need to explicitly form this operator, but our algorithm will behave as if we have. Notice that this composite operator is the natural extension of what would happen with a finite element discretization on the composite grid, and the grids can actually be chosen such that it exactly corresponds to the finite element discretization.

**3.4. AMG-DD.** We can now define AMG-DD in terms of the composite operators and grids. An AMG-DD cycle (Algorithm 3.1) starts with a given global initial guess, $u_0^{(0)}$, and right-hand side, $f_0$, partitioned among the processors. All processors then work in parallel to compute their pieces of the global residual and restrict them to their associated pieces of coarser grids. These residuals are then efficiently communicated at appropriate composite grids to other processors as outlined in the next section. Then each processor solves its composite-grid problem using a multigrid algorithm based on the original AMG setup, or some other geometric multigrid or AMG scheme based directly on composite-grid operator $A_c$. The parameter $\rho$ in Algorithm 3.1 specifies the number of local (communication-free) cycles that are done before the solution is patched together and the next iteration is performed.

**Algorithm 3.1** $u$ = AMG-DDCycle($\rho, u_0, f_0$) for solving $A_0 u_0 = f_0$

Form $r_0 = f_0 - A_0 u_0$.
Restrict the residual to $\Omega_1, \ldots, \Omega_L$.
Communicate the restricted residuals to form $r_c^p$ on each processor.
**Execute in parallel** on all processors $p = 1, \ldots, N_p$
  Set $u_c^p = 0$.
  Do $\rho$ cycles on $A_c^p u_c^p = r_c^p$.
  Update $u_0 \leftarrow u_0 + Q_p P_c^p u_c^p$.
**return** $u_0$.

**3.5. Communicating the Residual in AMG-DD.** While no inter-processor communication is required in the AMG-DD composite-grid solves, the residual within each subdomain must be transferred to all other processors, albeit generally at coarser scales. The global residual, $r = f - Au$, can be calculated in parallel using the original matrix and the pieces of the operator on each subdomain. Note that the residual for each composite problem can then be calculated in serial merely by forming the matrix-vector product $r_c^p = (P_c^p)^T r$. As such, the step in Algorithm (3.1) that calls for a residual communication is straightforward in serial. The implications of this requirement in a parallel implementation are addressed in Section 6.

**3.6. AMG-RD.** AMG-RD is based on the same composite grids and operators that are formed for AMG-DD. However, instead of patching together the solutions after an iteration, AMG-RD adds together the individual global approximations in each iteration. For the global summation of solutions to make sense, AMG-RD solves residual problems of the form $A_c^p u_c^p = (P_c^p)^T Q_p r$. This approach relies on the fact that the solutions are smooth outside the region, and can therefore be easily represented and approximated on the composite grid.

**Algorithm 3.2** $u = \text{AMG-RDCycle}(\rho, u_0, f_0)$ for solving $A_0 u_0 = f_0$

**Require:** $\rho$ : Number of on-processor cycles.
**Require:** $u_0$ : Initial guess.
**Require:** $f_0$ : Right-hand side.
Form $r_0 = f_0 - A_0 u_0$.
**Execute in parallel** on all processors $p = 1, \ldots, N_p$
Set $u_c^p = 0$.
Do $\rho$ cycles on $A_c^P u_c^p = (P_c^p)^T Q_p r_o$.
Update $u \leftarrow u_0 + \sum_p P_c^p u_c^p$.
**return** $u_0$.

**4. Theory.** This section establishes a two-level convergence theory for AMG-DD and AMG-RD. We first show that the two methods are duals of each other in the energy inner product. We then develop an abstract theory that establishes optimal convergence under full regularity and approximation property assumptions on the origin of the global fine-grid matrix equation, $Au = f$. In particular, we assume that $A$ is symmetric and positive definite, and that this matrix equation was created by conforming finite element discretization of a two- or three-dimensional uniformly elliptic operator on a region that is either a convex polygon or has smooth boundary. Below, we assume further that the finite elements are chosen on the fine, composite, and coarse levels in such a way that they satisfy the strong approximation property and that $\|A\|$ is bounded above in terms of the norm of the coarse-level matrix. These assumptions, while much more general, hold for continuous piecewise linear or bilinear elements on uniform grids that admit coarsening by a factor of 2.

In what follows, we use $C$ to denote a generic constant, independent of the number of processors and problem dimension, that may change meaning with each occurrence. Recall that $Q_p$ is the matrix that zeros out the entries in $f$ outside processor $p$'s subdomain, $D_0^p$, but leaves $f$ inside $D_0^p$ unchanged. As before, we let $\Omega_c^p$ and $G_c^p$ denote the respective composite grid and associated function space determined by the global grids, $\Omega_i, i = 0, 1, \ldots, L$. Also as before, we let $P_c^p : G_c^p \to G_0$ denote the interpolation operator from processor $p$'s composite grid to the fine grid, and write $A_c^p = (P_c^p)^T A P_c^p$. Then, one iteration of AMG-DD starting with a zero initial guess is

$$u \leftarrow \Sigma_p Q_p P_c^p (A_c^p)^{-1} (P_c^p)^T f.$$

The error propagation matrix for this method is

$$M_d = I - \Sigma_p Q_p P_c^p (A_c^p)^{-1} (P_c^p)^T A. \tag{4.1}$$

Similarly, one iteration of AMG-RD starting with a zero initial guess is

$$u \leftarrow \Sigma_p P_c^p (A_c^p)^{-1} (P_c^p)^T Q_p f,$$

with the error propagation matrix

$$M_r = I - \Sigma_p P_c^p (A_c^p)^{-1} (P_c^p)^T Q_p A. \tag{4.2}$$

Now, since the adjoint of any matrix $B$ in the energy inner product $< A\cdot, \cdot >$ is $A^{-1} B^T A$, then it is easy to see by the Euclidean symmetry of $P_c^p (A_c^p)^{-1} (P_c^p)^T$

and $Q_p$ that $M_d$ and $M_r$ are energy adjoints of each other. One take-home from this result is that the range and domain decomposition approaches exhibit the same energy convergence bounds. Another is that we could use one of these methods following the other to create a symmetric iteration.

While it may be more natural to measure convergence of either error propagation matrix in the energy norm, the Euclidean norm of $M_d$ may be simpler by virtue of the orthogonality of its individual terms. To see this for $M_d$ as defined in (4.1), let $S_p = P_c^p (A_c^p)^{-1} (P_c^p)^T A$, the energy-orthogonal projection of fine-grid vectors onto the range of $P_c^p$. Since the $Q_p$ sum to $I$, we have

$$\|M_d x\|^2 = \sum_p ||Q_p(I - S_p)x||^2.$$

Our aim now is to show that $||M_d||$ is bounded uniformly above by a constant less than 1.

The bottom line here is that we need to know how well composite-grid vectors approximate fine-grid vectors in the corresponding subdomain. It can, of course, provide no approximation at all to oscillatory vectors outside $D_0^p$ – vectors that are energy orthogonal to the composite grid. The key here is the presence of $Q_p$, which means that we just need to know the *local* accuracy in $D_0^p$, where there is no de-refinement. To avoid estimates of the above sum from depending on the number of processors, we need to localize the effect of $x$ on these terms. Our convergence theorem below shows that $I - S_p$ has enough structure to allow us to write $Q_p(I - S_p)x$ in terms of a component of $x$ that is nonzero only on $D_0^p$ and some close neighborhood about it.

To achieve this structure, we content ourselves with a two-grid result, but with minimal padding in the sense that $\eta_0 = 1$. We therefore now assume that each composite grid, $\Omega_c^p$, consists of $D_{0,1}^p$, the fine grid with uniform mesh spacing $h$ on $D_0^p$ and its nearest neighbors, and a $2h$ grid beyond that to the boundary, in what we call the de-refinement region, $\mathcal{D} = \Omega_1 - D_1^p$. Note that the composite grid has only coarse points in $\mathcal{D}$. We begin with two assumptions that hold for our problem setting. These assumptions are made with regard to a global coarsening of the fine grid by a factor of 2. In particular, let $P$ (without a subscript) denote the operator that interpolates from the *global* coarse grid to the fine grid, and let $P_1^p$ denote the operator that interpolates from the global coarse grid to $p$'s composite grid. Notice that $P_1^p : G_1 \to G_c^p$, and $A_1 = (P_1^p)^T A_c^p P_1^p$.

Assume now that the following *strong approximation property (SAP)*[14, 15] holds from the global coarse level to both the global fine and the composite levels, respectively: Letting $T = I - P\left(P^T A P\right)^{-1} P^T A$ and $T_p = I - P_1^p \left((P_1^p)^T A_c^p P_1^p\right)^{-1} (P_1^p)^T A_c^p$, then

$$||Tw|| \leq \frac{C}{||A||} ||Aw||, \qquad \forall w \in G_0, \tag{4.3}$$

and

$$||T_p w|| \leq \frac{C}{||A_c^p||} ||A_c^p w||, \qquad \forall w \in G_c^p. \tag{4.4}$$

Assume also that

$$\|A\| \leq C \|A_c^p\|. \tag{4.5}$$

Consider the doubly extended domain $\hat{\mathcal{D}}_0^p$ formed from $\mathcal{D}_0^p$ by extending it first to include the coarse-grid neighbors in the composite grid that are nearest to it and then to the next ring of composite-grid coarse-grid neighbors that are nearest the first extension. Our final assumption is that the $\hat{\mathcal{D}}_0^p$ overlap each other by a fixed amount. For example, for the two-dimensional discrete nine-point Laplacian and for moderate-size processor domains and small enough padding, no point exists in more than 9 doubly extended domains.

To establish convergence of AMG-DD, we incorporate a global relaxation step into the process:

$$u \leftarrow u - \frac{1}{\|A\|}(Au - f),$$

which has the error propagation matrix $I - \frac{1}{\|A\|}A$. The algorithm then consists of applying relaxation until it stalls and then using an AMG-DD correction step. We prove in the following theorem that uniform convergence is obtained either by relaxation alone or by the AMG-DD correction step alone. That is, the overall method converges independently of $h$ and $N_p$.

THEOREM 4.1. (Two-Grid Convergence) *AMG-DD converges uniformly in the Euclidean norm in the sense that there exists an $\epsilon < 1$, independent of $h$ and $N_p$, such that either*

$$||(I - \frac{1}{||A||}A)x|| \leq \epsilon||x||, \qquad \forall x \in G_0$$

*or*

$$||M_d x|| \leq \epsilon||x||, \qquad \forall x \in G_0.$$

*Proof.* If relaxation by itself satisfies the bound, then we are done. Otherwise, we may assume that relaxation has stagnated in the sense that the Euclidean norm of the error converges slowly:

$$||(I - \frac{1}{||A||}A)x|| \geq \epsilon||x||,$$

where $\epsilon \in (0, 1)$ is to be specified below. A little algebra then shows that $x$ has a relatively small Rayleigh quotient:

$$\frac{\langle Ax, x\rangle}{\langle x, x\rangle} \leq (1 - \epsilon^2)||A||. \tag{4.6}$$

But then the "oscillatory" component, $t = Tx = (I - P(P^TAP)^{-1}P^TA)x$, of $x$ that is energy orthogonal to $\mathcal{R}(P)$ must be relatively small in the Euclidean sense because the SAP in (4.3) implies that

$$||t||^2 \leq \frac{C}{||A||^2} < Ax, Ax > \leq \frac{C}{||A||} < Ax, x > \leq C(1 - \epsilon^2)||x||^2. \tag{4.7}$$

The strong sense of smoothness of the error expressed by (4.7) is the connection between the Euclidean and energy vector spaces that we need. We now hshow that

the composite-grid step effectively reduces such error. Specifically, our aim now is to prove that there exists a constant $c < \infty$ such that

$$\sum_p ||Q_p(I - S_p)t||^2 \leq c||t||^2, \qquad \forall t \in \mathcal{R}^{\perp_A}(P). \tag{4.8}$$

We could then combine (4.7) and (4.8) to establish the convergence bound

$$\|M_d x\|^2 = \sum_p ||Q_p(I - S_p)x||^2 = \sum_p ||Q_p(I - S_p)t||^2 \leq cC(1 - \epsilon^2)||x||^2.$$

The last equality here follows because $x$ differs from $t$ by $s \in \mathcal{R}(P) \subset \mathcal{R}(P_c^p)$, which is in the null space of $I - S_p$. We could then complete the proof by choosing

$$\epsilon = \sqrt{\frac{cC}{1 + cC}},$$

which, after a little algebra, reduces to the desired convergence bound.

We are now left with the sole task of establishing (4.8). To this end, note that a little more algebra yields

$$\sum_p ||Q_p(I - S_p)t||^2 \leq 2||t||^2 + 2\sum_p ||Q_p S_p t||^2 .$$

Focusing on the last term of this inequality, for a given $p$, define $r_c^p = (P_c^p)^T At$. It suffices now to establish that

$$\left|\left|Q_p P_c^p (A_c^p)^{-1} r_c^p\right|\right| \leq \frac{C}{||A_c^p||}||r_c^p||. \tag{4.9}$$

Since $||Q_p P_c^p|| = 1$, this would follow if we could prove that

$$||\tau_p|| \leq \frac{C}{||A_c^p||}\,||A_c^p \tau_p||\,, \tag{4.10}$$

where $\tau_p = (A_c^p)^{-1} r_c^p$. Since $T_p \tau_p = \tau_p$ and $P = P_c^p P_1^p$, then more algebra yet shows that $\tau_p \in \mathcal{R}(P_1^p)^{\perp_{A_c^p}}$, with superscript $\perp_{A_c^p}$ denoting $A_c^p$-orthogonal complement. But (4.10) follows directly from the SAP, (4.4), so we have thus established (4.9).

$P$ and $P_c^p$ agree outside $\mathcal{D}_0^p$, so $P^T At = 0$ implies that $r_c^p$ is nonzero only on $\mathcal{D}_0^p$ and its nearest coarse-grid neighbors. Thus, $||(P_c^p)^T At||$ only involves values of $t$ on $\hat{\mathcal{D}}_0^p$. Let $t_{loc}^p$ denote $t$ restricted to $\hat{\mathcal{D}}_0^p$. We then use (4.9) to conclude that

$$\left|\left|Q_p P_c^p (A_c^p)^{-1} (P_c^p)^T At\right|\right| \leq \frac{C}{||A_c^p||}||A||||t_{loc}^p||.$$

Finally, by the boundedness assumption, (4.5), on $\frac{||A||}{||A_c^p||}$ and the limited overlap assumption on $\hat{\mathcal{D}}_0^p$, we obtain

$$\sum_p ||Q_p P_c^p (A_c^p)^{-1} (P_c^p)^T At||^2 \leq \sum_p \frac{C}{\min_p ||A_c^p||^2}||A||^2||t_{loc}^p||^2 \leq C||t||^2.$$

We have therefore established (4.8) and, thus, the proof. □

REMARK 4.1. *Theorem 1 establishes uniform convergence of AMG-DD in the Euclidean norm. A direct consequence of this result is uniform convergence of its dual, AMG-RD, in the residual norm:*

$$||M_r||_{A^2} \equiv ||AM_r A^{-1}|| = ||M_d^T|| \leq \epsilon < 1.$$

**5. AMG-DD/RD Numerics.** While the previous section establishes theoretical convergence properties of the method in a two-level setting, more nuanced questions need to be addressed in practice. With the AMG-DD/RD algorithms, a very large parameter space must be explored if optimal choices are to be identified. The parameters selected can have effects on the maximum possible size of the problem due to the overlapping grids and on the time to solution as a function of computation/communications required. To obtain some insight into good practical ranges for these parameters, numerical tests were performed based on a simple model problem whose convergence properties are well understood in the context of AMG methods. The results of these experiments are reported here. These results were then used to determine reasonable choices for the parameters used to form the composite grids for the model problem. We establish choices for the padding on each level that seem reasonable in terms of their effect on convergence and complexity or, in other words, on time to solution. We then attempt to determine how much computational work should be expended on solving the composite grid subproblems before communication is allowed between processors.

The numerical results described below were obtained from a sequential implementation of the parallel AMG and AMG-DD algorithms. For AMG, we use $V(1,1)$ cycles (see Algorithm 2.1), hybrid Gauss-Seidel, and coarsening (i.e., inter-level transfer and coarse-level operators) based on the classical Ruge-Stüben algorithm [5]. These particular AMG parameter choices were made because they are fairly common, but other algorithmic choices would also lead to similar results and conclusions.

**5.1. Size of Problem on a Processor.** To model the computational cost of the AMG-DD algorithm, we have to first calculate the number of unknowns needed to form a composite problem. This is a function of the number of unknowns assigned to a processor, $|\mathcal{D}_0^p|$, and the padding on each grid, $\eta_i$. It is important to note that AMG-DD requires more memory than AMG to solve the same problem, so AMG can solve somewhat larger problems on any given machine, depending on the size of the padding.

As described earlier, the pad amount is the amount of nearest neighbors that we add to a problem on each grid. To develop a closed form for this, we start out with the assumption that the processor owns a region of the original problem of size $n^d$ in $d$ dimensions, and we assume a constant padding of $\eta$ and a constant coarsening factor $c \geq 2$. We can show recursively that the length of the side of $\mathcal{D}_{k,\eta}$, $k = 0, \ldots, L$, satisfies

$$l_k = \frac{n}{c^k} + \sum_{i=0}^{k} \frac{2\eta + \alpha_{k-i}}{c^i} \approx \frac{n}{c^k} + \sum_{i=0}^{k} \frac{2\eta}{c^i},$$

where each $-1 < \alpha_k < 1$ accounts for the fact that $l_{k-1}$ may not be perfectly divisible by $c$. The sizes of the $\mathcal{D}_{k,\eta}$ are then given by

$$|\mathcal{D}_{0,\eta}| = l_0^d, \quad |\mathcal{D}_{1,\eta}| = l_1^d, \quad \ldots \quad |\mathcal{D}_{L-1,\eta}| = l_{L-1}^d, \quad |\mathcal{D}_L| = |\Omega_L|.$$

To get the total size of the composite grid, it is easier to first imagine that there are only two grids, and then see that the size of the composite grid, $|\Omega_c|$, is the padded fine region combined with the coarse grid outside of that region:

$$|\Omega_c| = |\mathcal{D}_{0,\eta}| + |\mathcal{D}_{1,\eta} - \mathcal{D}_1| \approx |\mathcal{D}_{0,\eta}| + |\mathcal{D}_{1,\eta}| - \frac{|\mathcal{D}_{0,\eta}|}{c^d}.$$

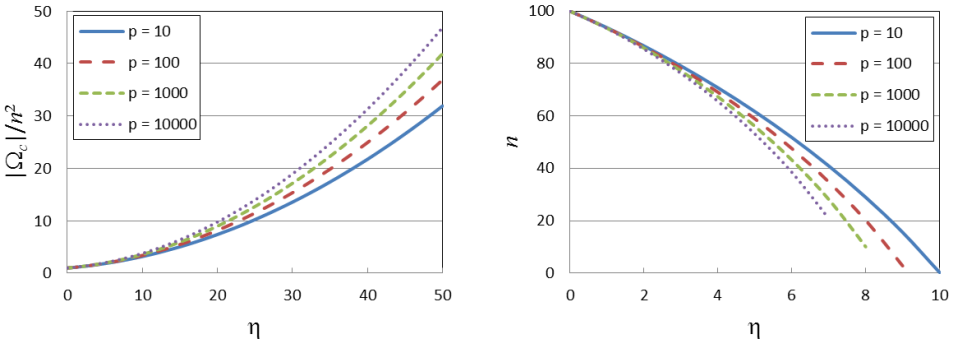


Fig. 5.1: Left: Growth of the composite grid for $n = 100$ as a function of $\eta$. Right: Effect of $\eta$ on local problem size $n$ under fixed memory constraints.

Now, using this same logic on all grids yields

$$\begin{aligned}
|\Omega_c| &= |\mathcal{D}_{0,\eta}| + |\mathcal{D}_{1,\eta} - \mathcal{D}_1| + \cdots + |\mathcal{D}_{L-1,\eta} - \mathcal{D}_{L-1}| + |\Omega_L - \mathcal{D}_L| \\
&\approx \left(1 - \frac{1}{c^d}\right) l_0^d + \left(1 - \frac{1}{c^d}\right) l_1^d + \ldots. + \left(1 - \frac{1}{c^d}\right) l_{L-1}^d + |\Omega_L| \\
&\approx \left(1 - \frac{1}{c^d}\right) \sum_{k=0}^{L-1} \left(\frac{n}{c^k} + \sum_{i=0}^{k} \frac{2\eta}{c^i}\right)^d + |\Omega_L| \\
&\approx \left(1 - \frac{1}{c^d}\right) \sum_{k=0}^{L-1} \left(\frac{n}{c^k} + 2\eta\left(\frac{c}{c-1}\right)\right)^d + |\Omega_L| \\
&= O\left(n^d + n^{d-1}\eta + \cdots + n\eta^{d-1} + L\eta^d + |\Omega_L|\right).
\end{aligned}$$

In particular, for the case of $d = 2$ dimensions and coarsening factor $c = 2$, we have

$$|\Omega_c| \approx n^2 + 12n\eta + 12L\eta^2 + |\Omega_L|.$$

Notice that the size of the composite grid depends on: $n^2$, the size of the fine-grid region assigned to a processor; $N_p$, the number of processors on the machine (assuming a fully coarsened hierarchy of grids so that $L \approx \log(N) \approx \log(N_p)$); and $\eta$, the size of the padding region. Considering this expression for $n = 100$ and several different fixed numbers of processors, Figure 5.1 shows how $|\Omega_c|$ scales with $\eta$ and also how the problem size $n$ is affected under fixed memory constraints.

Notice that Figure 5.1 shows that keeping $\eta$ small, especially on the order of one or two, means that the size of the composite grid is essentially the same as the size of the problem that would be assigned to a processor in a typical parallel implementation. Changing the scale of the processors has very little effect on the growth of the composite problem size, especially for small padding amounts.

**5.2. Parameter Effects on Convergence.** One issue that needs to be addressed is the ability of composite grids to approximate the solution on the fine-grid processor subdomains. Since AMG-DD is intended as a replacement for a V-cycle, its convergence factors are compared to that of a V-cycle for solving global problems of the same size. These results are presented in the context of weak scaling, that is, the

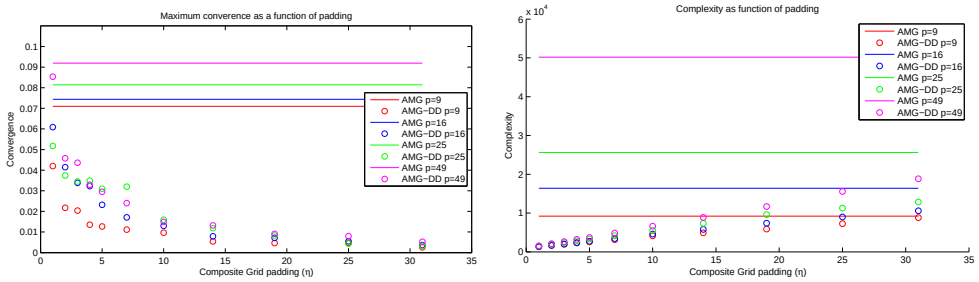


Fig. 5.2: Convergence and complexity (number of nonzeros in the composite-grid matrix) as functions of uniform padding for varying numbers of processors: 9=red, 16 = blue, 25 = green, and 49=pink. The AMG plots in the complexity graph show the number of unknowns in the global fine grid.

number of unknowns per processor is fixed and the number of processors is varied. (We ignore memory constraints in these results; see earlier discussion and Figure 5.1.)

Figure 5.2 compares optimal convergence of the AMG-DD cycle, with $\eta$ constant for each grid, to a standard AMG V-cycle. To achieve this optimal convergence, we solve each processor's problem as well as we can before doing any communication. This would of course only make sense if computation were actually "free". In Figure 5.2, we also show problem *complexity* (i.e., the number of nonzeros in the composite-grid matrix) on each processor as the padding increases for the case of uniform padding, which we present in relation to the number of unknowns in the global fine grid.

Figure 5.2 indicates that the composite problems can give good approximation to the global solution and, as expected, increasing the overlap between the processors enhances the convergence rate for the algorithm. However, convergence must be understood in the context of the attendant complexity. The complexity graph shows us that, for minimal padding, the size of the problem on each processor is almost independent of the size of the global problem, which is directly proportional to the number of processors. One question this leads to is how important it is to maintain a high level of padding outside of the fine-grid region.

Figure 5.3 is the same test as Figure 5.2, but now only $\eta_0$ is allowed to grow, while $\eta_k$ is fixed at 2 for all $k > 0$. Figure 5.3 shows that increasing the padding only on the fine grid greatly reduces its effectiveness; however growth of the problem in this setting is now directly tied just to the size of the fine grid patch on each processor, and is nearly independent of the number of processors and the global problem size.

In Table 5.1, we present sample convergence factors for parameters that are commensurate with the scaling models that we created for AMG-DD/AMG-RD. Each processor was assigned 5000 points in the original fine grid, and tests were run with various padding levels determined by $\eta = 1, 2, 3$, and 4 and various numbers of local solves per cycle determined by $\rho = 1, 2, 3$, and 4.

One important take-away from these results is that they appear to be almost independent of the number of processors that we selected. There is mild growth as the number of processors increases, but it is similar to growth exhibited by AMG. Convergence actually improves in some cases, but these fluxuations are quite small and probably due to minor variations in the convergence rate of the AMG method used to approximate the composite solves. Table 5.1 also indicates that, while it is

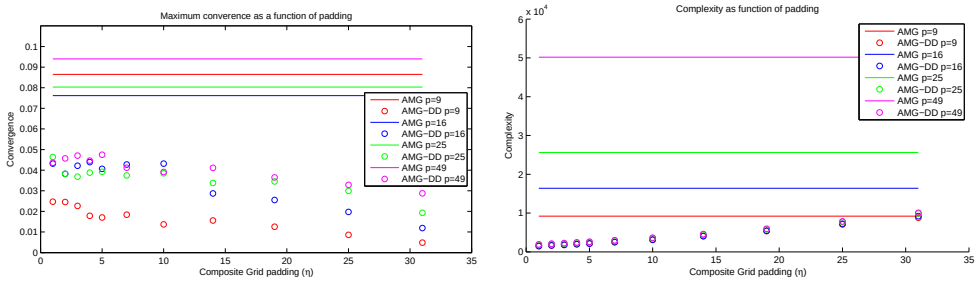


Fig. 5.3: Convergence and complexity (number of nonzeros in the composite-grid matrix) as functions of fine-grid padding (coarse-grid padding is fixed at 2) for varying numbers of processors: 9=red, 16 = blue, 25 = green, and 49=pink. The AMG plots in the complexity graph show the number of unknowns in the global fine grid.

NumProcessors = 9

| | $\rho=1$ | $\rho=2$ | $\rho=3$ | $\rho=4$ |
|---|---|---|---|---|
| $\eta=1$ | 0.119 | 0.053 | 0.043 | 0.054 |
| $\eta=2$ | 0.105 | 0.037 | 0.031 | 0.031 |
| $\eta=3$ | 0.121 | 0.036 | 0.027 | 0.026 |
| $\eta=4$ | 0.098 | 0.037 | 0.023 | 0.028 |

NumProcessors = 16

| | $\rho=1$ | $\rho=2$ | $\rho=3$ | $\rho=4$ |
|---|---|---|---|---|
| $\eta=1$ | 0.142 | 0.071 | 0.064 | 0.062 |
| $\eta=2$ | 0.139 | 0.058 | 0.050 | 0.041 |
| $\eta=3$ | 0.127 | 0.048 | 0.028 | 0.041 |
| $\eta=4$ | 0.123 | 0.043 | 0.027 | 0.030 |

NumProcessors = 25

| | $\rho=1$ | $\rho=2$ | $\rho=3$ | $\rho=4$ |
|---|---|---|---|---|
| $\eta=1$ | 0.147 | 0.073 | 0.061 | 0.062 |
| $\eta=2$ | 0.119 | 0.053 | 0.038 | 0.039 |
| $\eta=3$ | 0.122 | 0.049 | 0.035 | 0.035 |
| $\eta=4$ | 0.111 | 0.037 | 0.027 | 0.027 |

NumProcessors = 36

| | $\rho=1$ | $\rho=2$ | $\rho=3$ | $\rho=4$ |
|---|---|---|---|---|
| $\eta=1$ | 0.146 | 0.070 | 0.071 | 0.068 |
| $\eta=2$ | 0.156 | 0.061 | 0.058 | 0.045 |
| $\eta=3$ | 0.152 | 0.060 | 0.037 | 0.036 |
| $\eta=4$ | 0.128 | 0.044 | 0.031 | 0.039 |

Table 5.1: Convergence of AMG-DD for 9, 16, 25, and 36 processors with 5000 points per processor for varying padding ($\eta$) and number of cycles ($\rho$).

beneficial to perform two cycles on each processor before the residual is calculated, the benefit of any additional cycles is very mild. The table also suggests that any padding past $\eta = 2$ does not generate significant increase in convergence of the overall iteration. The results for AMG-RD are omitted since, as expected, they are nearly identical to those for AMG-DD.

Based on the results from the model problem, it is clear to see that two lines of grid-point padding on each level yields an effective process. Using such a small padding means that the problem stays within an acceptable region in terms of the amount of memory that must be wasted in order to support the composite grids. It is also clear that, within this framework for the model problem, only two iterations should be performed before communication is required. In all of the tests reported here, the local solves converged with a consistent factor of about 0.1, so $\rho$ really reflects the number of decimal places of accuracy in the solution of the composite-grid equations. This suggests that, for the model problem, two decimal places of accuracy

is an appropriate target for the local composite-grid solves.

**6. Parallel Issues.** As we said above, to simplify the task of studying AMG-RD and AMG-DD in a large parameter space, we restrict ourselves to weak scaling tests in anticipation of relatively weak processor architectures. One goal of this study is to envision modifications to AMG methods that may have better scaling characteristics for future generations of computers. As the number of processors increases, the cost of communication, especially on coarser levels, affects the weak-scaling characteristics of these algorithms. Our investigation of these methods is in the context of the effects of exascale computing on AMG algorithms, so strong-scaling is not studied here because the sheer size of the resulting problems would preclude such models.

We should also note that, in practice, not all communications are equal, and machines are tending to become hierarchical in nature. As the number of processors dramatically expand, we must reach a point where the subdomain contains a tiny fraction of the global problem, leaving the composite grid with extremely coarse grids on the global domain. We can thus expect convergence of AMG-RD and AMG-DD to stall at these extreme scales, unless we employ these methods in a hierarchical way on subsets of processors (e. g., "nodes") where communication may be relatively inexpensive.

In a naive implementation, if the padding is allowed to grow, then communication costs would have significant drawbacks in parallel. However, in the previous section, the parameters that we identified fall into a regime where we can accumulate the residual in a manner with little overhead. In fact, we introduce an algorithm here that allows communication within a $log(N_p)$ communication pattern, but with a reduced constant compared to a typical V-cycle.

**6.1. Setup.** Implementation of AMG-DD and AMG-RD can be done using only the constructs (coarse grids, coarse-grid operators, and intergrid transfer operators) of an existing AMG hierarchy. As in all AMG algorithms, the cost of this setup process for AMG-DD and AMG-RD becomes an issue. In a purely serial setting, setup costs for AMG-DD and AMG-RD can be up to about twice that of standard AMG due to the redundant calculation that we use. However, in parallel, the increase in setup costs over that of AMG itself can be reduced essentially to the cost of communicating the necessary components of the operators to each processor. This is a result of the fact that the construction of the setup is based on existing AMG constructs. This added cost is therefore bounded by that of performing one extra V-Cycle because the necessary communication process can be done using the same pattern as used in the residual communication process, described next.

**6.2. Residual Communication in Parallel.** Since each processor only needs the residual on its composite grid, communication can be minimized by starting the residual updates on the coarsest grid, $\Omega_L$, and bubbling the information up as detailed in Algorithm 6.1. The algorithm proceeds up through the hierarchy, exchanging data with neighbors on each grid. For example, consider the one-dimensional illustration in Figure 6.1. Every point in processor $p$'s composite grid can be labeled as being in at least one of three sets: $\mathcal{D}_0^p \cap \Omega_k$; a set $\Psi$ of distance-$\eta_k$ neighbors (indicated by squares in the figure); or the composite grid $\Psi_c$ formed from $\Psi \cap \Omega_{k+1}$, which is a coarsening of some set $\Psi$. The algorithm sends $\Psi$ and $\Psi_c$ (which may be empty) to neighboring processors. As presented, it communicates redundant information, but this is easy to modify in practice to reduce costs. We now show that the algorithm produces the desired residual update, and we show later in Section 7.2.1 that the total

**Algorithm 6.1** Collect residual for composite grids

**for** k $= L \to 0$ **do**
  Execute in parallel on all processors $(p = 1, 2, \ldots, N_p)$
  **if** $(\mathcal{D}_0^p \cap \Omega_k) \neq \emptyset$ **then**
    Identify the set of neighboring processors $\{p_1, \ldots, p_m\}$ (not equal to $p$) that contain points within distance $\eta_k$ of $(\mathcal{D}_0^p \cap \Omega_k)$.
    **for** $j = 1 \to m$ **do**
      Find all points $x \in (\mathcal{D}_0^p \cap \Omega_k)$ such that $dist(x, (\mathcal{D}_0^{p_j} \cap \Omega_k)) \leq \eta_k$, let $\Psi$ be the union of these points.
      Form$\Psi_c$ from $\Psi \cap \Omega_{k+1}$ as outlined in Section 3.2.
      Send the residual at all points and grids in $\Psi, \Psi_c$ to $p_j$.
    **end for**
  **end if**
**end for**

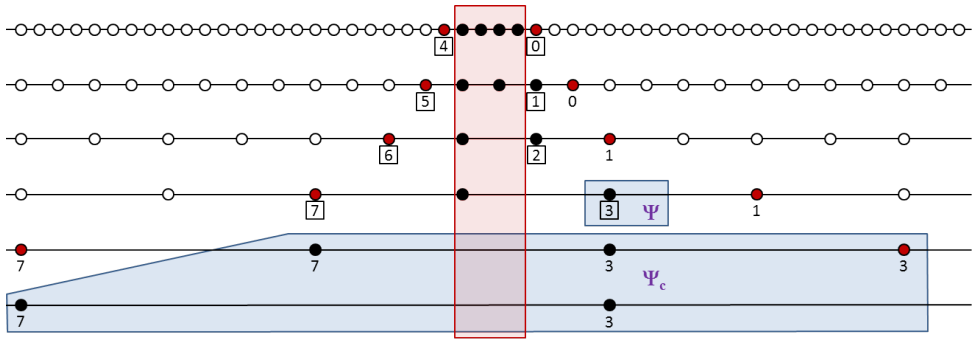


Fig. 6.1: One-dimensional example of a processor's composite grid and the residual update Algorithm 6.1. Processor $p$ owns the points in the red vertical rectangle. Points in the distance-$\eta_k$ neighborhood of $p$ are highlighted with squares and numbered 0 through 7. All other points are in the coarse composite grid of some distance-$\eta_k$ neighbor and are labeled accordingly. Example sets $\Psi$ and $\Psi_c$ are also given.

amount of data communicated by each processor is on the same order as the number of points on its boundary plus a log factor.

**Claim:** On grid $\Omega_k$, processor $p$ owns region $\mathcal{D}_0^p \cap \Omega_k$ . After the algorithm has executed for this grid, processor $p$ will have the correct residual for all points, denoted $\Omega_{c,k}^p$, on a composite grid built from $\mathcal{D}_0^p \cap \Omega_k$.

*Proof.* The needed information is assumed to be stored off processor; otherwise, the claim is trivially true.

- Suppose $k = L$ (the coarsest grid). Then, since $p$ communicates directly to all processors within $\eta_L$, it trivially has the correct residual for all $x \in \Omega_{c,L}^p$.
- Suppose $k = m$, and assume that each processor has obtained residual information at all points on $\Omega_{c,m+1}^p$, the composite grid formed from $\mathcal{D}_0^p \cap \Omega_{m+1}$. By construction, if $x \in \Omega_{c,m}^p$, then there exists $y \in \mathcal{D}_0^p \cap \Omega_m$ and a path of the form

$$y \to y_{m,\eta} \downarrow y_{(m+1)} \to y_{(m+1),\eta} \downarrow y_{(m+2)} \to \cdots \to x.$$

Notice, for some processor $q$, that $y_{(m+1)} \in \mathcal{D}_0^q \cap \Omega_{m+1}$ and $x \in \Omega_{c,m+1}^q$. Hence, processor $q$ has residual information for point $x$ by assumption and, since $y_{(m+1)}$ is within distance $\eta_m$, then processor $q$ passes that information to processor $p$.

The claim thus holds. □

This claim shows that, after the $k^{th}$ grid has been updated, each processor has the composite grid defined by the region it owns for that grid. Therefore, after all grids have been updated, each processor obtains the residual information it needs for its composite grid. The cost of this algorithm is detailed in Section 7.2.

**7. Parallel Numerics.** Ideally, to judge the scaling characteristics of this algorithm, we would present results based on tests run on different architectures. However, no fully parallel implementation of this algorithm has yet been developed, so we instead rely on models that give us indications of how we expect AMG and AMG-DD/AMG-RD to scale. Much of the analysis of the scalability of a parallel implementation of AMG is based on the work by Gahvari et al. [16]. Throughout this section, three main parameters are utilized:

1. $\alpha$: The cost of latency on the machine per message. For now, we assume that this is a constant that does not depend on the distance of connections.
2. $\beta$: Inverse bandwidth cost or, more generically, the cost per amount of data sent.
3. $\gamma$: The flop rate of the machine, that is, the amount of work per computation.

Based on data in Table 2 from [16], we make the assumption that $\alpha, \beta, \gamma$ have the following relations for the models:

$$\alpha = 10^4\gamma \quad \beta = 10\gamma \ .$$

We also assume, as before, that we are solving a nine-point discretization of a Laplacian in two dimensions, which allows us to predict stencil patterns in a full coarsening scenario. Also note that the models just describe the cost of V-cycles; setup modeling is omitted at this time.

**7.1. AMG V-Cycle Cost.** One standard process for modeling a parallel algorithm is to analyze the cost on the most expensive processor, so we follow this structure. In its most basic form, the model of one AMG V-Cycle can be written as

$$T_{amg} = T_{latency} + T_{comm} + T_{comp}.$$

The communication terms can be modeled as

$$T_{latency} + T_{comm} = \sum_{i=0}^{L} \left(\alpha m_i + \beta q_i\right),$$

where $L+1$ is the number of grids in the system (note that $L \approx log(N)$, with $N$ the number of unknowns in the system), $m_i$ is the number of messages sent on level $i$, and $q_i$ is the amount of data sent on level $i$. To calculate these numbers, we model the $V(1,1)$-Cycle (Algorithm 2.1). To actually specify the parameters $m_i, q_i$, we need to understand the structure of the operators in the multigrid hierarchy. For this analysis, we assume standard geometric full coarsening by a factor of 2, so complexity and operators are identical on all grids.

**7.1.1. Communication costs per AMG V-Cycle.** For calculating the parameters, let $\Gamma_i$ denote the number of unknowns on the boundary of the processor of interest on grid $i$. Since we are modeling the amount of information required to advance computation, then, to be precise, if the number of points on a side for a processor is $r$, the amount of information that needs to be gathered is $4r + 4$. However, the effect of the four corner points is negligible, so we assume that, given $n_i$ points on a side on grid $i$, we have

$$\Gamma_i = 4n_i.$$

This also leads to the convenient relation $\Gamma_i = \frac{\Gamma_0}{2^i}$. Parameters $m_i, q_i$ can now be split into their components for relaxation, residual calculation, restriction, interpolation, and coarse-grid solve.

First, we assemble the latencies:

$$m_i = \begin{cases} 2m_{i,relax} + m_{i,residual} + m_{i,restrict} + m_{i,interp} & i < L \\ 2m_{L,solve} & i = L. \end{cases}$$

For the case we are investigating, we have $m_i = 2 \cdot 8 + 8 + 8 + 8$ if $i < L$ and $2 \cdot 8$ if $i = L$. The latency cost can then be represented as

$$T_{latency} = \alpha \left( \sum_{i=0}^{L-1} 40 + 2 \cdot 8 \right) = \alpha(40L + 16).$$

The amount of data sent is

$$q_i = \begin{cases} 2q_{i,relax} + q_{i,residual} + q_{i,restrict} + q_{i,interp} & i < L \\ 2 \cdot q_{L,solve} & i = L, \end{cases}$$

which, for this case, is

$$q_i = \begin{cases} 2\Gamma_i + \Gamma_i + \frac{\Gamma_i}{2} + \frac{\Gamma_i}{2} & i < L \\ 2\Gamma_L & i = L. \end{cases}$$

The inverse bandwidth cost can be modeled as

$$T_{comm} = \beta \left( \sum_{i=0}^{L-1} 4\Gamma_i + 2\Gamma_L \right) = \beta \left( \sum_{i=0}^{L-1} 4\frac{\Gamma_0}{2^i} + 2\frac{\Gamma_0}{2^L} \right) \simeq 8\beta\Gamma_0.$$

**7.1.2. Computation Costs per AMG V-Cycle.** Assuming that there are $W_0 = n^2$ points on the finest grid, then the number of points, $W_i$, on the $i$-th grid is

$$W_i = \frac{W_0}{2^{2i}}.$$

Computation cost is then based on the amount of work done on each grid. So, for each grid, we can model the work in terms of the number of unknowns on individual grids:

- Relaxation (Gauss-Seidel) : 18 operations
- Restriction (Ideal) : $\frac{18}{4}$ operations

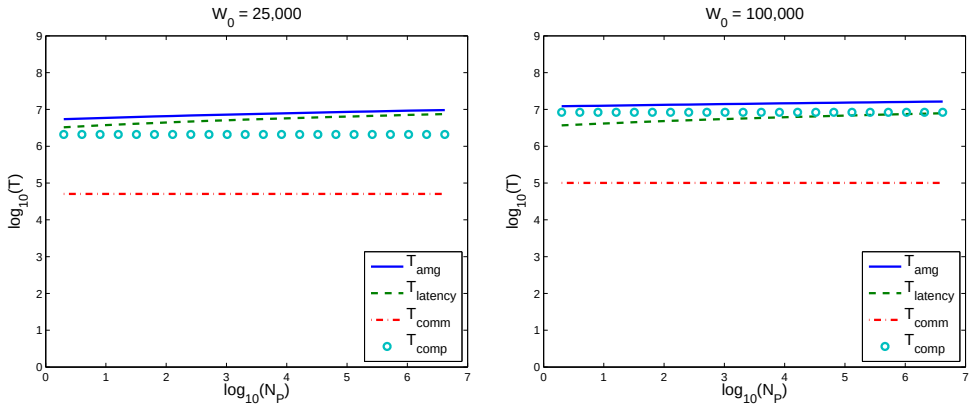


Fig. 7.1: AMG V-Cycle costs for $W_0 = 25,000$ and $100,000$, respectively.

- Interpolation (Ideal) : $\frac{18}{4}$ operations
- Residual calculation: 18 operations
- Coarse-grid solve : 2 relaxations = 36 operations.

This translates to per-grid costs of

$$\begin{aligned} \mathcal{N}^i_{relax} &= 18W_i, \quad \mathcal{N}^i_{restrict} = \tfrac{18}{4}W_i, \quad \mathcal{N}^i_{interp} = \tfrac{18}{4}W_i, \\ \mathcal{N}^i_{residual} &= 18W_i, \quad \mathcal{N}^i_{solve} = 36W_L. \end{aligned}$$

So the total computational cost is

$$\begin{aligned} T_{comp} &= \gamma \left( \sum_{i=0}^{L-1} 2\mathcal{N}^i_{relax} + \sum_{i=0}^{L-1} \mathcal{N}^i_{residual} + \sum_{i=0}^{L-1} \mathcal{N}^i_{restrict} + \sum_{i=0}^{L-1} \mathcal{N}^i_{interp} + \mathcal{N}_{solve} \right) \\ &= \gamma W_0 \left[ 63 \left( \frac{1-(1/4)^L}{1-(1/4)} \right) + \frac{36}{2^{2L}} \right] \simeq 84\gamma W_0. \end{aligned}$$

**7.1.3. Graphs for AMG Costs.** For the purpose of modeling, we scale by setting $\gamma = 1$. For a problem with $N_p$ processors and $n^2$ unknowns per processor, then $L = \lceil log_4(N_p \cdot n^2) \rceil$, $W_0 = n^2$, and $\Gamma_0 = 4n$. Using the derived functions for the AMG costs, the models for a $V(1,1)$-Cycle for $N_p = 2^i$, $i = 1, \ldots, 20$, while holding the number of unknowns per processor constant yields Figure 7.1. Notice that this figure shows that the only term that grows significantly as the number of processors increases is latency.

**7.2. AMG-DD Cycle Cost.** As in the derivation of costs for AMG, we again break the problem down into three parts:

$$T_{amgdd} = T_{latency} + T_{comm} + T_{comp}.$$

First, we have to be specific about the form of the algorithm that we are modeling. The assumption here is that there already has been an AMG setup, and the parts of the matrix needed to solve the composite problems have already been distributed. Therefore, we restrict ourselves to modeling the solve cycles of AMG-DD from Algorithm 3.1.

**7.2.1. Communication Costs per AMG-DD Cycle.** There are three main points of communication with the AMG-DD cycle: residual calculation; residual restriction; and communication of the composite residuals. Residual calculation and restriction, respectively, cost

$$\begin{aligned} m_{residual} &= 8, & q_{residual} &= \Gamma_0 \\ m_{restriction} &= 8L, & q_{restriction} &= \frac{\Gamma_0}{2}\left(\frac{1-(1/2)^L}{1-1/2}\right). \end{aligned}$$

For the communication of the residual, we model the algorithm developed for AMG-DD (Algorithm 6.1). To simplify the model so that the number of neighboring processors is fixed on all grid levels, we assume that $\eta$ is appropriately reduced to a smaller value on coarse grids. This is likely the best way to design the algorithm for parallel computation anyway. We do not, however, account for this reduction in the communication term, hence the model considered here is slightly more pessimistic than it should be for this latency-friendly case. Since nearest-neighbor communication is all that is required on each grid, then the latency cost is just

$$m_{res.comm} = \sum_{i=0}^{L} 8 = 8(L+1).$$

To bound the amount of data sent in Algorithm 6.1, we can use the result derived in Section 5.1 for the size of the composite grid. Given a fixed side length, $n$, on a processor and a fixed padding, $\eta$, on each grid, then the amount of data communicated is proportional to

$$|\Omega_c| - n^2 \approx n^2 + 12n\eta + 12L\eta^2 + |\Omega_L| - n^2.$$

Hence, if $|\Omega_L|$ is small, we can take

$$q_{res.comm} = 12n\eta + 12L\eta^2.$$

Notice that the total amount of data that must be communicated in the residual communication routine is on the order of the size of the boundary on a processor plus a log factor.

The communication costs are then

$$T_{latency} = \alpha(8 + 8L + 8(L+1)) = 16\alpha(L+1)$$

and

$$\begin{aligned} T_{comm} &= \beta\left(\Gamma_0 + \frac{\Gamma_0}{2}\left(\frac{1-(1/2)^L}{1-1/2}\right) + 12n\eta + 12L\eta^2\right) \\ &\simeq \beta\left(2\Gamma_0 + 12n\eta + 12L\eta^2\right). \end{aligned}$$

**7.2.2. Computation Costs per AMG-DD Cycle.** Using the work from the AMG derivations and the assumption that $W_0$ is the size of the fine grid on a processor, we know that the residual computation costs

$$\mathcal{N}_{residual.comp} = 19W_0.$$

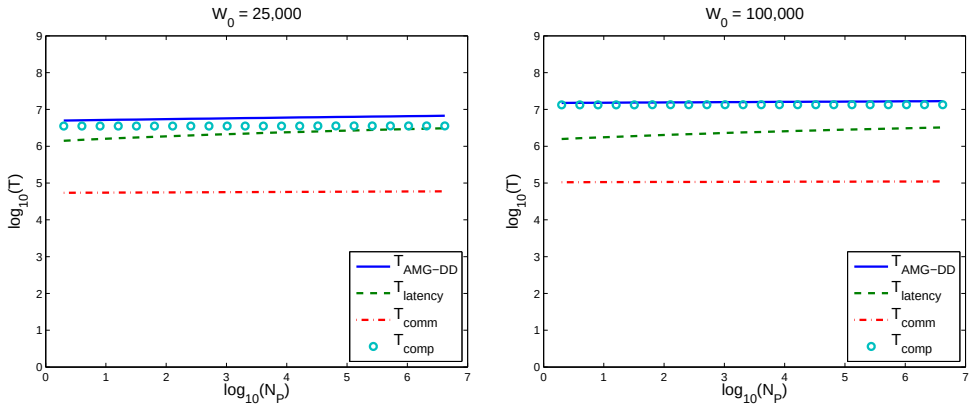


Fig. 7.2: AMG-DD costs for $W_0 = 25,000$ and $100,000$, with $\eta = 2$ and $\rho = 1$.

Restriction requires 18 operations, and must be done for each grid, so

$$\mathcal{N}_{restrict.comp} = \sum_{i=1}^{L} 18\frac{W_0}{2^{2i}} = 18W_0\left(\frac{1-(1/4)^L}{1-(1/4)}\right) \simeq 36W_0.$$

Finally, solving the composite problems can be estimated in the same fashion as for an AMG V-cycle, except that we now replace $W_0$ with the calculated size of the composite grid, $|\Omega_c|$, and then add the parameter $\rho$ that represents the number of cycles done between calculations:

$$\mathcal{N}_{amgdd.comp} = \rho|\Omega_c|\left[63\left(\frac{1-(1/4)^L}{1-(1/4)}\right) + \frac{36}{2^{2L}}\right],$$

where $\rho$ is on the order of two to three. This yields a total computation cost of

$$\begin{aligned} T_{comp} &= \gamma\left(19W_0 + 18W_0\left(\frac{1-(1/4)^L}{1-(1/4)}\right) + \rho|\Omega_c|\left[63\left(\frac{1-(1/4)^L}{1-(1/4)}\right) + \frac{36}{2^{2L}}\right]\right) \\ &\simeq \gamma\left(43W_0 + \rho 84|\Omega_c|\right). \end{aligned}$$

**7.2.3. Graphs for AMG-DD.** Using these derivations, we present scaling models in Figures 7.2 and 7.3 for AMG-DD cycles for a select sample of parameters. First, we fix the padding at $\eta = 2$ and the number of points per processor at $W_0 = 25,000$ and $100,000$, and then present the models for $\rho = 1$. Second, we model the scaling characteristics for larger padding, $\eta = 20$, fixing $\rho = 2$, for processor sizes $W_0 = 25,000$ and $100,000$. The effect of doing more, or less, computation (changing $\rho$) merely shifts the scale of computation up. For any $\rho > 2$, cost savings from reducing communication are lost by the increased time in computation.

Figures 7.2 and 7.3 show that, as desired, the AMG-DD algorithm achieves the goal of reducing communication for computation. While latency and communication both increase as the number of processors increases, the computation cost is now what drives the overall time per iteration of the algorithm. The amount of computation that has been added to each processor, as shown in Section 7.2.2, is on the same order of magnitude as the standard approach, so these figures also serve to highlight the balance between computation and communication that already exists in traditional AMG implementations.

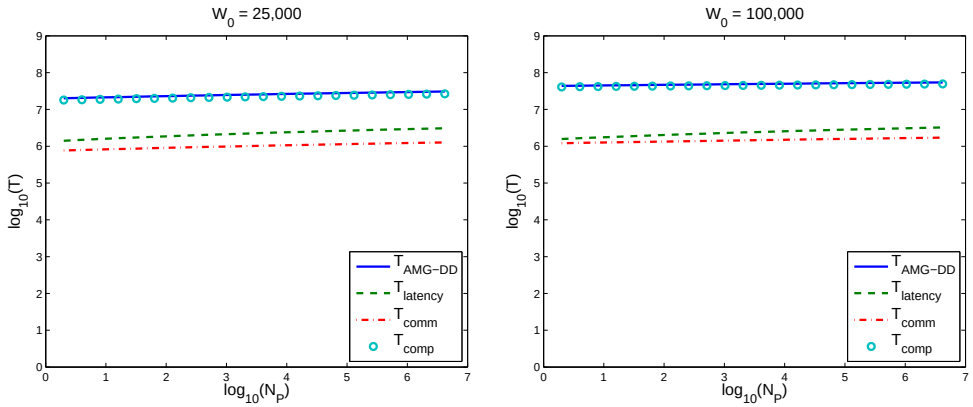


Fig. 7.3: AMG-DD costs for $W_0 = 25,000$ and $100,000$, with $\eta = 20$ and $\rho = 2$

.

**8. Conclusion.** Two new *algebraic* solvers, AMG-RD and AMG-DD, were developed and analyzed here that aim to trade communication for computation by forming global composite "grids" based only on the matrix, not the geometry. This trade-off is achieved efficiently by way of global composite grids on each processor that enable full subdomain overlap, but primarily at very coarse scales of resolution. One advantage of this approach is that it allows for the construction of composite problems with no prior knowledge of the original grid structure, but forms them algebraically, based only on components of an existing AMG setup. An important development of this methodology is a novel residual communication process that enables effective parallel computation on composite grids, avoiding the all-to-all communication costs of the geometric methods. The main purpose of this paper was to study the potential of these two algebraic methods as possible alternatives to existing AMG approaches for future parallel machines. To this end, we developed some theoretical properties of these methods and reported on serial numerical tests of their convergence properties over a spectrum of problem parameters. We also included a parameter study based on a performance model designed to anticipate their potential for use in emerging parallel architectures.

Our cost models show that these methods compete *as algebraic solvers* with current AMG algorithms on modern large-scale computers, but may surpass them in a significant way only if and when future architectures come with increased communication-to-computation cost ratios. This is, however, less of a limitation of the AMG-DD and AMG-RD algorithms than a testament to the scaling efficiency of current parallel AMG methods, and future research on improvements in the use of composite grids may tip the balance in favor of these new approaches.

An important aspect of AMG-DD and AMG-RD that we have not yet explored is the potential for use in a *nested iteration* (or *full multigrid*) process: While these methods seem comparable to AMG solvers in terms of *algebraic* convergence factors, they may prove to be very effective when used to reduce the error to discretization-error levels. Specifically, if we apply a nested iteration approach to the composite grid subproblems (which requires no communication between processors), it may well be that the resulting error will be acceptable in the sense that it is comparable to the error in the discrete solution itself–relative to the solution of the underlying PDE (assuming

here that there is one). These methods may thus be able to deliver acceptable results with only one communication phase. This capability could substantially reduce the communication costs that currently inhibit the use of full multigrid algorithms in large-scale parallel applications. Future work will therefore focus on the study of these methods in a nested iteration context.

## REFERENCES


[1] Brandt A, Diskin B. Multigrid solvers on decomposed domains. *Domain Decomposition Methods in Science and Engineering: The Sixth International Conference on Domain Decomposition*, *Contemporary Mathematics*, vol. 157, American Mathematical Society: Providence, Rhode Island, 1994; 135–155.

[2] Mitchell W. A parallel multigrid method using the full domain partition. *Electron. Trans. Numer. Anal.* 1998; **6**:224–233.

[3] Bank RE, Holst MJ. A new paradigm for parallel adaptive meshing algorithms. *SIAM J. Sci. Stat. Comp.* 2000; **22**:1411–1443.

[4] Brandt A, McCormick S, Ruge J. Algebraic multigrid (AMG) for sparse matrix equations. *In Sparsity and its Applications, D.J, Evans (ed.).*, 1984; 257–284.

[5] Ruge J, Stüben K. *Algebraic multigrid (AMG). In Multigrid Methods, vol. 5, McCormick SF (ed.)*. SIAM: Philadelphia, PA., 1986.

[6] Mitchell W. Parallel adaptive multilevel methods with full domain partitions. *App. Num. Anal. and Comp. Math.* 2004; **1**:36–48.

[7] Bank RE, Lu S. A domain decomposition solver for a parallel adaptive meshing paradigm. *SIAM J. Sci. Comput.* 2004; **26**(1):105–127.

[8] Bank RE, Vassilevski PS. Convergence analysis of a domain decomposition paradigm. *Comput. Visual Sci.* 2008; **11**:333–350.

[9] Bank RE. Some variants of the Bank and Holst parallel adaptive meshing paradigm. *Comput. Vis. Sci.* Oct 2006; **9**(3):133–144.

[10] Bank RE, Jimack P. A new parallel domain decomposition method for the adaptive finite element solution of elliptic partial differential equations. *Concurrency Computat.: Pract. Exper.* 2001; **13**:327–350.

[11] Bank RE, Lu S, Tong C, Vassilevski PS. Scalable parallel algebraic multigrid solvers. *Technical Report UCRL-TR-210788*, Lawrence Livermore National Laboratory, Livermore, California 2004.

[12] Henson VE, Yang UM. Boomeramg : A parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics* April2002; **41:155177**.

[13] McCormick S. *Multilevel Adaptive Methods for Partial Differential Equations*. No. 6 in Frontiers in Applied Mathematics, SIAM, 1989.

[14] Hackbusch W. Survey of convergence proofs for multi-grid iterations. *Special topics of applied mathematics : functional analysis, numerical analysis, and optimization : proceedings of the seminar held at the GMD, Bonn, 8-10 October 1979*, Frehse J, Pallaschke D, Trottenberg U (eds.). North-Holland Pub.: Amsterdam, 1980; 151–164.

[15] Vassilevski PS. Coarse spaces by algebraic multigrid: multigrid convergence and upscaling error estimates. *Advances in Adaptive Data Analysis* 2011; **3**(01n02):229–249.

[16] Gahvari H, Gropp W, Jordan KE, Schulz M, Yang UM. Modeling the performance of an algebraic multigrid cycle using hybrid mpi/openmp. *2012 41st International Conference on Parallel Processing* 2012; **0**:128–137.