

# Enabling Computational Technologies for Terascale Scientific Simulations

*S.F. Ashby*

**August 24, 2000**

*U.S. Department of Energy*



Lawrence  
Livermore  
National  
Laboratory

## DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

This work was performed under the auspices of the U. S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

This report has been reproduced directly from the best available copy.

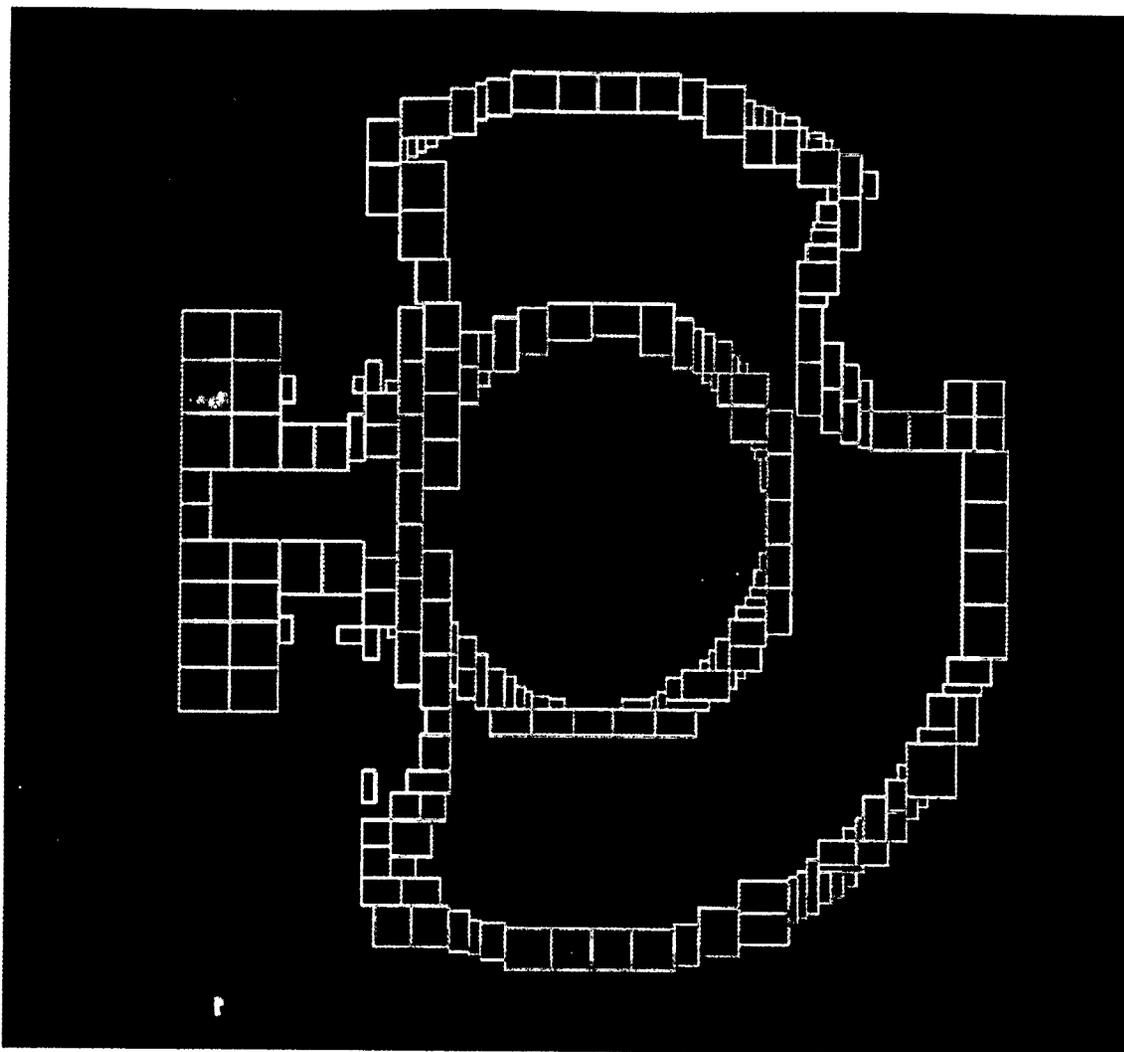
Available electronically at <http://www.doe.gov/bridge>

Available for a processing fee to U.S. Department of Energy  
and its contractors in paper from  
U.S. Department of Energy  
Office of Scientific and Technical Information  
P.O. Box 62  
Oak Ridge, TN 37831-0062  
Telephone: (865) 576-8401  
Facsimile: (865) 576-5728  
E-mail: [reports@adonis.osti.gov](mailto:reports@adonis.osti.gov)

Available for the sale to the public from  
U.S. Department of Commerce  
National Technical Information Service  
5285 Port Royal Road  
Springfield, VA 22161  
Telephone: (800) 553-6847  
Facsimile: (703) 605-6900  
E-mail: [orders@ntis.fedworld.gov](mailto:orders@ntis.fedworld.gov)  
Online ordering: <http://www.ntis.gov/ordering.htm>

OR

Lawrence Livermore National Laboratory  
Technical Information Department's Digital Library  
<http://www.llnl.gov/tid/Library.html>



Adaptive mesh refinement dynamically focuses computational effort in the areas of interest, such as near the shock fronts in this hydrodynamics simulation.

## **Enabling Computational Technologies for Terascale Scientific Simulations**

*A comprehensive final report for LDRD 97-ERD-114*

**Steven F. Ashby**

## Executive Summary of the Comprehensive Report

We develop scalable algorithms and object-oriented code frameworks for terascale scientific simulations on massively parallel processors (MPPs). Our research in multigrid-based linear solvers and adaptive mesh refinement enables Laboratory programs to use MPPs to explore important physical phenomena. For example, our research aids stockpile stewardship by making practical detailed 3D simulations of radiation transport.

The need to solve large linear systems arises in many applications, including radiation transport, structural dynamics, combustion, and flow in porous media. These systems result from discretizations of partial differential equations on computational meshes. Our first research objective is to develop multigrid preconditioned iterative methods for such problems and to demonstrate their scalability on MPPs.

Scalability describes how total computational work grows with problem size; it measures how effectively additional resources can help solve increasingly larger problems. Many factors contribute to scalability: computer architecture, parallel implementation, and choice of algorithm. Scalable algorithms have been shown to decrease simulation times by several orders of magnitude.

Multigrid is an example of a scalable linear solver. It uses a relaxation method such as Gauss-Seidel to damp high-frequency error, leaving only low-frequency (smooth) error—which can be efficiently solved for on a coarser (smaller) grid. Recursively applying this to each subsequent coarse-grid system creates a multigrid V-cycle, so named because one first descends a hierarchy of successively coarser grids, solves a small problem, and then ascends the hierarchy of grids. Interpolation and prolongation are used to traverse the hierarchy. If these operations are defined properly, the algorithm's computational costs grow linearly with problem size.

We explore geometric and algebraic multigrid techniques. Geometric multigrid is typically used in problems on structured meshes. Such an algorithm, based on Shaffer's semi-coarsening method, sped up the linear algebra in an ASCI code by a factor of 27, reducing overall simulation time 10-fold for a 2D test problem (128,000 unknowns). Algorithmic scalability was shown in 3D test problems on the ASCI Blue Pacific and Red MPPs. In particular, using Red we solved a problem with 134 million unknowns in 24 seconds on 2048 processors. We investigate algebraic multigrid methods for problems defined on unstructured meshes. We have parallelized Ruge's method and will run scalability experiments in FY99.

Our second research objective is to develop an object-oriented code framework for structured adaptive mesh refinement (AMR) applications. AMR allows efficient use of computing resources (CPU time and memory) by focusing numerical effort locally within the computational domain with varying degrees of spatial and temporal resolution. This makes practical simulations—especially those involving complex physics and large

spatial domains—that would be too expensive on a uniform mesh. Our framework, SAMRAI, provides extensible software support for rapid development of parallel AMR applications.

We completed the basic framework in FY98, including grid hierarchy management, adaptivity control, and visualization support. This parallel 3D framework is being used to develop several simulation codes, most notably one for studying laser–plasma interactions. SAMRAI is also used by the Utah ASCI Alliance Center of Excellence for its fire safety simulation code. In FY99, we will implement numerical methods and will work with application teams to run large-scale simulations on ASCI platforms.

**Publications** (A number of publications are attached in reverse chronological order.)

Ashby, S.F., et al., “A Numerical Simulation of Groundwater Flow and Contaminant Transport on the Cray T3D and C90 Supercomputers,” *Int'l J. Supercomp. Apps. and H-P Comp.*, in press February 1999. Also available as Lawrence Livermore National Laboratory technical report UCRL-JC-118635.

Brezina, M., A.J. Cleary, R.D. Falgout, V.E. Henson, J.E. Jones, T.A. Manteuffel, S.F. McCormick, and J.W. Ruge, “Algebraic Multigrid Based on Element Interpolation (AMGe),” *SIAM J. Sci. Comp.*, in press October 1998. Also available as Lawrence Livermore National Laboratory technical report UCRL-JC-131752.

Brooks, E., and K.A. Warren, “A Study of Performance on SMP and Distributed Memory Architectures Using a Shared Memory Programming Model,” *Supercomputing 97*. Also available as Lawrence Livermore National Laboratory technical report UCRL-JC-127449.

Cleary, A.J., R.D. Falgout, V.E. Henson, J.E. Jones, T.A. Manteuffel, S.F. McCormick, G.N. Miranda, and J.W. Ruge, “Robustness and Scalability of Algebraic Multigrid,” *SIAM J. Sci. Comp.* special issue on *5<sup>th</sup> Copper Mountain Conf. on Iterative Methods*, **21** (5/2000), pp. 1886–1908. Also available as Lawrence Livermore National Laboratory technical report UCRL-JC-130718.

Cleary, A.J., R.D. Falgout, V.E. Henson, J.E. Jones, “Coarse-Grid Selection for Parallel Algebraic Multigrid,” *Proc. 5<sup>th</sup> Intl Sym. on Solving Irregularly Structured Problems in Parallel*, **1457**(1998), *Lecture Notes in Computer Science*, (Springer-Verlag: New York) pp. 104–115. Lawrence Berkeley National Laboratory, Berkeley, CA, August 9–11, 1998. Also available as Lawrence Livermore National Laboratory technical report UCRL-JC-130893.

Falgout, R., “Scalable Linear Solvers,” *Supercomputing 97*. Also available as Lawrence Livermore National Laboratory technical report UCRL-TB-128636. 97-ERD-114.

Falgout, R.D., *AMGe: A Theory for a New Generation AMG for Finite Element Discretizations*, Lawrence Livermore National Laboratory technical report UCRL-MI-

130240 and LDRD 97-ERD-114.

Henson, Van Emden, and Panayot Vassilevski, Element-free AMGe: General Algorithms for Computing the Interpolation Weights in AMG, *SIAM Journal of Scientific Computing*, in press. Also available as LLNL Technical Report **UCRL-JC-139098**

Hornung, R., and S. Kohn, *Future Directions for AMR in ASCI and Other LLNL Simulation Projects*, Lawrence Livermore National Laboratory technical report UCRL-ID-128332. 97-ERD-114.

Jones, J. and P. Vassilevski, "AMGe Based on Element Agglomeration," *SIAM J. Sci. Comp.*, in press August 1999. Also available as Lawrence Livermore National Laboratory technical report UCRL-JC-135441.

Kohn, S., J. Weare, E. Ong, S. Baden, "Hierarchical Basis Preconditioners in Three Dimensions," *SIAM J. Sci. Comp.*, **18**, pp. 479-98, Also available as Lawrence Livermore National Laboratory technical report UCRL-JC-122279 and LDRD 97-ERD-114.

Kohn, S., J. Weare, E. Ong, S. Baden, "Software Abstractions and Computational Issues in Parallel Structured Adaptive Mesh Methods for Electronic Structure Calculations," *Proc. Workshop Structured AMR Grid Methods*, Minneapolis, MN, March 12-13, 1998. Also available as Lawrence Livermore National Laboratory technical report UCRL-JC-127598.

Kohn, S., J. Weare, E. Ong, S. Baden, "Parallel Adaptive Mesh Refinement for Electronic Structure Calculations," *Proc. 8<sup>th</sup> SIAM Conf. Parallel Processing for Sci. Comp.*, Minneapolis, MN, March 12-13, 1998. Also available as Lawrence Livermore National Laboratory technical report UCRL-JC-126043.

Kohn, S., R. Hornung, X. Garaizar, "SAMRAI: Structured Adaptive Mesh Refinement Applications Infrastructure," *Supercomputing 97*. Also available as Lawrence Livermore National Laboratory technical report UCRL-TB-128634.

Kohn, S., R. Hornung, "SAMRAI: Structured Adaptive Mesh Refinement Applications Infrastructure," *Supercomputing 98*. Also available as Lawrence Livermore National Laboratory technical report UCRL-TB-128634 Rev 1.

Kohn, S., A. Cleary, S. Smith, B. Smolinski, *Language Interoperability Mechanisms for High-Performance Scientific Applications*, Lawrence Livermore National Laboratory technical report UCRL-JC-131823.

Tompson, A.F.B., et al., "Analysis of Subsurface Contaminant Migration and Remediation using High Performance Computing," *Advances in Water Resources*, **22** (3/98), pp. 201-221, Also available as Lawrence Livermore National Laboratory technical report UCRL-JC-124650.

# ALGEBRAIC MULTIGRID BASED ON ELEMENT INTERPOLATION (AMGe)

M. BREZINA\*, A. J. CLEARY†, R. D. FALGOUT†, V. E. HENSON†, J. E. JONES†,  
T. A. MANTEUFFEL\*, S. F. MCCORMICK\*, AND J. W. RUGE\*

**Abstract.** *We introduce AMGe, an algebraic multigrid method for solving the discrete equations that arise in Ritz-type finite element methods for partial differential equations. Assuming access to the element stiffness matrices, AMGe is based on the use of two local measures derived from global measures that appear in existing multigrid theory. These new measures are used to determine local representations of algebraically “smooth” error components. These representations provide the basis for constructing effective interpolation and, hence, the coarsening process for AMG. Here, we focus on the interpolation process; choice of the coarse “grids” based on these measures is the subject of current research. We develop a theoretical foundation for AMGe and present numerical results that demonstrate the efficacy of the method.*

**1. Introduction.** Computer simulations play an increasingly important role in scientific investigations. Indeed, as experimentation becomes more expensive, impracticable, or even proscribed, scientists are turning more and more to numerical simulation. Modern packages are extremely complex, with many physics components: hydrodynamics, radiation, transport, structures, thermal, chemistry, and electromagnetic, among many others. Also, the problems are frequently posed in multi-material regimes, with contact surfaces, interpenetrability constraints, and intricate geometries complicating matters. As a result, codes are being developed to solve complex multi-physics problems on highly resolved, unstructured grids. Such large grid simulations often require massively parallel computing as well as scalable numerical algorithms such as multigrid (see e.g., [1]).

Algebraic Multigrid (AMG) [5, 3, 4, 6, 19, 16, 18, 17] is a method for solving matrix equations that is based on multigrid concepts. It examines the matrix entries to determine a sequence of smaller matrix problems that serve as coarse-level equations. AMG also determines associated interlevel transfer operators (restriction and prolongation), then solves the original matrix equation in a multigrid-like process based on these automatically-constructed components. AMG has been shown to be well-suited for solving unstructured grid problems, and to work well over a wide variety of applications (see, e.g., [9]). It has been applied successfully to M-matrix problems where the so-called strength of connection is easily measured (this measure is used to determine which variables are strongly representative of the errors left by relaxation, so that they can be used to construct the coarse levels). It also applies

---

\* Applied Math Department, Campus Box 526, University of Colorado at Boulder, Boulder, CO 80309-0526. *email:*{mbrezina, tmanteuf, stevem}@colorado.edu. This work was sponsored in part by the National Science Foundation under grant number DMS-9706866 and the Department of Energy under grant number DE-FG03-93ER25165.

† Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, P.O. Box 808, L-561, Livermore, CA 94551. *email:*{cleary, rfgout, vhenson, jjones}@llnl.gov

well to scalar problems that depart substantially from M-matrix discretizations. However, for problems where strength of connection is not easily measured, AMG is not effective without certain problem-specific modifications or careful parameter tuning. For such cases, there is no systematic AMG approach that has proven effective in any kind of general context. There are still other problems (e.g., thin-body elasticity on unstructured grids) for which AMG and other iterative methods in general have failed to achieve full optimality. The goal of our research is to develop a more robust AMG method for solving these difficult problems.

This paper introduces an algebraic multigrid method for solving partial differential equations discretized by Ritz finite element methods. As a departure from standard AMG, where only the operator matrix is required, this approach assumes access to element stiffness matrices. We thus refer to it as AMGe (the acronym AMG henceforth refers to the standard scheme). This new approach is based on the use of either of two measures (derived from global measures used in existing theory) to determine algebraically “smooth” error and to construct effective interpolation. AMGe uses a minimization principle based on the *element interpolation* scheme first introduced in [15]. Other multigrid methods, using minimization principles for constructing energetically stable inter-grid transfer operators, have recently appeared in [22, 23, 11]. While the focus here is on the interpolation process, we also briefly describe our current research that is aimed at using these measures to improve the coarse-grid selection process. Although the key ideas behind AMG are summarized in the next section for clarity, we assume that the reader is familiar with AMG methods and terminology. For more detail, see [17]. For recent results and for understanding the following in context, see [9].

In the next section, we introduce some notation and review the AMG algorithm. In particular, we discuss the notion of *strength of dependence* and its role in defining the basic AMG components. In section 3, we define a heuristic based on two *global measures* and establish a corresponding two-level convergence result. We “localize” these measures in section 4, and describe how they can be used to compute the interpolation operator for AMGe. We also discuss the relationship between the local and global measures in subsection 4.3. Section 5 contains numerical results supporting the theory and demonstrating the efficacy of the approach. In section 6, we discuss preliminary approaches for selecting coarse grids based on these local measures. Concluding remarks are made in section 7.

**2. Preliminaries.** We begin this section by describing notation. Capital Roman letters ( $A, B, P, R$ ) denote matrices and bold lower case Roman and Greek letters denote vectors ( $\mathbf{u}, \mathbf{v}, \boldsymbol{\varepsilon}$ ). The  $i$ th component of the vector  $\mathbf{q}$  is denoted by  $q_i$ . Other lower case letters denote scalars, while capital caligraphic letters denote sets and spaces ( $\mathcal{C}, \mathcal{F}, \mathcal{S}$ ), with the singular exception that  $\mathcal{A}$  is used to denote finite element stiffness matrices. We define the  $A$ -inner product by  $\langle \cdot, \cdot \rangle_A := \langle A \cdot, \cdot \rangle$ , where  $\langle \cdot, \cdot \rangle$  is the standard Euclidean inner product, and the  $A$ -norm (also called the energy norm) by  $\|\cdot\|_A := \langle \cdot, \cdot \rangle_A^{1/2}$ .

Assume that we are given an  $n \times n$  symmetric positive definite matrix  $A$  expressed as the sum of a given set of finite element stiffness matrices,

$$(2.1) \quad A = \sum_{\alpha \in \mathcal{T}} \mathcal{A}_\alpha,$$

where  $\mathcal{T}$  is the set of finite elements used to discretize the problem and each  $\mathcal{A}_\alpha$  is symmetric positive semi-definite. We do not assume access to a spatial grid or the ability to create new finite element stiffness matrices.

We seek the solution  $\mathbf{u} \in \mathbb{R}^n$  to the linear system

$$(2.2) \quad A\mathbf{u} = \mathbf{f},$$

for a given  $\mathbf{f} \in \mathbb{R}^n$ . Standard iterative schemes, like Gauss-Seidel and Krylov space methods, tend to converge slowly for large-scale problems of this type that arise from partial differential equations. The difficulty is that smooth error components are typically attenuated very slowly by these simple processes, because they are based on local properties (i.e., local connections in  $A$ ). Multigrid methods attempt to correct this limitation by representing the smooth errors on increasingly coarser, and, therefore, more global levels.

To describe how system (2.2) could be solved by a multilevel method, let  $P$  be an  $n \times n_c$  *interpolation* or *prolongation* matrix, with  $n_c < n$ . We call  $P^T$  the *restriction* matrix. The two-grid method for solving (2.2) is defined as follows:

$$(2.3a) \quad \text{Relax } \nu_1 \text{ times on } A\mathbf{u} = \mathbf{f}.$$

$$(2.3b) \quad \text{Correct } \mathbf{u} \leftarrow \mathbf{u} + P(P^T A P)^{-1} P^T (\mathbf{f} - A\mathbf{u}).$$

$$(2.3c) \quad \text{Relax } \nu_2 \text{ times on } A\mathbf{u} = \mathbf{f}.$$

Note the use of  $P^T A P$  in correction step (2.3b). This so-called *Galerkin* coarse-grid operator, together with the use of  $P^T$  as the restriction operator amount to a *variational* form of multigrid. When  $A$  is symmetric, it can be shown the correction step minimizes the energy norm of fine-grid error over all possible corrections from the range of  $P$ .

To solve (2.2) in practice, one would use a multilevel method that recursively applies algorithm (2.3) to solve the linear system involving  $P^T A P$  in correction step (2.3b).

Examining (2.3) reveals that relaxation and coarse-grid correction must be chosen to complement each other; that is, error not reduced by one must be reduced by the other. In this paper, we fix the choice of relaxation, then determine interpolation. The relaxation we choose is a simple pointwise method, like Richardson, damped Jacobi, or Gauss-Seidel, that satisfies the following heuristic:

**H1:** *Error in the direction of an eigenvector associated with a large eigenvalue is rapidly reduced by relaxation, while error in the direction of an eigenvector associated with a small eigenvalue is reduced by a factor that may approach 1 as the eigenvalue approaches 0.*

Error that is not rapidly reduced by relaxation is called *algebraically smooth*. The actual character of algebraically smooth error depends on the operator and the type of relaxation, but it loosely means that the residual is small when compared to the error itself (we will be more precise about this shortly). This does not mean that the error is smooth in any geometric sense. Thus, error at a point may be very different from the errors at neighboring points, yet it might be difficult to reduce the error by relaxation. Such is the case for anisotropic problems, where algebraically smooth error that point Gauss-Seidel relaxation cannot effectively reduce can be geometrically oscillatory in the direction of small coefficients

of the PDE. In any case the interpolation matrix,  $P$ , must be defined so that algebraically smooth error is effectively eliminated in step (2.3b) and the coarse-grid equations, which involve  $P^TAP$ , are amenable to solution by a two-level method.

**2.1. AMG.** To define the multigrid components in AMG, we use the following heuristic based on special properties of  $M$ -matrices:

**H2:** *Smooth error varies slowest in the direction of strong dependence.*

Here, we say that *unknown  $i$  strongly depends on unknown  $j$*  if

$$(2.4) \quad -a_{i,j} \geq \theta \max_{k \neq i} \{-a_{i,k}\}, \quad \text{for some fixed } \theta \in (0, 1).$$

Thus, strong dependence is characterized by matrix coefficients that are large in the sense of (2.4). A typical choice for parameter  $\theta$  is 0.25.

Although AMG was developed with  $M$ -matrices in mind, in practice it is not limited to this class of problems. However, the method does rely on **H2**, and our sense of strong dependence may not be suitable for many important classes of problems.

One simple problem with which AMG has difficulty is the Poisson equation on a rectangular grid, discretized with bilinear quadrilateral elements, where the fine-grid elements are stretched to a 10 : 1 aspect ratio. This yields the coefficient stencil

$$(2.5) \quad \begin{bmatrix} -1 & -3.9 & -1 \\ 1.9 & 8 & 1.9 \\ -1 & -3.9 & -1 \end{bmatrix}.$$

In (2.5), it is not readily apparent from the size of the off-diagonal entries that the direction of strongest dependence is vertical. Since **H2** is used to define all of the AMG components, and it requires a clear understanding of strong dependence, AMG can exhibit degrading performance (see Table 5.1). For this simple case, slow convergence of AMG can be ameliorated by simply tuning its parameters (e.g., setting  $\theta = 0.5$ ) or by more elaborate algorithmic “fixes” (e.g., *iterative weight interpolation* [9] or geometric/algebraic interpolation methods [10, 8, 7]). Another approach is to replace **H2** by a heuristic that leads to a more robust AMG algorithm. Exploring this possibility, as we begin to do in the next section, is the primary aim of this paper.

**3. Global Measures and Convergence Bounds.** This paper takes a slightly different approach, using a heuristic based not on  $M$ -matrices but on the eigenvectors of  $A$ . In a two-grid scheme, coarse-grid correction will completely eliminate error in  $\text{Range}(P)$ , the range of the interpolation operator  $P$ . To complement the action of relaxation, which satisfies **H1**, the interpolation matrix must satisfy the following heuristic.

**H3:** *Interpolation must be able to approximate an eigenvector with error bound proportional to the size of the associated eigenvalue.*

To make **H3** more rigorous, define  $Q : \mathbb{R}^n \rightarrow \mathbb{R}^n$  to be a convenient projection onto  $\text{Range}(P)$ , that is,

$$(3.1) \quad Q = PR,$$

for some restriction operator  $R : \mathbb{R}^n \rightarrow \mathbb{R}^{n_c}$  such that  $RP = I_c$ , the identity on  $\mathbb{R}^{n_c}$ . The specific form for  $Q$  (and, hence,  $R$ ) will not become important until section 4. For any vector  $\mathbf{e} \in \text{Range}(P)$ , we have  $Q\mathbf{e} = \mathbf{e}$ . Thus,  $I - Q$  can be used to measure the defect of interpolation. With this in mind, we now define two measures of how well **H3** is satisfied:

$$(3.2) \quad M_1(Q, \mathbf{e}) := \frac{\langle (I - Q)\mathbf{e}, (I - Q)\mathbf{e} \rangle}{\langle A\mathbf{e}, \mathbf{e} \rangle},$$

$$(3.3) \quad M_2(Q, \mathbf{e}) := \frac{\langle A(I - Q)\mathbf{e}, (I - Q)\mathbf{e} \rangle}{\langle A\mathbf{e}, A\mathbf{e} \rangle}.$$

The measure  $M_2$  was used in the early multigrid theory [14, 12, 13] to establish optimal convergence of the V-cycle algorithm under full regularity assumptions on the associated partial differential equation. The measure  $M_1$  was introduced in [4] and used more recently to establish convergence, independent of the coarse-grid size, of a two-level method for linear elasticity [21]. It is also an essential ingredient of the regularity-free multilevel theory found in [2]. We develop the relevant two-grid theory here for both measures so that we can tailor the results to our needs.

It has not been our practice to use diagonal conditioning of  $A$  in conventional AMG. Such a scaling generally changes the nature of smooth errors. Since current schemes at some point rely on a premise of how smooth error behaves (e.g., that it is locally constant), then diagonal scaling can make it more difficult for AMG to handle. However, no such premise of smoothness is made anywhere in AMGe. Thus, in the remainder of this paper, we are free to assume for convenience that matrix  $A$  has been scaled so that its diagonal is the identity. For a general symmetric positive-definite matrix with diagonal  $D \neq I$ , this can be assured by replacing  $A$  with  $D^{-1/2}AD^{-1/2}$ . Note that this transformation must be considered in the representation of  $A$  as a sum of local stiffness matrices, but this is just a straightforward rescaling of the variables. This scaling does, however, bear on the practicality of our results because we analyze AMG based on Richardson iteration, which is not generally a good smoother for matrices that have widely varying diagonal entries. For such matrices, a damped Jacobi method with proper under-relaxation should be used, but then measures  $M_1$  and  $M_2$  must be changed accordingly (e.g., for  $M_1$  in (3.2),  $D$  would appear in the numerator's inner product).

This theory assumes that either  $M_1$  or  $M_2$  be bounded uniformly in  $\mathbf{e} \in \mathbb{R}^n \setminus \{0\}$ . To see how this assumption relates to **H3**, suppose that  $\mathbf{e}$  is an eigenvector of  $A$  corresponding to a small eigenvalue. Then, for  $M_1$  or  $M_2$  to be bounded, since the denominators of the two measures are small, the numerators must also be small. Thus,  $Q$  must accurately interpolate eigenvectors belonging to small eigenvalues. On the other hand, if  $\mathbf{e}$  is an eigenvector of  $A$  corresponding to a large eigenvalue, then the denominators of the two measures are large, so the numerators may be large. Thus,  $Q$  need not accurately interpolate eigenvectors belonging to large eigenvalues.

We now prove convergence results based on  $M_1$  or  $M_2$  for the two-level algorithm (2.3).

**LEMMA 3.1.** *Let  $Q$  be any projection onto  $\text{Range}(P)$ . Assume that either of the following two approximation properties are satisfied for some constant  $K$ :*

$$(3.4) \quad M_1(Q, \mathbf{e}) \leq K \quad \forall \mathbf{e} \in \mathbb{R}^n \setminus \{0\},$$

$$(3.5) \quad M_2(Q, \mathbf{e}) \leq K \quad \forall \mathbf{e} \in \mathbb{R}^n \setminus \{0\}.$$

If  $\mathbf{e} \neq 0$  is  $A$ -orthogonal to  $\text{Range}(P)$ , then

$$(3.6) \quad \frac{1}{K} \leq \frac{\|A\mathbf{e}\|^2}{\langle A\mathbf{e}, \mathbf{e} \rangle} \leq \|A\|.$$

*Proof.* The upper bound in (3.6) follows easily from the definition of the matrix norm. To prove the lower bound, note that  $\text{Range}(Q) = \text{Range}(P)$ . Hence, if  $\mathbf{e}$  is  $A$ -orthogonal to  $\text{Range}(P)$ , then

$$(3.7) \quad \langle A\mathbf{e}, Q\mathbf{v} \rangle = 0 \quad \forall \mathbf{v} \in \mathbb{R}^n.$$

First, assume that (3.4) holds. From (3.7) and the Cauchy-Schwartz inequality, we have

$$\begin{aligned} \langle A\mathbf{e}, \mathbf{e} \rangle &= \langle A\mathbf{e}, (I - Q)\mathbf{e} \rangle \\ &\leq \|A\mathbf{e}\| \|(I - Q)\mathbf{e}\| \\ &\leq \|A\mathbf{e}\| \langle A\mathbf{e}, \mathbf{e} \rangle^{1/2} K^{1/2}. \end{aligned}$$

The lower bound in (3.6) now follows by dividing through by  $\langle A\mathbf{e}, \mathbf{e} \rangle K^{1/2}$  and squaring the result.

Now, assume that (3.5) holds. From (3.7) and the Cauchy-Schwartz inequality, we have

$$\begin{aligned} \langle A\mathbf{e}, \mathbf{e} \rangle &\leq \langle A\mathbf{e}, \mathbf{e} \rangle + \langle A Q \mathbf{e}, Q \mathbf{e} \rangle \\ &= \langle A\mathbf{e}, \mathbf{e} \rangle - \langle A\mathbf{e}, Q\mathbf{e} \rangle - \langle A Q \mathbf{e}, \mathbf{e} \rangle + \langle A Q \mathbf{e}, Q \mathbf{e} \rangle \\ &= \langle A(I - Q)\mathbf{e}, (I - Q)\mathbf{e} \rangle \\ &\leq \|A\mathbf{e}\|^2 K. \end{aligned}$$

The lower bound in (3.6) now follows by dividing through by  $\langle A\mathbf{e}, \mathbf{e} \rangle K$ . ■

Define the  $A$ -orthogonal projection onto the  $\text{Range}(P)$ :

$$(3.8) \quad S := P(P^T A P)^{-1} P^T A.$$

The error propagation matrix for the coarse-grid correction step (2.3b) is  $I - S$ . A Richardson iteration with relaxation parameter  $s = \omega / \|A\|$ ,  $\omega \in (0, 2)$ , has the error propagation matrix  $G = I - sA$ . If we choose  $(\nu_1, \nu_2) = (0, 1)$  in (2.3), then the associated error propagation matrix for this simple two-grid scheme is  $G(I - S)$ . The following theorem analyzes its convergence by bounding its error propagation matrix in the  $A$ -norm. Convergence results for other values of  $(\nu_1, \nu_2)$  then follow naturally [13].

Analogous multilevel results can be found in [14, 12, 13] for approximation property (3.5), and in [2, 20] for (3.4) under the additional assumption of *energetic stability of interpolation*, which requires that  $\|P(P^T P)^{-1} P^T\|_A$  is bounded uniformly on all levels.

**THEOREM 3.2.** *Assume that either approximation property (3.4) or (3.5) is satisfied for some constant  $K$ . Then*

$$(3.9) \quad \|G(I - S)\|_A \leq \left(1 - \frac{\omega(2 - \omega)}{K \|A\|}\right)^{1/2}.$$

*Proof.* First note that (3.6) implies  $K \geq 1/\|A\| \geq \omega(2-\omega)/\|A\|$ , so that (3.9) makes sense. We have

$$\begin{aligned} \langle AGe, Ge \rangle &= \langle Ae, e \rangle - 2s \langle Ae, Ae \rangle + s^2 \langle A^2e, Ae \rangle \\ &\leq \langle Ae, e \rangle - \frac{\omega(2-\omega)}{\|A\|} \langle Ae, Ae \rangle. \end{aligned}$$

Replacing  $e$  with  $(I-S)e$  and applying the result in Lemma 3.1 yields

$$\begin{aligned} \|G(I-S)e\|_A^2 &\leq \langle A(I-S)e, (I-S)e \rangle - \frac{\omega(2-\omega)}{\|A\|} \|A(I-S)e\|^2 \\ &\leq \left(1 - \frac{\omega(2-\omega)}{K\|A\|}\right) \|e\|_A^2. \end{aligned}$$

■

Notice that the bound on the convergence factor approaches 1 as  $K$  becomes large. Conversely, smaller  $K$  yields a smaller bound on the convergence factor. Our aim is to determine  $P$  so that, for some appropriate  $Q$ , either (3.4) or (3.5) is satisfied for a reasonably small  $K$ .

We also remark that the above results can be generalized to apply when (2.2) is a consistent system with symmetric positive semi-definite matrix  $A$ . Measures  $M_1$  and  $M_2$  must be restricted to  $e \notin \text{Null}(A)$ . A finite bound  $K$  in (3.4) or (3.5) then implies that interpolation is exact for  $e \in \text{Null}(A)$ , which in turn implies that the correction step involves a consistent system. A zero initial guess and relaxation using a polynomial method like Richardson iteration ensures that the approximate solution remains orthogonal to  $\text{Null}(A)$ .

**4. Interpolation Using Local Measures.** The quantities  $M_1$  and  $M_2$  are global measures of the quality of interpolation. Our intent is to use these measures to determine an effective strategy for constructing interpolation in AMG, but it is not practical to do this globally. In this section, we discuss an approach for localizing these measures for linear systems (2.2) that arise from finite element discretizations.

Recall that  $A$  is given as the sum of finite element stiffness matrices:  $A = \sum_{\alpha \in \mathcal{T}} \mathcal{A}_\alpha$ . While we do not assume access to an underlying spatial grid (see (2.1)), we can construct an artificial grid based on the graph associated with  $A$ , with vertices  $\mathcal{G} := \{1, 2, \dots, n\}$  and edges  $\mathcal{E} := \{(i, j) : a_{ij} \neq 0 \text{ for } i \neq j\}$ . Grid point (vertex)  $i \in \mathcal{G}$  is associated with unknown  $u_i$ .

We first define the point set of an element,

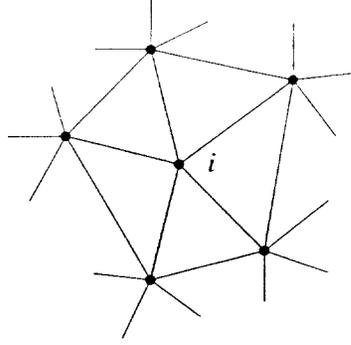
$$(4.1) \quad \mathcal{M}_\alpha := \{j : \varepsilon_j^T \mathcal{A}_\alpha \varepsilon_j \neq 0\},$$

where  $\varepsilon_j$  is the canonical basis vector associated with unknown  $j$ . Next, define the neighborhood of grid point  $i$  as the set of elements and set of points

$$(4.2) \quad \mathcal{T}_i := \{\alpha \in \mathcal{T} : \varepsilon_i^T \mathcal{A}_\alpha \varepsilon_i \neq 0\},$$

$$(4.3) \quad \mathcal{N}_i := \cup_{\alpha \in \mathcal{T}_i} \mathcal{M}_\alpha,$$

$$(4.4)$$

FIG. 4.1. *Local neighborhoods.*

respectively (see Figure 4.1). Define the local matrices on neighborhood  $i$  by

$$(4.5) \quad A_i = \sum_{\alpha \in \mathcal{T}_i} \mathcal{A}_\alpha.$$

We also assume that a coarse grid has been selected, that is, the points in  $\mathcal{G}$  have been partitioned into coarse-grid points  $\mathcal{C}$  and fine-grid points  $\mathcal{F}$  such that  $\mathcal{C} \cup \mathcal{F} = \mathcal{G}$  and  $\mathcal{C} \cap \mathcal{F} = \emptyset$ . We now seek the  $n \times n_c$  interpolation matrix  $P$ , where  $n_c = |\mathcal{C}|$ , that interpolates from the coarse-grid points  $\mathcal{C}$  to the entire grid  $\mathcal{G}$ .

Two conflicting goals drive the construction of  $P$ . The first is to minimize the bound on measure  $M_1$  or  $M_2$ , while the second is to preserve the sparsity of the coarse-grid system involving  $P^T A P$ . Focusing on the second goal first, we assume that the coarse-grid points interpolate to themselves exactly, that is,  $P$  restricted to  $\mathcal{C}$  is the identity, while fine-grid points interpolate only from coarse-grid points in their neighborhood, that is, from  $\mathcal{C}_i := \mathcal{N}_i \cap \mathcal{C}$ .

To make the construction more clear, suppose that the rows and columns of  $A$  have been arranged so that the fine-grid points come first, followed by the coarse-grid points. We may then write  $A$  in block form as follows:

$$(4.6) \quad A = \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix}.$$

In this context, the interpolation matrix has the block form

$$(4.7) \quad P = \begin{bmatrix} P_{fc} \\ I_c \end{bmatrix}.$$

Alternatively, we may define the projection

$$(4.8) \quad Q = \begin{bmatrix} 0 & P_{fc} \\ 0 & I_c \end{bmatrix},$$

which implies the choice of  $R = [0, I_c]$  as the restriction in (3.1).

In what follows, we develop a strategy for constructing the rows of  $P_{fc}$ , that is, the rows of  $Q$  corresponding to each point  $i \in \mathcal{F}$ , which we denote

$$(4.9) \quad \mathbf{q}_i^T := \boldsymbol{\varepsilon}_i^T Q.$$

Restricting interpolation to a neighborhood of coarse-grid points is equivalent to choosing

$$(4.10) \quad \mathbf{q}_i \in \mathcal{Z}_i := \{\mathbf{v} \in \mathbb{R}^n : v_j = 0 \text{ for } j \notin \mathcal{C}_i\}.$$

We now localize measures  $M_1$  and  $M_2$  by defining

$$(4.11) \quad M_{i,1}(Q, \mathbf{e}) := \frac{\langle \boldsymbol{\varepsilon}_i \boldsymbol{\varepsilon}_i^T (I - Q) \mathbf{e}, \boldsymbol{\varepsilon}_i \boldsymbol{\varepsilon}_i^T (I - Q) \mathbf{e} \rangle}{\langle A_i \mathbf{e}, \mathbf{e} \rangle},$$

$$(4.12) \quad M_{i,2}(Q, \mathbf{e}) := \frac{\langle A_i \boldsymbol{\varepsilon}_i \boldsymbol{\varepsilon}_i^T (I - Q) \mathbf{e}, \boldsymbol{\varepsilon}_i \boldsymbol{\varepsilon}_i^T (I - Q) \mathbf{e} \rangle}{\langle A_i \mathbf{e}, A_i \mathbf{e} \rangle},$$

for any  $\mathbf{e} \notin \text{Null}(A_i)$ . Notice for  $i \in \mathcal{C}$  that  $M_{i,1} = M_{i,2} = 0$ , while for  $i \in \mathcal{F}$  the above measures only depend on the  $i$ th row of  $Q$ , which is to be chosen in  $\mathcal{Z}_i$ . To emphasize this dependence, when the meaning is clear we write

$$(4.13) \quad M_{i,1}(\mathbf{q}_i, \mathbf{e}) = \frac{\langle (\boldsymbol{\varepsilon}_i - \mathbf{q}_i)^T \mathbf{e}, (\boldsymbol{\varepsilon}_i - \mathbf{q}_i)^T \mathbf{e} \rangle}{\langle A_i \mathbf{e}, \mathbf{e} \rangle},$$

$$(4.14) \quad M_{i,2}(\mathbf{q}_i, \mathbf{e}) = \frac{\langle (\boldsymbol{\varepsilon}_i - \mathbf{q}_i)^T \mathbf{e}, (\boldsymbol{\varepsilon}_i - \mathbf{q}_i)^T \mathbf{e} \rangle}{\langle A_i \mathbf{e}, A_i \mathbf{e} \rangle},$$

for  $\mathbf{q}_i \in \mathcal{Z}_i$  and  $\mathbf{e} \notin \text{Null}(A_i)$ . (Recall that  $A$  has unit diagonal.)

Heuristic **H3**, as applied to these local measures, now relates interpolation accuracy to local eigenvectors of  $A_i$ . This makes it practical to use  $M_{i,1}$  and  $M_{i,2}$  to compute interpolation. Since we wish to make these local measures small, interpolation is defined so that the  $\mathbf{q}_i$  in (4.9) is the arg min (that is, the argument that attains the minimum) of one of the following min-max problems:

$$(4.15) \quad K_{i,p} := \min_{\mathbf{q}_i \in \mathcal{Z}_i} \max_{\mathbf{e} \notin \text{Null}(A_i)} M_{i,p}(\mathbf{q}_i, \mathbf{e}),$$

for  $p = 1$  or  $2$ . Note that if there exists a  $\mathbf{q}_i \in \mathcal{Z}_i$  that yields  $K_{i,p} < \infty$ , then  $\mathbf{q}_i$  satisfies the constraint

$$(\boldsymbol{\varepsilon}_i - \mathbf{q}_i)^T \mathbf{e} = 0 \quad \forall \mathbf{e} \in \text{Null}(A_i).$$

Thus, the min-max problem (4.15) can be restated as the constrained min-max problem:

$$(4.16) \quad K_{i,p} = \min_{\mathbf{q}_i \in \mathcal{Z}_i} \max_{\mathbf{e} \perp \text{Null}(A_i)} M_{i,p}(\mathbf{q}_i, \mathbf{e}), \quad \text{subject to } (\boldsymbol{\varepsilon}_i - \mathbf{q}_i)^T \mathbf{e} = 0 \quad \forall \mathbf{e} \in \text{Null}(A_i),$$

for  $p = 1$  or  $2$ . The next two subsections focus on the solution of these min-max problems. In Section 4.3, we relate the local measures to the global measures.

**4.1. Computing Interpolation by Fitting Eigenvectors.** One way to compute the  $\mathbf{q}_i$  in (4.15) or (4.16) is to “fit” the eigenvectors of  $A_i$ , as quantified in the following theorem.

**THEOREM 4.1.** *Suppose we have computed the eigen-decomposition*

$$(4.17) \quad A_i V_i = V_i \Lambda_i, \quad V_i^T V_i = I.$$

The columns of  $V_i$  are the orthonormalized eigenvectors of  $A_i$ , and the diagonal entries of  $\Lambda_i$  are the corresponding eigenvalues. Assume that this eigen-decomposition is ordered to distinguish between zero and positive eigenvalues:

$$(4.18) \quad V_i = \begin{bmatrix} V_{i0} & V_{i+} \end{bmatrix}, \quad \Lambda_i = \begin{bmatrix} 0 & 0 \\ 0 & \Lambda_{i+} \end{bmatrix}.$$

Then min-max problem (4.16) is equivalent to the following constrained least-squares problem:

$$(4.19) \quad \min_{\mathbf{q}_i} \left\| \Lambda_{i+}^{-p/2} V_{i+}^T (\boldsymbol{\varepsilon}_i - \mathbf{q}_i) \right\|^2, \quad \text{subject to} \quad V_{i0}^T (\boldsymbol{\varepsilon}_i - \mathbf{q}_i) = 0,$$

for  $p = 1$  or  $2$ .

*Proof.* Note that the null-space constraint in (4.16) is equivalent to that in (4.19). Assume first that  $\mathbf{q}_i$  satisfies (4.16) with  $p = 1$ . Since  $\mathbf{e} \perp \text{Null}(A_i)$ , we can write  $\mathbf{e} = V_{i+} \Lambda_{i+}^{-1/2} \mathbf{w}$ , which yields

$$\begin{aligned} \min_{\mathbf{q}_i \in \mathcal{Z}_i} \max_{\mathbf{e} \perp \text{Null}(A_i)} M_{i,1}(\mathbf{q}_i, \mathbf{e}) &= \min_{\mathbf{q}_i \in \mathcal{Z}_i} \max_{\mathbf{w}} \frac{\left\| (\boldsymbol{\varepsilon}_i - \mathbf{q}_i)^T V_{i+} \Lambda_{i+}^{-1/2} \mathbf{w} \right\|^2}{\|\mathbf{w}\|^2} \\ &= \min_{\mathbf{q}_i \in \mathcal{Z}_i} \left\| \Lambda_{i+}^{-1/2} V_{i+}^T (\boldsymbol{\varepsilon}_i - \mathbf{q}_i) \right\|^2. \end{aligned}$$

Assume now that  $\mathbf{q}_i$  satisfies (4.16) with  $p = 2$ . Writing  $\mathbf{e} = V_{i+} \Lambda_{i+}^{-1} \mathbf{w}$ , we then have

$$\begin{aligned} \min_{\mathbf{q}_i \in \mathcal{Z}_i} \max_{\mathbf{e} \perp \text{Null}(A_i)} M_{i,2}(\mathbf{q}_i, \mathbf{e}) &= \min_{\mathbf{q}_i \in \mathcal{Z}_i} \max_{\mathbf{w}} \frac{\left\| (\boldsymbol{\varepsilon}_i - \mathbf{q}_i)^T V_{i+} \Lambda_{i+}^{-1} \mathbf{w} \right\|^2}{\|\mathbf{w}\|^2} \\ &= \min_{\mathbf{q}_i \in \mathcal{Z}_i} \left\| \Lambda_{i+}^{-1} V_{i+}^T (\boldsymbol{\varepsilon}_i - \mathbf{q}_i) \right\|^2. \end{aligned}$$

■

Computing the interpolation weights  $\mathbf{q}_i$  using (4.19) requires eigen-decomposition (4.17), which is not the most efficient method. We introduce a simpler approach in the next subsection. However, we include this notion of fitting eigenvectors because it is useful for understanding the basic principles involved in selecting interpolation.

**4.2. A More Practical Algorithm for Computing Interpolation.** This subsection describes a practical algorithm for determining when (4.15) or (4.16) has a (unique) solution for  $i \in \mathcal{F}$ , and for computing  $Q$  when a solution does exist. One important consequence of this characterization is that, whenever the solution with the current interpolatory set does not exist, we can add points to  $\mathcal{C}_i$  until a solution does exist.

Assume first that grid point  $i \in \mathcal{F}$  has a neighborhood, as depicted in Figure 4.1, consisting of  $n_i$  points in set  $\mathcal{N}_i$ , with  $n_f$  fine-grid points and  $n_c$  coarse-grid points in  $\mathcal{C}_i$ . Next, order the unknowns and equations of matrix  $A_i$  so that unknown  $i$  is first, followed by the other fine-grid points, with the coarse-grid points last. The neighborhood matrix and its square can then be written as

$$A_i = \begin{bmatrix} A_{ff}^{(1)} & A_{fc}^{(1)} \\ A_{cf}^{(1)} & A_{cc}^{(1)} \end{bmatrix} \quad \text{and} \quad A_i^2 = \begin{bmatrix} A_{ff}^{(2)} & A_{fc}^{(2)} \\ A_{cf}^{(2)} & A_{cc}^{(2)} \end{bmatrix},$$

respectively, and  $\varepsilon_i$  becomes  $\varepsilon_1$ .

In the remainder of this subsection, we drop the subscript  $i$  whenever the meaning is clear. Set  $\mathcal{Z}_i$  restricted to the neighborhood becomes

$$\mathcal{Z} := \{\mathbf{e} \in \mathbb{R}^{n_i} : e_j = 0 \forall j \notin C_i\}.$$

We can then interpret (4.16) with  $p = 1$  or  $2$  as the problem of determining a vector  $\mathbf{q} \in \mathcal{Z}$  that minimizes  $\max_{\mathbf{e} \in \text{Null}(A_i)} M_{i,p}(\mathbf{q}, \mathbf{e})$ , subject to the constraint

$$(\varepsilon_1 - \mathbf{q})^T \mathbf{e} = 0 \quad \forall \mathbf{e} \in \text{Null}(A_i) = \text{Null}(A_i^2).$$

That is, we require

$$(4.20) \quad \varepsilon_1 - \mathbf{q} \in \text{Range}(A_i) = \text{Range}(A_i^2).$$

Our first concern is the existence of such a vector  $\mathbf{q}$ . For this, we let  $\hat{\varepsilon}_1 \in \mathbb{R}^{n_f}$  denote the first canonical basis vector of length  $n_f$ .

LEMMA 4.2. *There exists  $\mathbf{q} \in \mathcal{Z}$  such that  $\varepsilon_1 - \mathbf{q} \in \text{Range}(A_i^p)$  if and only if*

$$\hat{\varepsilon}_1 \in \text{Range}(A_{ff}^{(p)}),$$

with  $p = 1$  or  $2$ .

*Proof.* Assume that  $\hat{\varepsilon}_1 \in \text{Range}(A_{ff}^{(p)})$  so that

$$\hat{\varepsilon}_1 = A_{ff}^{(p)} \hat{\delta}_1$$

for some  $\hat{\delta}_1 \in \mathbb{R}^{n_f}$ . Then

$$A_i^p \delta := \begin{bmatrix} A_{ff}^{(p)} & A_{fc}^{(p)} \\ A_{cf}^{(p)} & A_{cc}^{(p)} \end{bmatrix} \begin{pmatrix} \hat{\delta}_1 \\ 0 \end{pmatrix} = \varepsilon_1 - \mathbf{q} \in \text{Range}(A_i^p),$$

and  $\mathbf{q} \in \mathcal{Z}$ .

Conversely, suppose there exists  $\mathbf{q} \in \mathcal{Z}$  such that  $\varepsilon_1 - \mathbf{q} \in \text{Range}(A_i^p)$ , that is, there exists  $\delta$  such that

$$\varepsilon_1 - \mathbf{q} = A_i^p \delta.$$

This, in turn, implies that

$$\hat{\varepsilon}_1 = [A_{ff}^{(p)}, A_{fc}^{(p)}] \delta \in \text{Range}([A_{ff}^{(p)}, A_{fc}^{(p)}]).$$

The proof will be completed by demonstrating that

$$\text{Range}([A_{ff}^{(p)}, A_{fc}^{(p)}]) = \text{Range}(A_{ff}^{(p)}).$$

This is certainly true if  $A_{ff}^{(p)}$  is nonsingular. Assume otherwise, and let  $\hat{\delta}$  be a nonzero vector in  $\text{Null}(A_{ff}^{(p)})$ . Then

$$\left\langle \begin{bmatrix} A_{ff}^{(p)} & A_{fc}^{(p)} \\ A_{cf}^{(p)} & A_{cc}^{(p)} \end{bmatrix} \begin{pmatrix} \hat{\delta} \\ 0 \end{pmatrix}, \begin{pmatrix} \hat{\delta} \\ 0 \end{pmatrix} \right\rangle = 0.$$

Since  $A_i^p$  is symmetric positive semi-definite, 0 is an extreme value of  $\langle A_i^p \mathbf{e}, \mathbf{e} \rangle$ , which implies that the vector  $(\hat{\delta}, 0)^T$  is an eigenvector of  $A_i^p$  with eigenvalue 0. In other words,  $(\hat{\delta}, 0)^T \in \text{Null}(A_i^p)$ , which implies that

$$\text{Null}(A_{ff}^{(p)}) = \text{Null}(A_{cf}^{(p)}),$$

which, in turn, implies that

$$\text{Range}(A_{ff}^{(p)}) = \text{Range}\left(\left(A_{cf}^{(p)}\right)^T\right) = \text{Range}(A_{fc}^{(p)}),$$

and the lemma is proved. ■

Rewriting (4.20), we want  $\delta \in \mathbb{R}^{n_i}$  such that

$$\varepsilon_1 - \mathbf{q} = A_i^p \delta$$

for some  $\mathbf{q} \in \mathcal{Z}$ . By the proof of Lemma 4.2, the set of all such  $\delta$  is

$$Y^{(p)} := \left\{ \delta \in \mathbb{R}^{n_i} : \begin{bmatrix} A_{ff}^{(p)} & A_{fc}^{(p)} \end{bmatrix} \delta = \hat{\varepsilon}_1 \right\}.$$

If  $Y^{(p)}$  is empty, then the constraint in (4.16) cannot be satisfied and  $K_{i,p} = \infty$ . In this case, more points must be added to  $C_i$  for (4.16) to have a solution. If  $Y^{(p)}$  is not empty, then any  $\delta \in Y^{(p)}$  can be written as  $\delta = \delta^* + \gamma$ , where  $\delta^*$  is a particular element of  $Y^{(p)}$  and  $\gamma \in \text{Null}([A_{ff}^{(p)}, A_{fc}^{(p)}])$ . From the proof of Lemma 4.2, we may choose  $\delta^* = (\hat{\delta}_1, 0)^T$ , where  $A_{ff}^{(p)} \hat{\delta}_1 = \hat{\varepsilon}_1$ . We now show that

$$\varepsilon_1 - \mathbf{q}^* = A_i^p \delta^*$$

yields the unique solution to (4.15) or (4.16).

**THEOREM 4.3.** *If  $\hat{\varepsilon}_1 \notin \text{Range}(A_{ff}^{(p)})$ , then  $K_{i,p} = \infty$ . If  $\hat{\varepsilon}_1 = A_{ff}^{(p)} \hat{\delta}_1$ , then the unique solution of (4.15) is given by*

$$(4.21) \quad \mathbf{q}^* = \begin{pmatrix} 0 \\ -A_{cf}^{(p)} \hat{\delta}_1 \end{pmatrix} \in \mathcal{Z},$$

and  $K_{i,p} = \langle \hat{\varepsilon}_1, \hat{\delta}_1 \rangle$ , for  $p = 1$  or  $2$ .

*Proof.* The first statement follows from Lemma 4.2. To prove the second, let  $\delta^* = (\hat{\delta}_1, 0)^T$ . Using the substitution

$$\varepsilon_1 - \mathbf{q} = A^{(p)} \delta$$

with  $\delta \in Y^{(p)}$ , then (4.15) can be written as

$$(4.22) \quad \min_{\mathbf{q} \in \mathcal{Z}} \max_{\mathbf{e} \notin \text{Null}(A_i^p)} \frac{\langle (\varepsilon_1 - \mathbf{q})^T \mathbf{e}, (\varepsilon_1 - \mathbf{q})^T \mathbf{e} \rangle}{\langle A_i^p \mathbf{e}, \mathbf{e} \rangle} = \min_{\delta \in Y^{(p)}} \langle A_i^p \delta, \delta \rangle \\ = \min_{\gamma \in \text{Null}([A_{ff}^{(p)}, A_{fc}^{(p)}])} \langle A_i^p (\delta^* + \gamma), (\delta^* + \gamma) \rangle.$$

Any solution of (4.22) is characterized by  $\gamma^* \in \text{Null}([A_{ff}^{(p)}, A_{fc}^{(p)}])$  such that

$$(4.23) \quad \langle A_i^p (\delta^* + \gamma^*), \gamma \rangle = 0 \quad \forall \gamma \in \text{Null}([A_{ff}^{(p)}, A_{fc}^{(p)}]);$$

that is,

$$(4.24) \quad A_i^p (\delta^* + \gamma^*) \in \text{Range} \left( \begin{bmatrix} A_{ff}^{(p)} \\ A_{cf}^{(p)} \end{bmatrix} \right).$$

But  $\gamma^* = 0$  satisfies (4.23) by construction of  $\delta^*$ , which proves that (4.21) solves (4.15).

To prove uniqueness, suppose there are two such solutions to (4.23), say,  $\delta^*$  and  $\beta^*$ . Then

$$A_i^p (\delta^* - \beta^*) = \begin{bmatrix} A_{ff}^{(p)} \\ A_{cf}^{(p)} \end{bmatrix} \hat{\mathbf{w}}$$

for some  $\hat{\mathbf{w}} \in \mathbb{R}^{n_f}$ . Since both  $\delta^*$  and  $\beta^*$  are in  $Y^{(p)}$ , we have  $\hat{\mathbf{w}} \in \text{Null}(A_{ff}^{(p)})$ . From Lemma 4.2, we have  $\text{Null}(A_{ff}^{(p)}) = \text{Null}(A_{cf}^{(p)})$ , which implies that  $A_i^p (\delta^* - \beta^*) = 0$  and that  $\mathbf{q}^*$  is unique.

Finally, substituting  $\delta^*$  into (4.22) yields

$$K_{i,p} = \langle A_i^p \delta^*, \delta^* \rangle = \langle \hat{\varepsilon}_1, \hat{\delta}_1 \rangle,$$

which completes the proof. ■

A practical algorithm for determining  $Q$  is as follows:

For  $p = 1$ , set

$$A_{ff}^{(1)} = A_{ff}, \quad A_{cf}^{(1)} = A_{cf}.$$

For  $p = 2$ , set

$$A_{ff}^{(2)} = A_{ff}^2 + A_{fc} A_{cf}, \quad A_{cf}^{(2)} = A_{cf} A_{ff} + A_{cc} A_{cf}.$$

Perform a  $QR$  factorization on  $A_{ff}^{(p)}$  using Householder reflections and column pivoting to detect rank deficiency. If

$$A_{ff}^{(p)} \hat{\delta}_1 = \hat{\varepsilon}_1$$

has a solution, then set

$$\mathbf{q}^* = \begin{pmatrix} 0 \\ -A_{cf}^{(p)} \hat{\delta}_1 \end{pmatrix}$$

and  $K_{i,p} = \langle \hat{\varepsilon}_1, \hat{\delta}_1 \rangle$ ; otherwise, set  $K_{i,p} = \infty$ .

**4.3. Local-Global Measure.** This subsection shows that if  $M_{i,1}$  or  $M_{i,2}$  is bounded for every  $i \in \mathcal{F}$ , then the global measure  $M_1$  is also bounded.

**THEOREM 4.4.** *Let  $p = 1$  or  $2$  and assume that the local approximation property*

$$(4.25) \quad M_{i,p}(Q, \mathbf{e}) \leq K_{i,p} \quad \forall \mathbf{e} \in \mathbb{R}^n$$

*holds for some  $K_{i,p}$  and all  $i \in \mathcal{F}$ . Then global approximation property (3.4) is also satisfied with*

$$(4.26) \quad K = \max_{\alpha \in \mathcal{T}} \sum_{i \in \mathcal{M}_\alpha \cap \mathcal{F}} K_{i,p} \|A_i\|^{p-1}.$$

*Proof.* We have

$$\begin{aligned} \langle (I - Q)\mathbf{e}, (I - Q)\mathbf{e} \rangle &= \sum_{i \in \mathcal{F}} \langle \boldsymbol{\varepsilon}_i \boldsymbol{\varepsilon}_i^T (I - Q)\mathbf{e}, \boldsymbol{\varepsilon}_i \boldsymbol{\varepsilon}_i^T (I - Q)\mathbf{e} \rangle \\ &\leq \sum_{i \in \mathcal{F}} K_{i,p} \langle A_i^p \mathbf{e}, \mathbf{e} \rangle \\ &\leq \sum_{i \in \mathcal{F}} K_{i,p} \|A_i\|^{p-1} \langle A_i \mathbf{e}, \mathbf{e} \rangle \\ &= \sum_{\alpha \in \mathcal{T}} \langle A_\alpha \mathbf{e}, \mathbf{e} \rangle \sum_{i \in \mathcal{M}_\alpha \cap \mathcal{F}} K_{i,p} \|A_i\|^{p-1} \\ &\leq K \sum_{\alpha \in \mathcal{T}} \langle A_\alpha \mathbf{e}, \mathbf{e} \rangle \\ &= K \langle A\mathbf{e}, \mathbf{e} \rangle. \end{aligned}$$

■

Straightforward application of the above techniques can be used to bound  $M_2$  in terms of  $M_{i,2}$ . However, the resulting bounds on  $M_2$  can be much larger than the maximum value of  $M_{i,2}$ . While this may not be sharp, it is simple to construct an example where  $M_2$  is much larger than the largest  $M_{i,2}$  and, hence, much larger than  $M_1$ . In this case, using  $M_2$  to estimate convergence could lead to the erroneous conclusion that the resulting two-level method is slow to converge.

The local measure bounds,  $K_{i,p}$ , can be used as a diagnostic tool: Theorem 4.4 shows that they contribute to the bound  $K$ , used to establish convergence in Theorem 3.2. While neither measure provides a sharp bound when the algorithm exhibits a small convergence factor, they can provide a warning: if  $K_{i,p}$  is large for some  $i$ , it may be profitable to reexamine the choice of the coarse grid, perhaps adding more grid points to  $\mathcal{C}$ . This will be discussed in greater detail in section 6.

As an alternative to increasing the size of  $\mathcal{C}$ , we could respond to large values of  $K_{i,p}$  locally by increasing the size of the neighborhood. Define the set  $\mathcal{N}_i^{(k)}$  of  $k$ th removed neighbors recursively by letting  $\mathcal{N}_i^{(1)} := \mathcal{N}_i$  and

$$(4.27) \quad \mathcal{N}_i^{(\ell+1)} := \cup_{j \in \mathcal{N}_i^{(\ell)}} \mathcal{N}_j.$$

Then interpolation could be allowed from the set  $\mathcal{C}_i := \mathcal{N}_i^{(k)} \cap \mathcal{C}$ , which are the coarse-grid points connected to point  $i$  by a path of length  $k$  in the graph of  $A$ . While this would yield more accurate interpolation, the complexity of  $P^T A P$  would certainly increase.

**5. Numerical Results.** In this section the element interpolation methods are applied to two illustrative examples: a Poisson equation discretized on stretched quadrilaterals and a plane-stress cantilever beam. We compare our numerical results to the *bounds* predicted by our theory, and demonstrate the improved robustness of the new methods over AMG.

The only difference here between AMGe and AMG is that we use the element interpolation method in AMGe to construct the interpolation operators. Thus, the coarse grids are selected in the same way that they are in AMG. We are currently exploring the possibility of using the AMGe measures to determine coarsening. Some comments in this direction are included in the next section.

To conform to the theory, for the AMGe tests the linear systems are scaled so that the diagonal is the identity. That is, we actually solve  $\hat{A}\hat{\mathbf{u}} = \hat{\mathbf{f}}$ , where  $\hat{A} = D^{-1/2}AD^{-1/2}$ ,  $\hat{\mathbf{u}} = D^{1/2}\mathbf{u}$ , and  $\hat{\mathbf{f}} = D^{-1/2}\mathbf{f}$ . Our initial experiments use  $V(0, 1)$  cycles based on damped Jacobi with a relaxation parameter of  $1/2$ . In the examples below  $\|A\|$  is between 2.5 and 3.0 so that  $\frac{1}{\|A\|} \leq s \leq \frac{2}{\|A\|}$ . For AMG, we use the original unscaled matrix  $A$ .

From equation (3.9) in Theorem 3.2 we have a bound on the convergence factor:

$$(5.1) \quad \rho \leq 1 - \frac{4 - \|A\|}{4K},$$

where  $K$  is the bound on either  $M_1$  or  $M_2$ . As we will see, the bound is rather pessimistic. If we replace  $K$  from (4.26) by  $K_p = \max_i K_{i,p}$ , then we have a more realistic but still pessimistic estimate for the convergence factor. These factors are included in the numerical results below.

Three different definitions are considered for interpolation: AMG, local measure 1 (AMGe1), and local measure 2 (AMGe2).

In the multilevel algorithm, we construct “coarse element stiffness matrices”  $A_{c,\alpha}$  as follows:

$$(5.2) \quad A_{c,\alpha} = P^T A_\alpha P.$$

To reduce computational complexity and storage costs, we combine coarse elements that operate on the same points by summing them. That is, we define

$$(5.3) \quad \mathcal{M}_{c,\alpha} := \{j : \boldsymbol{\epsilon}_j^T A_{c,\alpha} \boldsymbol{\epsilon}_j \neq 0\}$$

and, when  $\mathcal{M}_{c,\alpha} = \mathcal{M}_{c,\beta}$ , we combine  $A_{c,\alpha}$  and  $A_{c,\beta}$  to form a single coarse element stiffness matrix.

**5.1. Stretched Quadrilateral.** Consider the stretched quadrilateral problem introduced in Section 2, which consists of a Poisson equation on a rectangular grid discretized with  $n_x \times n_y$  bilinear quadrilateral elements. The fine-grid elements have a 10 : 1 aspect ratio, yielding the stencil in (2.5). The boundary conditions are Dirichlet, which are eliminated from the matrix during discretization.

In all cases, we use the AMG coarsening algorithm with parameter  $\theta = 0.25$ . This produces a semi-coarsened grid for the first coarsening. The resulting interpolation operators

(at interior points) for this grid have the following stencils:

$$(5.4) \quad P_{\text{AMG}} = \begin{bmatrix} 0.084 & 0.332 & 0.084 \\ & * & \\ 0.084 & 0.332 & 0.084 \end{bmatrix}.$$

$$(5.5) \quad P_{\text{AMGe1}} = \begin{bmatrix} 0.007 & 0.486 & 0.007 \\ & * & \\ 0.007 & 0.486 & 0.007 \end{bmatrix}.$$

$$(5.6) \quad P_{\text{AMGe2}} = \begin{bmatrix} 0.003 & 0.494 & 0.003 \\ & * & \\ 0.003 & 0.494 & 0.003 \end{bmatrix}.$$

The stencils at boundaries are similar. Note that interpolation for all of the algorithms involves corner points, but the associated weights for AMGe1 and AMGe2 are much smaller than for AMG. The large element aspect ratio effectively decouples each vertical line of grid points from the other vertical lines. In geometric multigrid, this situation is treated by semi-coarsening, that is, by choosing coarse-grid points along each vertical line. Interpolation is then performed only in the  $y$  direction. The typical interpolation weights used in geometric semi-coarsening do not involve corner points, so smaller weights intuitively make more sense here.

The experimental results are presented in Table 5.1. Two grid sizes,  $64 \times 64$  and  $128 \times 128$ , are used. For each grid we show the convergence factors resulting from application of AMG, AMGe1, and AMGe2. Factors are shown for both two-level and multilevel cases. For the two-level case, we show the bound on the convergence factor corresponding to using (4.26) in Theorem 3.2 for  $M_1$ . This is computed using  $\|A\| = 2.97$  and  $K_1 = 2.68$ . As expected, the bound is pessimistic. We also show the convergence factor (labeled “estimate”) that would result from substituting  $K_1 = \max_i K_{i,1} = 1.34$  and  $K_2 = \max_i K_{i,2} = 2.0$  for (4.26) in Theorem 3.2. This provides a more realistic, but still pessimistic, value for the convergence factor. This behavior is typical of most multigrid theory, where results often exceed theoretical expectations.

The key observation to be made from the data in Table 5.1 is that both AMGe1 and AMGe2 produce substantial improvement over AMG for stretched quadrilaterals. As noted in section 2, there are “fixes” available in AMG (such as iterative weight definition or a judicious choice of the threshold  $\theta$ , or perhaps the geometric/algebraic interpolation methods of [10, 8, 7]) that improve the performance of AMG. For example, if we apply iterative weight definition to either the  $64 \times 64$  or  $128 \times 128$  stretched quadrilateral problem, we obtain convergence factors of 0.28 for both two-level and multilevel cycling. Similarly, if we use AMG with a threshold of  $\theta = 0.5$  we obtain convergence factors of 0.28 (two-level) or 0.29 (multilevel) on both problem sizes. Such techniques, however, tend to be somewhat *ad hoc*, and are not based on theoretical considerations. As such, we cannot determine in advance whether such treatments will be useful for a given problem. By contrast, we expect AMGe1 and AMGe2 to perform well in more general problems involving high aspect ratios, so they should find wide applicability for problems based on unstructured grids having thin domains

Size	Two-Level			Multilevel		
	AMG	AMGe1	AMGe2	AMG	AMGe1	AMGe2
$64 \times 64$	0.82	0.27	0.27	0.84	0.32	0.27
$128 \times 128$	0.82	0.28	0.28	0.84	0.31	0.28
Bound	0.97	0.90	–	–	–	–
Estimate		0.81	0.87	–	–	–

TABLE 5.1

Computed convergence factors, bound predicted by theory, and ‘improvement’ of observed over predicted for stretched quadrilateral problem.

or regions.

**5.2. Plane-Stress Cantilever Beam.** Consider the 2D linear elasticity equations

$$u_{xx} + \frac{1-\nu}{2}u_{yy} + \frac{1+\nu}{2}v_{xy} = f_1,$$

$$v_{yy} + \frac{1-\nu}{2}v_{xx} + \frac{1+\nu}{2}u_{xy} = f_2,$$

where  $u$  and  $v$  are displacements in the  $x$  and  $y$  directions, respectively. We take  $\nu = 0.3$  for the tests. The problem, depicted in Figure 5.1, has free boundaries, except on the left where  $u = v = 0$ . We discretize with bilinear finite elements on a uniform rectangular mesh with spacing  $h$  in both directions (square elements). To make a fair comparison between the different methods, we use the geometric coarsening strategy of doubling the element size in both directions until there is only one element in the  $y$  direction, then doubling the element size in the  $x$  direction only. For the multilevel results, we coarsen until  $h_x = 2h_y$ . The so-called “unknown approach” [17] was used to define interpolation for AMG.

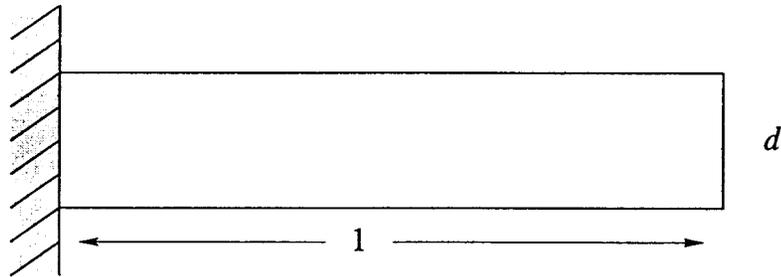


FIG. 5.1. Plane-stress cantilever beam problem.

Experimental results from the plane-stress cantilever beam problem are shown in Table 5.2. Several different thicknesses are used for the beam, ranging from a square cross section,  $d = 1$ , to a very thin beam,  $d = 1/64$ . AMG is ineffective on this problem, both as a two-level and as a multilevel algorithm. Indeed, the theoretical bounds and estimates suggest extremely slow convergence for AMGe1 and AMGe2, and do not indicate that AMG will converge at all. In fact, however, both AMGe1 and AMGe2 achieve substantial improvement, especially for the two-level algorithm, where they greatly exceed the predictions. The bound

d	Two-Level				Multilevel			
	AMG	AMGe1	AMGe2	It.Wt.	AMG	AMGe1	AMGe2	It.Wt.
1	0.97	0.49	0.48	0.37	0.98	0.65	0.85	0.87
1/4	0.97	0.48	0.47	0.80	0.98	0.68	0.90	0.98
1/8	0.98	0.47	0.45	0.87	0.99	0.64	0.87	0.94
1/16	0.97	0.49	0.45	1.00	0.99	0.58	0.77	0.97
1/32	0.97	0.45	0.50	0.97	0.98	0.51	0.56	0.96
1/64	0.94	0.39	0.28	0.98	0.98	0.39	0.28	0.98
Bound	1.00	0.97	–	–	–	–	–	–
Estimate	–	0.87	0.97	–	–	–	–	–

TABLE 5.2

Computed convergence factors, bound predicted by theory, and 'improvement' of observed over predicted for plane-stress,  $h = 1/64$ .

is based on  $\|A\| = 2.50$  and  $K_1 = 12.25$ , while the predictions are based on  $\max_i K_{i,1} = 2.84$  and  $\max_i K_{i,2} = 8.31$ .

Two observations are significant: the two-level performance of AMGe1 and AMGe2 is generally independent of the beam thickness until  $d = h_x$ , where even greater improvement occurs; and the multilevel performance of AMGe1 and AMGe2 improves steadily as the beam becomes thinner. We also include columns indicating the best values obtained with the "fixes," of iterative weight definition and choice of  $\theta$ . The best combination (determined empirically) uses iterative weight definition and  $\theta = 0.5$ . Unlike the previous example, here these methods do not result in improvements similar to those produced by the AMGe1 and AMGe2 methods. We note that the results of [8, 7] apply to a somewhat different elasticity problem than the thin beam considered here, and are not comparable to the experiments we report.

While this paper concentrates on the effect of the new interpolation method, it should be kept in mind that there are other techniques that may be applied to enhance performance of the algorithm. For instance, all multilevel experiments shown here were attained using Jacobi relaxation and a (0,1)  $V$ -cycle. The relaxation method and its parameters can be chosen differently. For example, the multilevel AMGe1 case with  $d = 1/4$  shows a convergence factor of 0.65 in Table 5.2. A Jacobi (1,1)  $V$ -cycle improves this factor to 0.58, while a (1,1)  $F$ -cycle (see [14]) attains a convergence factor of 0.31. Nearly identical results, 0.65 for  $V(0,1)$ , 0.56 for  $V(1,1)$ , and 0.33 for  $F(1,1)$ , are obtained if the Jacobi relaxation is replaced by nodal Gauss-Seidel symmetric  $CF$  relaxation, which sweeps over the  $C$  points followed by the  $F$  points on the downward leg of the  $V$ -cycle, and over the  $F$  points followed by the  $C$  points on the upward leg. Another possibility is the use of a single multigrid  $V(1,1)$  cycle as a preconditioner for a conjugate gradient iteration. Applied to the plane-stress problem using the nodal relaxation described above, this yields convergence factors ranging from 0.16 to 0.26 per CG iteration.

For both sets of experiments, the AMGe interpolation achieves significant improvement over conventional AMG performance. We believe that further improvement is possible by using more sophisticated coarse-grid selection. We observe that local measures  $M_{i,1}$  and

$M_{i,2}$  carry a great deal of information about the nature of the underlying problem and its discretization, and we should be able to exploit this information to determine more effective coarse grids. The following section hints at some possible directions of this ongoing research.

**6. Coarse-Grid Selection.** The focus of this paper is on defining interpolation using local measures  $M_{i,1}$  in (4.11) and  $M_{i,2}$  in (4.12). The numerical tests in the previous section examine a version of AMGe that differs from AMG only in its construction of the interpolation operators. But, we would also like to use these measures to guide the coarse-grid selection process. Defining a practical procedure for doing this, however, is not straightforward and is the subject of current research. The purpose of this section is to present one very simple approach that fits into the current AMG coarsening procedure. The only modification to conventional AMG that this approach makes is to use local measures  $M_{i,1}$  or  $M_{i,2}$  to redefine the AMG notion of strength of dependence.

We proceed by first defining a *strength matrix*,  $S$ , with the same nonzero structure as  $A$ . Denote the  $(i, j)$  entry of  $S$  by  $s_{i,j}$ . We say that *point  $i$  strongly depends on point  $j$*  if

$$(6.1) \quad s_{i,j} \geq \theta \max_{k \neq i} \{s_{i,k}\}, \quad \text{for some fixed } \theta \in (0, 1).$$

The matrix  $S$  may be defined in any number of ways. The simplest approach we consider here is to define

$$(6.2) \quad s_{i,j} = \frac{1}{M_i^{(j)}}$$

for each  $j \in \mathcal{N}_i$ , where  $M_i^{(j)}$  denotes either  $M_{i,1}$  or  $M_{i,2}$  restricted to the case  $\mathcal{C}_i = \{j\}$ . This approach can be interpreted as a way of measuring the “importance” of each potential interpolation point  $j \in \mathcal{N}_i$  individually as if  $j$  were the only point in  $\mathcal{C}_i$ . This approach fits easily into the current AMG coarse-grid selection process.

Because of its focus on individual points, we do not expect this simple scheme to support effective coarsening in all cases. However, we do see improvement over standard AMG in some important cases, as the stretched quadrilateral problem in Section 5.1 illustrates. For this case, the strength matrix based on  $M_{i,1}$  is

$$(6.3) \quad S = \begin{bmatrix} 0.19 & 5.09 & 0.19 \\ 0.20 & * & 0.20 \\ 0.19 & 5.09 & 0.19 \end{bmatrix}.$$

This shows that the direction of strongest coupling is vertical, as it should be.

This approach can break down for more complicated systems. For example, in 2D elasticity on unstructured grids, local stiffness matrix  $A_i$  has three null space eigenvectors. It is generally not possible to fit all three vectors with interpolation from a single  $j \in \mathcal{N}_i$ , and the result is that  $M_i^{(j)} = \infty$  for all  $j \in \mathcal{N}_i$ . This difficulty is due to the limitation of this approach to the examination of only two points at a time, asking the question “How well can the value  $u_i$  be interpolated if  $j$  is the only interpolation point?” A possibly more effective approach is to examine  $M_1$  or  $M_2$  for several (or even all) subsets of  $\mathcal{N}_i$ , with the

aim of determining which *set* of neighbors can best be used to interpolate  $u_i$ . The measures should, of course, be accompanied with cost estimates to control complexity of the coarsening process. We could then assign those points with the best measure as  $F$  points, while assigning the sets that achieve the best measures as  $C$  points.

**7. Conclusions.** For any multigrid method to work, errors that remain after relaxation must be well approximated by the range of interpolation. Since algebraic multigrid does not rely on geometric information, its fundamental challenge is to construct coarse grids and interpolation operators that approximate these errors. The core of this challenge is to determine errors that cannot be effectively reduced by local processing.

Two local measures were introduced here, to quantify how well the coarsening processes determine algebraically smooth error, and used to construct new interpolation operators. Experimental data for two representative test problems confirm that these operators lead to improved AMG convergence rates for these cases.

Current research focuses on using these measures also to assess the ability of coarse-grid points to represent the necessary error components, that is, to determine *which* points are best suited to be on the coarse grid. Combined with the improved interpolation operator, this may lead to very efficient AMG algorithms for a much wider range of problems than is currently available.

## REFERENCES

- [1] C. BALDWIN, P. N. BROWN, R. D. FALGOUT, J. JONES, AND F. GRAZIANI, *Iterative linear solvers in a 2d radiation-hydrodynamics code: methods and performance*. Submitted to Journal of Computational Physics, 1998.
- [2] J. H. BRAMBLE, J. E. PASCIAK, J. WANG, AND J. XU, *Convergence estimates for multigrid algorithms without regularity assumptions*, Math. Comp., 57 (1991), pp. 23–45.
- [3] A. BRANDT, *Algebraic multigrid theory: The symmetric case*, in Preliminary Proceedings for the International Multigrid Conference, Copper Mountain, Colorado, April 1983.
- [4] A. BRANDT, *Algebraic multigrid theory: The symmetric case*, Appl. Math. Comput., 19 (1986), pp. 23–56.
- [5] A. BRANDT, S. F. MCCORMICK, AND J. W. RUGE, *Algebraic multigrid (AMG) for automatic multigrid solutions with application to geodetic computations*. Report, Inst. for Computational Studies, Fort Collins, Colo., October 1982.
- [6] ———, *Algebraic multigrid (AMG) for sparse matrix equations*, in Sparsity and Its Applications, D. J. Evans, ed., Cambridge University Press, Cambridge, 1984.
- [7] Q. CHANG, Y. S. WONG, AND H. FU, *On the algebraic multigrid method*, J. of Comp. Phys., 125 (1996), pp. 279–292.
- [8] Q. CHANG, Y. S. WONG, AND Z. LI, *New interpolation formulas using geometric assumptions in the algebraic multigrid method*, Appl. Math. and Comp., 50 (1992), pp. 223–254.
- [9] A. J. CLEARY, R. D. FALGOUT, V. E. HENSON, J. E. JONES, T. A. MANTEUFFEL, S. F. MCCORMICK, G. N. MIRANDA, AND J. W. RUGE, *Robustness and scalability of algebraic multigrid*. To appear in SIAM Journal on Scientific Computing special issue on the Fifth Copper Mountain Conference on Iterative Methods, 1998.
- [10] W. Z. HUANG, *Convergence of algebraic multigrid methods for symmetric positive definite matrices with weak diagonal dominance*, Appl. Math. and Comp., 46 (1991), pp. 145–164.
- [11] J. MANDEL, M. BREZINA, AND P. VANĚK, *Energy optimization of algebraic multigrid bases*. Submitted.

- [12] S. F. MCCORMICK, *Multigrid methods for variational problems: further results*, SIAM J. Numer. Anal., 21 (1984), pp. 255–263.
- [13] ———, *Multigrid methods for variational problems: general theory for the V-cycle*, SIAM J. Numer. Anal., 22 (1985), pp. 634–643.
- [14] S. F. MCCORMICK AND J. W. RUGE, *Multigrid methods for variational problems*, SIAM J. Numer. Anal., 19 (1982), pp. 924–929.
- [15] J. RUGE, *Element interpolation for algebraic multigrid (AMG)*. Presentation at the 4th Copper Mountain Conference on Multigrid Methods, Copper Mountain, CO, 1989.
- [16] J. W. RUGE AND K. STÜBEN, *Efficient solution of finite difference and finite element equations by algebraic multigrid (AMG)*, in *Multigrid Methods for Integral and Differential Equations*, D. J. Paddon and H. Holstein, eds., The Institute of Mathematics and its Applications Conference Series, Clarendon Press, Oxford, 1985, pp. 169–212.
- [17] ———, *Algebraic multigrid (AMG)*, in *Multigrid Methods*, S. F. McCormick, ed., vol. 3 of *Frontiers in Applied Mathematics*, SIAM, Philadelphia, PA, 1987, pp. 73–130.
- [18] K. STÜBEN, *Algebraic multigrid (AMG): experiences and comparisons*, Appl. Math. Comput., 13 (1983), pp. 419–452.
- [19] K. STÜBEN, U. TROTTEBERG, AND K. WITSCH, *Software development based on multigrid techniques*, in *Proc. IFIP-Conference on PDE Software, Modules, Interfaces and Systems*, B. Enquist and T. Smedsaas, eds., Sweden, 1983, Söderköping.
- [20] P. VANĚK, M. BREZINA, AND J. MANDEL, *Convergence analysis of algebraic multigrid based on smoothed aggregation*. Submitted.
- [21] P. VANĚK, M. BREZINA, AND R. TEZAU, *Two-grid method for linear elasticity on unstructured meshes*, SIAM J. Sci. Comput., (1998). To appear.
- [22] W. L. WAN, *An energy-minimizing interpolation for multigrid methods*, tech. rep., Department of Mathematics, UCLA, April 1997. UCLA CAM Report 97-18.
- [23] W. L. WAN, T. F. CHAN, AND B. SMITH, *An energy-minimizing interpolation for robust multigrid methods*, tech. rep., Department of Mathematics, UCLA, February 1998. UCLA CAM Report 98-6.

# ELEMENT-FREE AMGe: GENERAL ALGORITHMS FOR COMPUTING INTERPOLATION WEIGHTS IN AMG

VAN EMDEN HENSON AND PANAYOT S. VASSILEVSKI

ABSTRACT. We propose a new general algorithm for constructing interpolation weights in algebraic multigrid (AMG). It exploits a proper extension mapping outside a neighborhood about a fine degree of freedom (dof) to be interpolated. The extension mapping provides boundary values (based on the coarse dofs used to perform the interpolation) at the boundary of the neighborhood. The interpolation value is then obtained by matrix dependent harmonic extension of the boundary values into the interior of the neighborhood.

We describe the method, present examples of useful extension operators, provide a two-grid analysis on model problems, and, by way of numerical experiments, demonstrate the successful application of the method to discretized elliptic problems.

## 1. INTRODUCTION

The classical algebraic multigrid (AMG) algorithm [2, 3, 9] was developed for operators represented by symmetric, positive-definite,  $M$ -matrices. While the algorithm works well for many real-world problems [10, 6, 11], there are situations in which it does not perform particularly well. One reason for this is that in some instances the classical definition of interpolation does not adequately interpolate the smooth modes of the error. More specifically, standard AMG interpolation makes certain assumptions about the nature of the smooth error which may not be valid for operators that are not  $M$ -matrices. A more sophisticated characterization of smooth error is required to develop an adequate interpolation formula.

To provide a better characterization of smooth error, a method known as AMGe, for element-based algebraic multigrid, was developed recently [4] for finite-element discretizations. AMGe provides an accurate interpolation formula by using the individual element stiffness matrices to construct a neighborhood matrix for each fine degree of freedom (dof). The sum of the individual stiffness matrices for all the elements containing the point at which the dof is defined, the neighborhood matrix acts as a local “Neumann”-type version of the original operator. According to AMGe theory, once the local matrix is developed and coarse-grid points are chosen, solving

---

*Date:* January 4, 2000–beginning; Today is June 16, 2000.

*1991 Mathematics Subject Classification.* 65F10, 65N20, 65N30.

*Key words and phrases.* algebraic multigrid, interpolation weights, sparse matrices, finite elements, unstructured meshes.

Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, 7000 East Avenue, Mail stop 560, Livermore, CA 94550, email:{vhenson,panayot}@llnl.gov. This work was performed under the auspices of the U. S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract W-7405-Eng-48.

a simple minimization problem yields the optimal interpolation operator for each dof. It is shown in [4] that the method indeed produces superior interpolation and leads to improved convergence rates on several types of problems, including both scalar problems and systems of PDEs, such as elasticity problems.

An obvious drawback to AMGe, naturally, is that it requires that the element stiffness matrices be available. While this is often the case, their storage can be expensive. Further, AMGe requires that coarse level elements be constructed and their individual stiffness matrices be available. Determining the coarse elements is a difficult and laborious task.

In this paper we examine the construction of the interpolation operator in both classical AMG and AMGe, and present them within a common framework. Our purpose is to extend and generalize the classical interpolation, originally motivated for  $M$ -matrices, to develop a rule applicable in more general settings. Accordingly, we then propose a new method for determining the interpolation weights that attempts to capture the benefits of AMGe interpolation without requiring access to the individual element stiffness matrices. This method is applicable to finite difference, finite element, or finite volume discretizations, and we concentrate on the symmetric positive definite case. Essentially, it seeks to determine, for each fine dof, a neighborhood matrix that can be utilized in the same manner that the local assembled stiffness matrix is used in AMGe. We do this by defining a neighborhood for the fine dof and examining the rows of the original matrix that correspond to the points in that neighborhood. A set of exterior dofs is defined, and a mapping developed that extends functions on the neighborhood to the exterior dofs. This essentially imposes a set of boundary conditions on the neighborhood. Here we propose a unified way of building these boundary conditions. One may view them as an extension (extrapolation) of a vector defined on the neighborhood to its immediate exterior. This extension can be performed using constant vectors or any other vectors that may be of interest (such as the rigid body motions in elasticity problems). The extension can be built for each dof in the exterior based on the matrix sparsity pattern.

By incorporating the action of the extension operator into the local connections of the neighborhood, a modified local matrix is created. This matrix is then used in a manner similar to that employed in AMGe, that is, by solving a minimization problem, to create the interpolation operator. The construction of the extension operator and the respective minimization procedure to build the interpolation weights we consider as our main contribution. We give examples of several extension operators and show how they relate to both classical AMG and other, more recently proposed algorithms. A two-grid model analysis of the properties of the resulting interpolation mappings is provided as well. In particular, we prove that they exhibit approximately “harmonic” property as well as “partition of unity” property, desirable in standard two-grid analyses of the AMG methods.

Numerical results are presented demonstrating the method. We include both scalar problems and systems of PDEs in the form of elasticity problems. Finally, we draw some conclusions and comment of the direction that continued research will take.

Some notational convention: to denote a vector we will use boldface, e.g.,  $\mathbf{v}$ ,  $\mathbf{w}$ , .... The  $i$ th component of  $\mathbf{v}$  will be denoted in different contexts as  $\mathbf{v}(i)$ ,  $v(i)$  or  $v_i$ . In the latter two cases  $v$  (i.e. not in boldface) will have a meaning of a "grid" function.

## 2. A FRAMEWORK FOR AMG INTERPOLATION

Assume that the problem  $A\mathbf{x} = \mathbf{f}$  is to be solved, where  $A$  is a sparse, symmetric, positive-definite matrix. AMG is a multigrid method in which no geometric grid information is used (and often isn't available or doesn't even exist). Accordingly, all of the components of a multigrid algorithm, the hierarchy of grids, interpolation and restriction operators, and the coarse-grid versions of the original operator, must be constructed using only the information contained in the entries of  $A$ . For any multigrid algorithm, several basic components are required. In the case of AMG, they can be described as follows:

- A fine grid is required. For AMG, this is generally a set  $D$  comprising the degrees of freedom of the original problem.
- A coarse grid  $D_c$  is necessary. This set of dofs is typically a subset of  $D$ . Generally, a hierarchy of coarse grids  $D \supset D_1 \supset D_2 \supset \dots \supset D_J$  is present.
- An interpolation (prolongation) operator is necessary to map vector functions defined on the coarse grid  $D_c$  to the fine grid  $D$ ,  $P : D_c \rightarrow D$ . Such an operator is required mapping functions on each grid to the next finer grid. Unlike many conventional (geometric) multigrid algorithms, in AMG the interpolation operators are rarely the same for different levels.
- A restriction operator  $R : D \rightarrow D_c$ , mapping fine-grid functions to the coarse grid, is needed. For AMG the restriction is frequently defined by  $R = P^T$ , and we will use that definition here.
- A coarse-grid version of the original operator  $A$  is needed for each coarse level. For AMG the coarse operators are generally defined by the Galerkin relation  $A_c = P^T A P$ .
- A smoothing iteration is used on each level (except the coarsest grid). It is typical to use a point-relaxation method such as Gauß-Seidel or Jacobi relaxation.

There are many ways in which to select the coarse-grid dofs in AMG [9, 11, 5]. Commonly, the coarse set  $D_c$  is a maximally independent subset of  $D$ , but this is not required. We will not discuss the question of coarse-grid selection further, except to note that each fine-grid dof  $i$  is connected to its nearest neighbors (e.g.,  $j$ ) by way of having a nonzero coefficient  $a_{ij}$ , and that the value of a prolonged function at  $i$  is typically an interpolation of the values of its nearest neighbors that are coarse-grid dofs. For the remainder of this paper, we shall simply assume that a coarse grid has been selected and that the coarse neighbors are known for any fine dof.

With this description of the basic components of AMG, we can describe a simple framework for computing the entries of the interpolation operator. Let  $i \in D$  be a fine-grid dof whose value is to be interpolated. We first define a subset  $\Omega(i) \subset D$  to be the *neighborhood* of  $i$ . For now we place no particular restrictions on what dofs can be in  $\Omega(i)$ . For example, the set  $\Omega(i)$  could consist of  $i$  and all of its nearest neighbors, or  $i$  and its nearest coarse neighbors, or  $i$ , its neighbors, and all of their

neighbors. Indeed, within the framework we describe here, the exact character of the interpolation operator will depend largely on what sort of neighborhood is defined. Since the value at  $i$  will be interpolated from coarse points in the neighborhood, it is useful to denote the set of coarse dofs in the neighborhood to be  $\Omega_c(i)$ .

To construct the interpolation for  $i$ , we examine the entries of the operator  $A$  in the following way. We begin, without loss of generality, by permuting the rows and columns of  $A$  and partitioning it so the the first set of rows and columns corresponds to  $i$  and the fine dofs in the neighborhood, that is, to  $\Omega(i) \setminus \Omega_c(i)$ . The next set of rows and columns corresponds to the coarse neighbors  $\Omega_c(i)$ , while the final set of rows and columns corresponds to the rest of the grid  $D \setminus \Omega(i)$ . Hence the partitioning of  $A$ , along with the identity of the rows corresponding to the partitions, appears as

$$A = \left( \begin{array}{ccc} A_{ff} & A_{fc} & * \\ * & * & * \\ * & * & * \end{array} \right) \begin{array}{l} \} \Omega(i) \setminus \Omega_c(i) \\ \} \Omega_c(i) \\ \} D \setminus \Omega(i). \end{array}$$

For our purposes we are only concerned with two blocks of the partitioned matrix. The block  $A_{ff}$  gives the connections among  $i$  and the fine-grid neighbors while the block  $A_{fc}$  links  $i$  and the fine neighbors to the coarse neighbors.

### 3. INTERPOLATION IN AMG

For classical AMG [9], the interpolation is computed in the following fashion. The neighborhood  $\Omega(i)$  is defined to be the dof  $i$  and all dofs connected to it (all  $j$  for which  $a_{ij} \neq 0$ ). The fine-to-fine block,  $A_{ff}$  is then replaced with a modified version,  $\widehat{A}_{ff}$ . This block is modified in two ways. First, we modify the row corresponding to the dof  $i$  (which we will hereafter refer to as the  $i$ th row, regardless of the actual numerical ordering) by adding to the diagonal element  $a_{ii}$  any off-diagonal entries  $a_{ij}$  for dofs  $j$  that are weakly connected to  $i$ , and then setting  $a_{ij} = 0$ . By weakly connected we mean that the magnitude of  $a_{ij}$  is smaller than some pre-defined threshold. A common choice is that if the magnitude of  $a_{ij}$  is less than  $\theta$  times the largest magnitude of all off-diagonal entries in the  $i$ th row then  $j$  is considered to be weakly connected to  $i$ . The second modification to  $A_{ff}$  is that for each row  $j$  corresponding to a dof strongly connected to  $i$ , the diagonal element  $a_{jj}$  is replaced by

$$a_{jj} \leftarrow - \sum_{k \in \Omega(i)} a_{jk}$$

after which the off diagonal entries of the  $j$ th row are set to zero. Once the modified block  $\widehat{A}_{ff}$  is computed, the entries of the  $i$ th row of the interpolation matrix  $P$  are determined by taking the entries of the  $i$ th row of the matrix

$$- \left( \widehat{A}_{ff}^{-1} A_{fc} \right).$$

### 4. INTERPOLATION IN AMGE

For AMGe a similar description of the interpolation is easily given. In this setting, the neighborhood  $\Omega(i)$  is defined naturally as the union of all finite elements having  $i$  as a vertex (Figure 1). In the figure, the set  $\Omega(i)$  consists of all vertices in the shaded

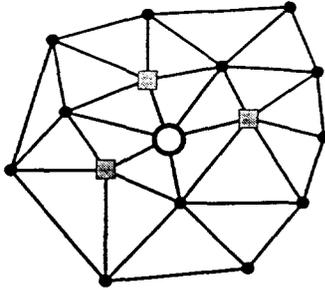


FIGURE 1. *The neighborhood of the fine dof  $i$  (large open circle).*

region, including  $i$  (the open circle in the center). The shaded region consists of the six triangular finite elements having  $i$  as a vertex. Members of  $\Omega_c(i)$  are indicated by the square vertices. Since AMGe gives us access to the individual element stiffness matrices, we may create a neighborhood matrix  $A_{\Omega(i)}$  simply by summing together all the individual element stiffness matrices of the elements in the neighborhood. In AMGe the interpolation operator for the dof  $i$  is determined by solving a constrained min-max problem, that is, by finding interpolation coefficients that minimize a certain measure from finite element theory. The solution to the min-max problem can be computed in several ways, one of which fits into the framework we are developing here. We partition the neighborhood matrix into the rows and columns associated with the fine dofs in the neighborhood and the rows and columns associated with the coarse dofs, as

$$A_{\Omega(i)} = \begin{pmatrix} A_{ff} & A_{fc} \\ * & * \end{pmatrix} \begin{array}{l} \} \Omega(i) \setminus \Omega_c(i) \\ \} \Omega_c(i). \end{array}$$

Again, our only interest is in the rows of the neighborhood matrix corresponding to the fine dofs, including  $i$ . With this partitioning, it turns out that one way to solve the min-max problem is to take, as the coefficients for the interpolation operator for  $i$ , the entries of the  $i$ th row of the matrix

$$- (A_{ff}^{-1} A_{fc}).$$

It is useful to note that, unlike the classical AMG case, there is no need to modify the matrix  $A_{ff}$  prior to computing the interpolation coefficients. Essentially, this is because the element stiffness matrices automatically carry with them the correct handling of strong and weak connections, so that the neighborhood matrix already has the correct relationships built into it.

For many problems the AMGe method produces a superior interpolation, and results in good convergence rates [4]. In the remainder of this paper our goal is to accomplish a similar superior interpolation without the knowledge (and hence, expense) of the individual stiffness matrices.

## 5. INTERPOLATION FOR ELEMENT-FREE AMGE

The process we propose for building the interpolation operator is very similar to the processes described for AMG and AMGe. Once again, we will proceed by defining

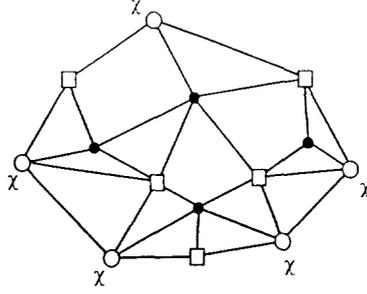


FIGURE 2. The extended neighborhood  $\bar{\Omega}(\psi)$ , including the fine dofs to be interpolated (solid circles), the coarse interpolatory set  $\Omega_c(\psi)$  (squares), and the extension dofs (open circles marked  $\chi$ ).

a neighborhood of the fine dofs and an associated neighborhood matrix. Let  $\psi$  be a set of fine dofs whose values we wish to interpolate. We define  $\Omega(\psi)$  to be the neighborhood of  $\psi$ , which includes the coarse dofs that will be used to interpolate the dofs in  $\psi$ . The set of coarse dofs in the neighborhood we denote  $\Omega_c(\psi)$ .

Now, however, we define a third set of dofs

$$\Omega_{\chi}(\psi) = \{j \notin \Omega(\psi) \mid a_{ij} \neq 0 \text{ for some } i \in \Omega(\psi) \setminus \Omega_c(\psi)\}.$$

That is,  $\Omega(\psi)$  can be viewed as the interior of the set  $\bar{\Omega}(\psi) \equiv \Omega(\psi) \cup \Omega_{\chi}(\psi)$ . Figure 2 gives an example of such a neighborhood.

We begin the construction of a neighborhood matrix by examining the rows of the matrix  $A$  that correspond to the fine dofs in  $\psi$ ; that is, we will be concerned with the following partitioning of  $A$ :

$$A = \begin{pmatrix} A_{ff} & A_{fc} & A_{f\chi} & 0 \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{pmatrix} \begin{array}{l} \} \Omega(\psi) \setminus \Omega_c(\psi) \\ \} \Omega_c(\psi) \\ \} \Omega_{\chi}(\psi) \\ \} \text{everything else on grid.} \end{array}$$

**5.1. Local (neighborhood) quadratic form.** Our task next is to define a matrix associated with  $\psi$  that yields a local version of the operator  $A$ , performing the same function as does the neighborhood matrix in AMG. To do this we first build an extension mapping (matrix)  $E(\psi)$  that maps a vector defined on  $\Omega(\psi)$  to  $\bar{\Omega}(\psi)$

$$E(\psi) : \begin{pmatrix} \mathbf{v}_f \\ \mathbf{v}_c \end{pmatrix} \rightarrow \begin{pmatrix} \mathbf{v}_f \\ \mathbf{v}_c \\ \mathbf{v}_{\chi} \end{pmatrix}$$

using the relation

$$\mathbf{v}_{\chi} = E_{\chi f}(\psi)\mathbf{v}_f + E_{\chi c}(\psi)\mathbf{v}_c.$$

That is, the extension operator looks like

$$E = \begin{pmatrix} I & 0 \\ 0 & I \\ E_{\chi f}(\psi) & E_{\chi c}(\psi) \end{pmatrix}.$$

For now we will not be specific about the exact nature of the extension operator. Rather, we will describe how it may be used to develop an interpolation formula, after which we shall discuss desirable properties of the operator.

We construct a neighborhood matrix from the first block of rows of the partitioned matrix

$$\left( \widehat{A}_{ff}, \widehat{A}_{fc} \right) = (A_{ff}, A_{fc}, A_{f\mathcal{X}}) \begin{pmatrix} I & 0 \\ 0 & I \\ E_{\mathcal{X}f}(\psi) & E_{\mathcal{X}c}(\psi) \end{pmatrix}$$

so that

$$\widehat{A}_{ff} = A_{ff} + A_{f\mathcal{X}}E_{\mathcal{X}f}(\psi) \quad \text{and} \quad \widehat{A}_{fc} = A_{fc} + A_{f\mathcal{X}}E_{\mathcal{X}c}(\psi).$$

For any vector  $\begin{bmatrix} \mathbf{v}_f \\ \mathbf{v}_c \end{bmatrix}$ , consider its extension  $\mathbf{v} = \begin{bmatrix} \mathbf{v}_f \\ \mathbf{v}_c \\ \mathbf{v}_{\mathcal{X}} \end{bmatrix}$ , where  $\mathbf{v}_{\mathcal{X}}$  is given by  $\mathbf{v}_{\mathcal{X}} = E_{\mathcal{X}f}(\psi)\mathbf{v}_f + E_{\mathcal{X}c}(\psi)\mathbf{v}_c$ . Let

$$\widehat{\mathbf{v}} = \begin{bmatrix} -A_{ff}^{-1}(A_{fc}\mathbf{v}_c + A_{f\mathcal{X}}\mathbf{v}_{\mathcal{X}}) \\ \mathbf{v}_c \\ \mathbf{v}_{\mathcal{X}} \end{bmatrix}$$

be the so-called harmonic extension of  $\mathbf{v}|_{\Omega_c(\psi) \cup \Omega_{\mathcal{X}}(\psi)}$  into  $\Omega(\psi) \setminus \Omega_c(\psi)$ . That is, one extends  $\mathbf{v}$ , restricted to the ‘‘boundary’’  $\Omega_c(\psi) \cup \Omega_{\mathcal{X}}(\psi)$ , into the ‘‘interior’’  $\Omega(\psi) \setminus \Omega_c(\psi)$ .

We use the  $\mathbf{v}_f$  that minimizes the difference  $\mathbf{v} - \widehat{\mathbf{v}}$  in energy norm in the interpolation procedure. Since

$$\mathbf{v} - \widehat{\mathbf{v}} = \begin{bmatrix} \mathbf{v}_f - (\widehat{\mathbf{v}})_f \\ 0 \\ 0 \end{bmatrix},$$

its energy norm is computable and equals

$$\begin{aligned} \|\mathbf{v}_f - (\widehat{\mathbf{v}})_f\|_A &= (\mathbf{v}_f - (\widehat{\mathbf{v}})_f)^T A_{ff} (\mathbf{v}_f - (\widehat{\mathbf{v}})_f) \\ &= (\mathbf{v}_f + A_{ff}^{-1}(A_{fc}\mathbf{v}_c + A_{f\mathcal{X}}\mathbf{v}_{\mathcal{X}}))^T A_{ff} (\mathbf{v}_f + A_{ff}^{-1}(A_{fc}\mathbf{v}_c + A_{f\mathcal{X}}\mathbf{v}_{\mathcal{X}})). \end{aligned}$$

Since  $A_{ff}$  is positive definite, this implies that if we solve the equation

$$\begin{aligned} 0 &= A_{ff}\mathbf{v}_f + (A_{fc}\mathbf{v}_c + A_{f\mathcal{X}}\mathbf{v}_{\mathcal{X}}) \\ &= (A_{ff} + A_{f\mathcal{X}}E_{\mathcal{X}f})\mathbf{v}_f + (A_{fc} + A_{f\mathcal{X}}E_{\mathcal{X}c})\mathbf{v}_c \\ &= \widehat{A}_{ff}\mathbf{v}_f + \widehat{A}_{fc}\mathbf{v}_c, \end{aligned}$$

the minimization of  $\|\mathbf{v}_f - (\widehat{\mathbf{v}})_f\|_A$  is attained with zero minimum by  $\mathbf{v}_f = -\widehat{A}_{ff}^{-1}\widehat{A}_{fc}\mathbf{v}_c$ .

We can actually show (see Remark 7.1 and Lemma 7.1) that in the model finite element case considered in Section 7 the minimization procedure is equivalent to a quadratic functional minimization involving Neumann assembled matrices, as in the AMGe method (cf., [4]).

It is natural to ask whether  $\widehat{A}_{ff}$  is invertible. If  $E_{\mathcal{X}f} = 0$ , there is no difficulty, since then  $\widehat{A}_{ff} = A_{ff}$ . In general, if  $E_{\mathcal{X}f}$  is sufficiently small in norm  $A_{ff} + A_{f\mathcal{X}}E_{\mathcal{X}f}$  will be invertible.

## 6. EXAMPLES OF EXTENSION OPERATORS

We describe here three extension operators  $E$  that can be used to construct the interpolation operator in the element-free approach. These are by no means all the useful extensions that we could concoct; they form, however, a simple set of examples that will allow us to demonstrate the efficacy of the method and its underlying philosophy.

The first we call the  $L_2$ -extension because it is a simple averaging method. Given  $\mathbf{v}$  defined on  $\Omega(i)$ , we wish to extend it to  $\mathbf{v}_{\mathcal{X}}$ , defined on  $\Omega_{\mathcal{X}}(\psi)$ . Suppose that  $i_{\mathcal{X}}$  is an exterior dof, that is, a point from  $\Omega_{\mathcal{X}}(\psi)$  whose value we wish to determine from the values of the dofs in  $\Omega(i)$ . Let  $S = \{j \in \Omega(i) : a_{i_{\mathcal{X}},j} \neq 0\}$ ; that is,  $S$  comprises those dofs in  $\Omega(i)$  to which the point  $i_{\mathcal{X}}$  is connected. It seems natural to consider using a simple average over these dofs as the extension at  $i_{\mathcal{X}}$ . Thus, the extension formula, for the dof  $i_{\mathcal{X}}$ , is given by

$$\mathbf{v}_{\mathcal{X}}(i_{\mathcal{X}}) = \frac{1}{\sum_{j \in S} 1} \sum_{j \in S} \mathbf{v}(j).$$

A somewhat more sophisticated extension we call the  $A$ -extension because it is a simple operator-induced method. The  $A$ -extension operator for the dof  $i_{\mathcal{X}}$  is given by the formula

$$\mathbf{v}_{\mathcal{X}}(i_{\mathcal{X}}) = \frac{1}{\sum_{j \in S} |a_{i_{\mathcal{X}},j}|} \sum_{j \in S} (|a_{i_{\mathcal{X}},j}| \mathbf{v}(j)).$$

It may be seen that in this case the extension to the exterior is a simple weighted average of the values of the neighborhood dofs to which the exterior point is connected. The weights in the average are given by the absolute values of the matrix coefficients.

The two methods just described share the property that they are computed point-by-point. That is, the extension formulas for the dofs in  $\Omega_{\mathcal{X}}(\psi)$  are determined independently. A second feature shared by the methods is that if the neighborhood vector  $\mathbf{v}$  is constant, then the extended values are also constant, and have the same value as the entries of the neighborhood vector. This feature is clearly desirable for many elliptic PDEs, where the constant vector is in the null space or near-null space of the operator  $A$ .

The third example we describe is based on the minimization of a quadratic functional. Again, let  $\mathbf{v}$  be a vector defined on  $\Omega(i)$  that we wish to extend to  $\Omega_{\mathcal{X}}(\psi)$ . We construct the extension to be that operator which produces  $\mathbf{v}_{\mathcal{X}}$  that minimizes the functional  $Q(\mathbf{v}_{\mathcal{X}})$ , where

$$Q(\mathbf{v}_{\mathcal{X}}) = \sum_{\substack{i_{\mathcal{X}} \in \Omega_{\mathcal{X}}(\psi) \\ j \in \Omega(i)}} |a_{i_{\mathcal{X}},j}| (v_{i_{\mathcal{X}}} - v_j)^2.$$

It is evident that, like the previous extension operators, if  $\mathbf{v}$  is constant on  $\Omega(i)$  then the dofs in  $\Omega_{\mathcal{X}}(i)$  will also have the same constant value. Unlike the previous extension operators, which are determined one dof at a time, this is a “simultaneous” extension, computing formulas for extending to all of the exterior dofs together. As such, it is necessarily more expensive to compute. We also note that this extension, and the interpolation it generates, is equivalent to the method recently proposed in [1].

A final example is given by minimizing the following “cut-off” quadratic functional:

$$(\theta \mathbf{v})^T A_{\overline{\Omega}(\psi)} (\theta \mathbf{v}) \mapsto \min$$

subject to  $\mathbf{v}_f, \mathbf{v}_c$  fixed. Here

$$\theta = \begin{bmatrix} I & 0 \\ 0 & \theta_{\mathcal{X}} \end{bmatrix} \begin{array}{l} \} \Omega(\psi) \\ \} \Omega_{\mathcal{X}}(\psi) \end{array}$$

is a diagonal matrix. A good choice is, a diagonal matrix formed from the vector

$$\theta_{\mathcal{X}} = -(A_{\mathcal{X}\mathcal{X}})^{-1} [A_{\mathcal{X}f}, A_{\mathcal{X}c}] \begin{bmatrix} (1)_f \\ (1)_c \end{bmatrix}.$$

Here we used the blocks of  $A$  corresponding to its  $\Omega_{\mathcal{X}}(\psi)$  rows.

It is easily seen that the extension mapping is actually defined as

$$\begin{aligned} \mathbf{v}_{\mathcal{X}} &= E_{\mathcal{X}c} \mathbf{v}_c + E_{\mathcal{X}f} \mathbf{v}_f \\ &= -\theta_{\mathcal{X}}^{-1} (A_{\mathcal{X}\mathcal{X}})^{-1} [A_{\mathcal{X}f}, A_{\mathcal{X}c}] \begin{bmatrix} \mathbf{v}_f \\ \mathbf{v}_c \end{bmatrix}. \end{aligned}$$

Note that this extension mapping is also a simultaneous extension operator and an averaging one; i.e., if  $\mathbf{v}_c = (1)_c$  and  $\mathbf{v}_f = (1)_f$ , then  $\mathbf{v}_{\mathcal{X}} = (1)_{\mathcal{X}}$ .

**6.1. Classical AMG as an extension method.** The interpolation method of the classical AMG algorithm popularized by Ruge and Stüben [9] may be viewed as an extension method. Here the neighborhood is just the dof to be interpolated together with the dofs that will be used to compute the interpolated value. That is,  $\Omega(i) = \{i\} \cup \Omega_c(i)$ . The extended neighborhood then includes all fine dofs that are connected to  $i$ ,

$$\Omega_{\mathcal{X}}(\psi) = \{j \notin \Omega(i) : a_{ij} \neq 0\}.$$

An  $A$ -extension is defined in the following manner. For each  $i_{\mathcal{X}} \in \Omega_{\mathcal{X}}(\psi)$ , set  $v_{i_{\mathcal{X}}} = v_i$  if  $i_{\mathcal{X}}$  is weakly connected to  $i$  (Recall that in classical AMG, as developed for  $M$ -matrices, the dof  $i$  is said to be strongly connected to the dof  $j$  if

$$-a_{ij} > \theta \max_{k \neq i} (-a_{ik})$$

where  $\theta$  is a user-specified parameter, and weakly-connected otherwise). If  $i_{\mathcal{X}}$  is strongly connected to  $i$  the extension is defined by

$$v_{i_{\mathcal{X}}} = \frac{1}{\sum_{j \in \Omega_c(\psi)} a_{i_{\mathcal{X}}j}} \sum_{j \in \Omega_c(\psi)} a_{i_{\mathcal{X}}j} v_j.$$

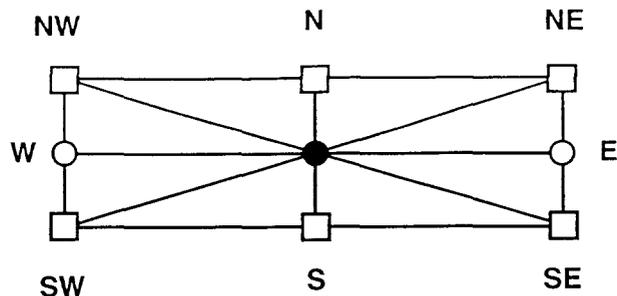


FIGURE 3. *The neighborhood of the fine dof  $i$  (large solid circle) for the stretched quadrilateral element problem. The problem is semicoarsened; squares denote the coarse neighbors  $\Omega_c(i)$  while the open circles are the exterior points  $\Omega_{\mathcal{X}}(\psi)$ .*

A simple example should suffice to illustrate these extension methods. Suppose the problem  $-U_{xx} - U_{yy} = f(x, y)$  is discretized using a regular Cartesian grid of points making up the vertices of quadrilateral elements. Suppose further that the elements had dimension  $h_x \times h_y$  where  $h_x \gg h_y$ . As  $h_y/h_x \rightarrow 0$  the operator stencil tends toward

$$\begin{bmatrix} -1 & -4 & -1 \\ 2 & 8 & 2 \\ -1 & -4 & -1 \end{bmatrix}.$$

Since there is effectively no coupling between a given point and its neighbors to the east or west, the appropriate choice is to semicoarsen, selecting every other line of points with constant  $y$ -coordinate to be coarse points. Using the same logic, the natural interpolation is to have each fine dof interpolated only using the values to the north and south of it, each with equal weighting of  $1/2$ . Consider the interpolation of one point,  $i$ , shown in the center of its neighborhood in Figure 3. For either the  $L_2$ - or  $A$ -extensions, we might select  $\Omega(i) = i \cup \Omega_c(i)$  where, in this instance,  $\Omega_c(i) = \{N, S, SW, NW, SE, NE\}$ . Then  $\Omega_{\mathcal{X}}(\psi) = \{W, E\}$ . We see then that  $A_{ff} = [8]$ ,  $A_{fc} = [-4 \ -4 \ -1 \ -1 \ -1 \ -1]$ , and  $A_{f\mathcal{X}} = [2 \ 2]$ . For the  $A$ -extension it is easy to compute the extension operators

$$E_{\mathcal{X}c} = \frac{1}{12} \begin{pmatrix} 1 & 1 & 4 & 4 \\ 1 & 1 & & 4 & 4 \end{pmatrix} \quad \text{and} \quad E_{\mathcal{X}f} = \frac{1}{12} \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

from which

$$\hat{A}_{ff} = \left( \frac{104}{12} \right) \quad \text{and} \quad \hat{A}_{fc} = \frac{1}{3} \begin{pmatrix} -11 & -11 & -1 & -1 & -1 & -1 \end{pmatrix}$$

which yields a interpolation operator

$$P_A = \frac{1}{26} \begin{pmatrix} 11 & 11 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

We see that the values to the north and south are used in the interpolation with weights  $11/26 \approx 0.423$  and that the four points diagonally adjacent to  $i$  all are weighted  $1/26 \approx 0.038$ . The ideal weights, of course, are 0.5 and 0, respectively, so

the interpolation weights computed by the  $A$ -extension method, while quite good, are not perfect.

A similar calculation for the weights using the  $L_2$ -extension yields the interpolation operator

$$P_{L_2} = \frac{1}{44} \begin{pmatrix} 16 & 16 & 3 & 3 & 3 & 3 \end{pmatrix}.$$

Here the dofs to the north and south are weighted  $16/44 \approx 0.364$  while the diagonally adjacent dofs are weighted by  $3/44 \approx 0.068$ . For this problem, then, the  $A$ -extension is significantly better than the  $L_2$ -extension.

By contrast, it is a straightforward calculation to show that classical AMG produces the interpolation operator

$$P_{AMG} = \frac{1}{12} \begin{pmatrix} 4 & 4 & 1 & 1 & 1 & 1 \end{pmatrix},$$

where the north and south dofs are weighted by  $4/12 \approx 0.333$  and the diagonally adjacent dofs are weighted by  $1/12 \approx 0.083$ ; these weights are farther from the ideal than the weights produced by either the  $A$ - or  $L_2$ -extension.

Finally consider the extension operator based on minimizing the ‘‘cut-off’’ quadratic functional. The additional matrix blocks involved read:

$$\begin{aligned} A_{\mathcal{X}\mathcal{X}} &= \begin{bmatrix} 8 & 0 \\ 0 & 8 \end{bmatrix}, \\ A_{\mathcal{X}f} &= \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \\ A_{\mathcal{X}c} &= \begin{bmatrix} -1 & -1 & -4 & -4 & 0 & 0 \\ -1 & -1 & 0 & 0 & -4 & -4 \end{bmatrix}. \end{aligned}$$

The vector  $\underline{\theta}_{\mathcal{X}} = -A_{\mathcal{X}\mathcal{X}}^{-1}[A_{\mathcal{X}f} \ A_{\mathcal{X}c}] \begin{bmatrix} (1)_f \\ (1)_c \end{bmatrix} = (1)_{\mathcal{X}}$ . This is seen as follows

$$A_{\mathcal{X}f}(1)_f = 2(1)_{\mathcal{X}}, \quad A_{\mathcal{X}c}(1)_c = -10(1)_{\mathcal{X}},$$

and hence

$$A_{\mathcal{X}f}(1)_f + A_{\mathcal{X}c}(1)_c = -8(1)_{\mathcal{X}},$$

which implies

$$\underline{\theta}_{\mathcal{X}} = -A_{\mathcal{X}\mathcal{X}}^{-1}(A_{\mathcal{X}f}(1)_f + A_{\mathcal{X}c}(1)_c) = -\frac{1}{8}[-8(1)_{\mathcal{X}}] = (1)_{\mathcal{X}}.$$

That is, the diagonal matrix  $\theta$  is the identity and hence the extension matrices then read:

$$\begin{aligned} E_{\mathcal{X}f} &= -A_{\mathcal{X}\mathcal{X}}^{-1}A_{\mathcal{X}f} = -\frac{1}{4} \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \\ E_{\mathcal{X}c} &= -A_{\mathcal{X}\mathcal{X}}^{-1}A_{\mathcal{X}c} = \frac{1}{8} \begin{bmatrix} 1 & 1 & 4 & 4 & 0 & 0 \\ 1 & 1 & 0 & 0 & 4 & 4 \end{bmatrix}. \end{aligned}$$

The modified matrices  $\widehat{A}_{ff}$  and  $\widehat{A}_{fc}$  take the form:

$$\begin{aligned}\widehat{A}_{ff} &= A_{ff} + A_{f\mathcal{X}}E_{\mathcal{X}f} = 8 - [2, 2] \frac{1}{4} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ &= 7, \\ \widehat{A}_{fc} &= A_{fc} + A_{f\mathcal{X}}E_{\mathcal{X}c} = [-4, -4, -1, -1, -1, -1] + [2, 2] \frac{1}{8} \begin{bmatrix} 1 & 1 & 4 & 4 & 0 & 0 \\ 1 & 1 & 0 & 0 & 4 & 4 \end{bmatrix} \\ &= [-4, -4, -1, -1, -1, -1] + [\frac{1}{2}, \frac{1}{2}, 1, 1, 1, 1] \\ &= [-\frac{7}{2}, -\frac{7}{2}, 0, 0, 0, 0].\end{aligned}$$

That is, the interpolation coefficients are the “perfect” ones:

$$[\frac{1}{2}, \frac{1}{2}, 0, 0, 0, 0].$$

## 7. TWO-GRID ANALYSIS FOR A MODEL FINITE ELEMENT PROBLEM

Before providing numerical results, we present an analysis of the quality of the “element-free AMGe” interpolation. That is, we prove an “approximate” harmonic property of the interpolation mapping and show that it provides a partition of unity. Specifically, we assume that the problem is a standard finite element discretization of a second order elliptic problem

$$a(u, v) \equiv \int a(x) \nabla u \cdot \nabla v \, dx = (f, v) \quad v \in V,$$

where  $V$  is a finite element space of piecewise linear functions over quasiuniform triangular elements that cover a given 2-d polygonal domain. For simplicity, we assume that homogeneous Neumann boundary conditions are imposed and that  $(f, 1) = 0$  (to insure solvability).

Let us denote, for any element  $e$ ,

$$(7.1) \quad \varrho(e) = \sup_{x \in e} \max_{\underline{\xi} \in \mathbb{R}^2} \frac{\underline{\xi}^T a(x) \underline{\xi}}{\underline{\xi}^T \underline{\xi}}.$$

In the following, we assume (only for simplicity) that the neighborhood  $\overline{\Omega}(i) \equiv \Omega(i) \cup \Omega_{\mathcal{X}}(i)$  for any fine dof  $i$  is formed by union of triangles that share dof  $i$  as a common vertex. Thus we will use  $i$  instead of  $\psi$  denoting the neighborhoods  $(\Omega(i), \Omega_{\mathcal{X}}(i),$  and  $\Omega_c(i))$  and the extension mappings. In particular, we denote  $E_i = [E_{\mathcal{X}f}, E_{\mathcal{X}c}]$  where for brevity  $E_{\mathcal{X}f} = E_{\mathcal{X}f}(i)$  and  $E_{\mathcal{X}c} = E_{\mathcal{X}c}(i)$ . A closer look at the analysis to follow, however, shows that it applies as well to more general (i.e., larger) neighborhoods.

In what follows, for any subdomain (union of triangles)  $G$ , we let  $a_G(\cdot, \cdot)$  denote the bilinear form  $a$  restricted to  $G$ . The corresponding subdomain matrix (assembled from the individual element matrices  $A_e$ ) will be denoted by  $A_G^N$ . We omit the superscript  $N$  when there is no confusion between  $A_G^N$  and  $A_G$ , the submatrix of the original matrix  $A$  (corresponding to  $G$ ). Note that in the latter case  $A_G$  corresponds to a matrix with homogeneous Dirichlet boundary conditions imposed on  $\partial(G \cup \{\text{elements neighboring } G\})$ .

For this discussion we assume that  $E_i$ , the local extension mapping used to build the interpolation coefficients, is based on averaging, although no specific rule is assumed. We do, however, assume that  $E = E(i)$  has the particular form

$$\begin{bmatrix} I & 0 \\ 0 & I \\ 0 & E_{\mathcal{X}c} \end{bmatrix} \begin{matrix} \} \Omega(i) \\ \} \Omega_c(i) \\ \} \Omega_{\mathcal{X}}(i) \end{matrix}$$

That is,  $E_{\mathcal{X}f} = 0$  and  $E_i = [0, E_{\mathcal{X}c}]$ .

**Remark 7.1.** *The general case of  $E_i = [E_{\mathcal{X}f}, E_{\mathcal{X}c}]$  can be reduced to the particular case above by using the modified extension mapping  $\widehat{E}_i = [0, \widehat{E}_{\mathcal{X}c}]$  where*

$$\widehat{E}_{\mathcal{X}c} = E_{\mathcal{X}f} \left( -\widehat{A}_{ff}^{-1} \widehat{A}_{fc} \right) + E_{\mathcal{X}c}.$$

To see this, recall that  $\widehat{A}_{ff} = A_{ff} + A_{f\mathcal{X}}E_{\mathcal{X}f}$  and  $\widehat{A}_{fc} = A_{fc} + A_{f\mathcal{X}}E_{\mathcal{X}c}$ , and note that the modified extension mapping extends a constant vector defined on  $\Omega_c(i)$  to be the same constant on  $\Omega_{\mathcal{X}}(i)$ , that is,

$$\begin{aligned} \widehat{E}_{\mathcal{X}c}(1)_c &= -E_{\mathcal{X}f} \widehat{A}_{ff}^{-1} \widehat{A}_{fc}(1)_c + E_{\mathcal{X}c}(1)_c \\ &= E_{\mathcal{X}f}(1)_f + E_{\mathcal{X}c}(1)_c \\ &= (1)_{\mathcal{X}}. \end{aligned}$$

Here we have used the fact that since (for the second order elliptic problem)  $A_{ff}(1)_f + A_{f\mathcal{X}}(1)_{\mathcal{X}} + A_{fc}(1)_c = 0$  then  $A_{ff}(1)_f + A_{f\mathcal{X}}(E_{\mathcal{X}f}(1)_f + E_{\mathcal{X}c}(1)_c) + A_{fc}(1)_c = 0$ . That is,  $\widehat{A}_{ff}(1)_f + \widehat{A}_{fc}(1)_c = 0$ , implying that  $(1)_f = -\widehat{A}_{ff}^{-1} \widehat{A}_{fc}(1)_c$ .

We still must show that the modified extension mapping  $\widehat{E}_i$  leads to the same interpolation as does  $E_i$ , i.e., that

$$-A_{ff}^{-1} \left( A_{f\mathcal{X}} \widehat{E}_{\mathcal{X}c} + A_{fc} \right) = -\widehat{A}_{ff}^{-1} \widehat{A}_{fc}.$$

For this we observe that

$$\begin{aligned} -A_{ff}^{-1} (A_{f\mathcal{X}} \widehat{E}_{\mathcal{X}c} + A_{fc}) &= -A_{ff}^{-1} \left[ A_{f\mathcal{X}} \left( E_{\mathcal{X}f} (-\widehat{A}_{ff}^{-1} \widehat{A}_{fc}) + E_{\mathcal{X}c} \right) + A_{fc} \right] \\ &= -A_{ff}^{-1} \left[ A_{f\mathcal{X}} E_{\mathcal{X}c} + A_{fc} - A_{f\mathcal{X}} E_{\mathcal{X}f} \widehat{A}_{ff}^{-1} \widehat{A}_{fc} \right] \\ &= -A_{ff}^{-1} \left[ \widehat{A}_{fc} - A_{f\mathcal{X}} E_{\mathcal{X}f} \widehat{A}_{ff}^{-1} \widehat{A}_{fc} \right] \\ &= -A_{ff}^{-1} \left[ \widehat{A}_{ff} - A_{f\mathcal{X}} E_{\mathcal{X}c} \right] \widehat{A}_{ff}^{-1} \widehat{A}_{fc} \\ &= -A_{ff}^{-1} (A_{ff}) \widehat{A}_{ff}^{-1} \widehat{A}_{fc} \\ &= -\widehat{A}_{ff}^{-1} \widehat{A}_{fc}. \end{aligned}$$

◇

Consider the minimization problem

$$(7.2) \quad \text{find } v_f \text{ such that } \begin{bmatrix} v_f \\ v_c \\ E_i v \end{bmatrix}^T A_{\overline{\Omega}(i)}^N \begin{bmatrix} v_f \\ v_c \\ E_i v \end{bmatrix} = \inf_{\substack{w: w_c = v_c \\ w_{\mathcal{X}} = E_i w}} a_{\overline{\Omega}(i)}(w, w).$$

Thus we seek  $v_f$ , the value of  $w$  on  $\Omega(i) \setminus \Omega_c(i)$ , which minimizes the quadratic form  $a_{\bar{\Omega}(i)}(w, w)$  when the values of  $w$  are fixed at the coarse points and are “slave” at the exterior points ( $\Omega_{\mathcal{X}}(i)$ ): that is, they are extrapolated from the interior  $\Omega(i)$  and the coarse points  $\Omega_c(i)$  by  $E_i w$ .

**Lemma 7.1.** *The solution to the minimization problem (7.2) produces the same interpolation coefficients element-free AMGe, namely, those given by  $-A_{ff}^{-1}(A_{f\mathcal{X}}E_{\mathcal{X}c} + A_{fc})$ . That is, the minimizer is given by  $w_f = v_f \equiv -A_{ff}^{-1}(A_{f\mathcal{X}}E_{\mathcal{X}c} + A_{fc})v_c$ .*

*Proof.* Consider the Neumann matrix

$$A_{\bar{\Omega}(i)}^N = \begin{bmatrix} A_{ff} & A_{fc} & A_{f\mathcal{X}} \\ A_{cf} & A_{cc}^N & A_{c\mathcal{X}}^N \\ A_{\mathcal{X}f} & A_{\mathcal{X}c}^N & A_{\mathcal{X}\mathcal{X}}^N \end{bmatrix}.$$

We use the superscript “N” for the blocks which differ from the corresponding blocks of  $A_{\bar{\Omega}(i)}$ , the principal submatrix of the original matrix  $A$  corresponding to the subdomain  $\bar{\Omega}(i)$ . Note that the “N” blocks are not accessible (available) and not used in our algorithm. We have  $E_i v|_{\Omega_{\mathcal{X}}(i)} = E_{\mathcal{X}c} v_c$ . Hence,  $a_{\bar{\Omega}(i)}(w, w)$  for  $w_c = v_c$  and  $w_{\mathcal{X}} = E_i w|_{\Omega_{\mathcal{X}}(i)}$  leads to the following matrix expression:

$$\begin{aligned} a_{\bar{\Omega}(i)}(w, w) &= \begin{bmatrix} w_f \\ v_c \\ E_{\mathcal{X}c} v_c \end{bmatrix}^T \begin{bmatrix} A_{ff} & A_{fc} & A_{f\mathcal{X}} \\ A_{cf} & A_{cc}^N & A_{c\mathcal{X}}^N \\ A_{\mathcal{X}f} & A_{\mathcal{X}c}^N & A_{\mathcal{X}\mathcal{X}}^N \end{bmatrix} \begin{bmatrix} w_f \\ v_c \\ E_{\mathcal{X}c} v_c \end{bmatrix} \\ &= \begin{bmatrix} w_f \\ v_c \end{bmatrix}^T \begin{bmatrix} A_{ff} & A_{fc} + A_{f\mathcal{X}}E_{\mathcal{X}c} \\ A_{cf} + E_{\mathcal{X}c}^T A_{\mathcal{X}f} & A_{cc}^N + A_{c\mathcal{X}}^N E_{\mathcal{X}c} + E_{\mathcal{X}c}^T (A_{\mathcal{X}c}^N + A_{\mathcal{X}\mathcal{X}}^N E_{\mathcal{X}c}) \end{bmatrix} \\ &\quad \times \begin{bmatrix} w_f \\ v_c \end{bmatrix}. \end{aligned}$$

Minimizing this symmetric positive semi-definite quadratic form with respect to  $w_f$  is equivalent to solving the equation

$$A_{ff} w_f + (A_{fc} + A_{f\mathcal{X}}E_{\mathcal{X}c})v_c = 0,$$

which is the same equation that specifies  $v_f$  in the element-free AMGe interpolation procedure.  $\square$

In the next lemma we will remove the constraint on  $v$  being fixed at the  $\Omega_{\mathcal{X}}(i)$  points.

**Lemma 7.2.** *The following quadratic forms are spectrally equivalent,*

$$q_1(v_c, v_c) \equiv \inf_{v: v|_{\Omega_c(i)} = v_c} a_{\bar{\Omega}(i)}(v, v), \quad \text{and} \quad q_2(v_c, v_c) \equiv \inf_{\substack{v: v_c \text{ fixed} \\ v_{\mathcal{X}} = E_i v}} a_{\bar{\Omega}(i)}(v, v).$$

*That is, there exists a positive constant  $\eta$  such that*

$$q_1(v_c, v_c) \leq q_2(v_c, v_c) \leq \eta q_1(v_c, v_c) \quad \text{for all } v_c.$$

*Proof.* It suffices to show that the two quadratic forms have the same null-space. The null-space of  $q_1$  is  $v_c = \text{const}$  and the null-space of  $q_2$  is the same as that of  $a_{\bar{\Omega}(i)}(v, v)$  with  $v : v_c = \text{const}$  and  $E_i v = \text{const}$  on  $\Omega_{\mathcal{X}}(i)$ . Note that  $a_{\bar{\Omega}(i)}(v, v) = 0$  implies  $v_f$  is the same constant as  $v_c$ . Then  $E_i v$  is also the same constant, since it is an averaging operator based on the values of  $v_c$  and  $v_f$ . Hence the forms  $q_1$  and  $q_2$  both vanish only for constant  $v_c$ . In order to show that the constant  $\eta$  is bounded independently of  $E_i$ , one first easily sees that

$$a_{\bar{\Omega}(i)}(v, v) \leq C \sum_{e \subset \bar{\Omega}(i)} \varrho(e) \sum_{l, k \in e} (v(l) - v(k))^2.$$

The constant  $C$  depends only on the number of points used in the averaging procedure ( $E_i$ ), i.e., it is bounded by the total number of coarse points  $\Omega_c(i)$  (plus the interior point  $i$ ). The dofs  $l$  and  $k$  in the summation are either coarse dofs or  $i$ , and  $\varrho(e)$  is defined in (7.1) to be the maximal value of the local ellipticity bound associated with the original elliptic operator coefficient  $a(x)$ . More specifically, for each  $i_{\mathcal{X}} \in \Omega_{\mathcal{X}}(i)$

$$v(i_{\mathcal{X}}) \equiv (E_i v)(i_{\mathcal{X}}) = \sum_{k \in \Omega_c(i) \cup \{i\}} \alpha_{i_{\mathcal{X}}, k} v(k),$$

where

$$\sum_{k \in \Omega_c(i) \cup \{i\}} \alpha_{i_{\mathcal{X}}, k} = 1, \quad \text{and} \quad \alpha_{i_{\mathcal{X}}, k} \geq 0.$$

Then, for any  $j \in \Omega_c(i) \cup \{i\}$ ,

$$(v(i_{\mathcal{X}}) - v(j)) = \sum_{k \in \Omega_c(i) \cup \{i\}} \alpha_{i_{\mathcal{X}}, k} (v(k) - v(j)),$$

and hence

$$\begin{aligned} (v(i_{\mathcal{X}}) - v(j))^2 &\leq \sum_{k \in \Omega_c(i) \cup \{i\}} \alpha_{i_{\mathcal{X}}, k}^2 \sum_{k \in \Omega_c(i) \cup \{i\}} (v(k) - v(j))^2 \\ &\leq (1 + |\Omega_c(i)|) \sum_{k \in \Omega_c(i) \cup \{i\}} (v(k) - v(j))^2. \end{aligned}$$

As a result we see that, for  $v_{\mathcal{X}} = E_i v$ ,

$$\begin{aligned} q_2(v_c, v_c) &\leq a_{\bar{\Omega}(i)}(v, v) \\ &\leq C \sum_{e \subset \bar{\Omega}(i)} \varrho(e) \sum_{l, k \in e} (v(l) - v(k))^2 \\ &\leq C(1 + |\Omega_c(i)|) \frac{\max_{e \in \bar{\Omega}(i)} \varrho(e)}{\min_{e \in \bar{\Omega}(i)} \varrho(e)} \sum_{e \subset \bar{\Omega}(i)} \varrho(e) \sum_{k, j \in e \cap (\Omega_c(i) \cup \{i\})} (v(k) - v(j))^2. \end{aligned}$$

Finally, since  $v_f$  is arbitrary on the right-hand side of this inequality,

$$q_2(v_c, v_c) \leq C \frac{\max_{e \in \bar{\Omega}(i)} \varrho(e)}{\min_{e \in \bar{\Omega}(i)} \varrho(e)} (1 + |\Omega_c(i)|) \inf_{v_f} \left( \sum_{e \subset \Omega_c(i)} \varrho(e) \sum_{k, j \in e \cap (\Omega_c(i) \cup \{i\})} (v(k) - v(j))^2 \right).$$

It is also true that

$$q_1(v_c, v_c) \simeq \inf_{v: v_c \text{ fixed}} \left( \sum_{e \subset \bar{\Omega}(i)} \varrho(e) \sum_{l, k \in e} (v(l) - v(k))^2 \right).$$

This shows that  $\eta$  can be chosen bounded independently of the actual averaging extension mapping  $E_i$ .  $\square$

Then the following corollary, involving the element-free AMGe interpolated vector  $Pv_c$ , is proved in the same way as Lemma 7.2.

**Corollary 7.1.** *Consider the extended neighborhood of  $i$ ,  $\hat{\Omega}(i) = \cup\{e, e \subset \bar{\Omega}(i) \text{ or } e \subset \bar{\Omega}(j), \text{ for all } j \in \Omega_{\mathcal{X}}(i)\}$ . There is a constant  $\kappa = \kappa_{\hat{\Omega}(i)} > 0$ , locally estimated, such that the following bound holds:*

$$a_{\bar{\Omega}(i)}(Pv_c, Pv_c) \leq \kappa \inf_{w: w_c = v_c} a_{\hat{\Omega}(i)}(w, w).$$

*Proof.* Let  $v$  be defined on  $\hat{\Omega}(i)$  as follows:

$$v(k) = \begin{cases} (Pv_c)(k), & k \in \bar{\Omega}(i), \\ v_c(k), & k \text{ is a coarse dof outside } \bar{\Omega}(i), \\ (E_j v)(k), & k \in \Omega_{\mathcal{X}}(j), \text{ for some } j \in \Omega_{\mathcal{X}}(i). \end{cases}$$

We see that  $v$  at every fine dof  $k$  in  $\hat{\Omega}(i)$  is an average value of some neighboring coarse dofs from  $\hat{\Omega}(i)$ . Hence, in the same way as in the proof of Lemma 7.2, we establish the inequality

$$a_{\hat{\Omega}(i)}(v, v) \leq \kappa \inf_{w: w_c = v_c} a_{\hat{\Omega}(i)}(w, w).$$

Since  $a_{\bar{\Omega}(i)}(Pv_c, Pv_c) \leq a_{\hat{\Omega}(i)}(v, v)$ , the desired result follows.  $\square$

For each fine dof  $i$ , define  $\mathcal{Z}(i)$  to be the number of overlapping domains on  $\hat{\Omega}(i)$ , that is, the number of domains  $\hat{\Omega}(j)$  such that  $\hat{\Omega}(j) \cap \hat{\Omega}(i) \neq \emptyset$ . Then we may state the following theorem.

**Theorem 7.1.** *The element-free interpolation mapping  $P$  exhibits the following approximate harmonic property:*

$$a(Pv_c, Pv_c) \leq \kappa \inf_{w: w_c = v_c} a(w, w),$$

where the constant  $\kappa = \max_{i=\text{fine dof}} \kappa_{\hat{\Omega}(i)} \mathcal{Z}(i)$ , and the  $\kappa_{\hat{\Omega}(i)}$  are the local constants from Corollary 7.1.

*Proof.* The proof simply follows from the fact that

$$a(Pv_c, Pv_c) \leq \sum_{i=\text{fine dof}} a_{\bar{\Omega}(i)}(P_cv, P_cv)$$

and by summation of the local estimates from Corollary 7.1.  $\square$

Another important property of the element-free interpolation mapping  $P$  is that it partitions unity, as we show in the following theorem.

**Theorem 7.2.**  *$P$  provides a partition of unity. Specifically, the row sums of  $P$  are 1.*

*Proof.* Let  $v_f = Pv_c$  be given by  $v_f = \sum_{i_c \in \Omega_c(i)} \alpha_{i, i_c} v_c(i_c)$ . Assume that  $v_c(i_c) = 1$  on  $\Omega_c(i)$ . Now,  $P$  uses the formula that minimizes (7.2) and the minimum (zero) is achieved for  $E_i v(j) = 1$  at  $\Omega_{\mathcal{X}}(i)$  and  $v_f = 1$ . That is, we find that

$$1 = \sum_{i_c \in \Omega_c(i)} \alpha_{i, i_c},$$

which is the desired unity row-sum property of  $P$ .  $\square$

**Remark 7.2.** *Theorems 7.1 and 7.2 are the main goals of many two-grid convergence analyses and they imply convergence of the respective two-grid AMG methods, cf., e.g., [14], [8], [13], and [7].*

## 8. NUMERICAL EXPERIMENTS

We describe here several sets of numerical experiments designed to test the efficacy of the element-free AMGe methods described above. For each of several problems we apply a set of interpolation rules within an AMG code. The problems are then solved using a CG solver, preconditioned with one V-cycle of AMG.

The interpolation rules are:

- the AMGe rule [7] for the finite element problems;
- three element-free AMGe rules from Section 6:
  1.  $L_2$ -extension;
  2.  $A$ -extension;
  3. (only for scalar PDE) the simultaneous extension based on minimizing the quadratic functional described in Section 6.

For systems problems the unknowns are split into physical variables. That is, for scalar problems the rule is as described in Section 6, while for 2-d elasticity, with physical variables  $u$  and  $v$  (displacement in the  $x$ - and  $y$ -directions, respectively) we perform the extensions (and associated interpolation) of exterior dofs of type  $u$  using only neighborhood dofs of type  $u$ ; similarly, the extension to exterior dofs of type  $v$  are carried out using neighborhood dofs of type  $v$ ; this applies both to  $L_2$  and  $A$ -extensions. The local neighborhood about a point is defined by the sparsity pattern of the matrix about that point and the averaging involves only dofs from the sparsity pattern set  $S$  (see Section 6).

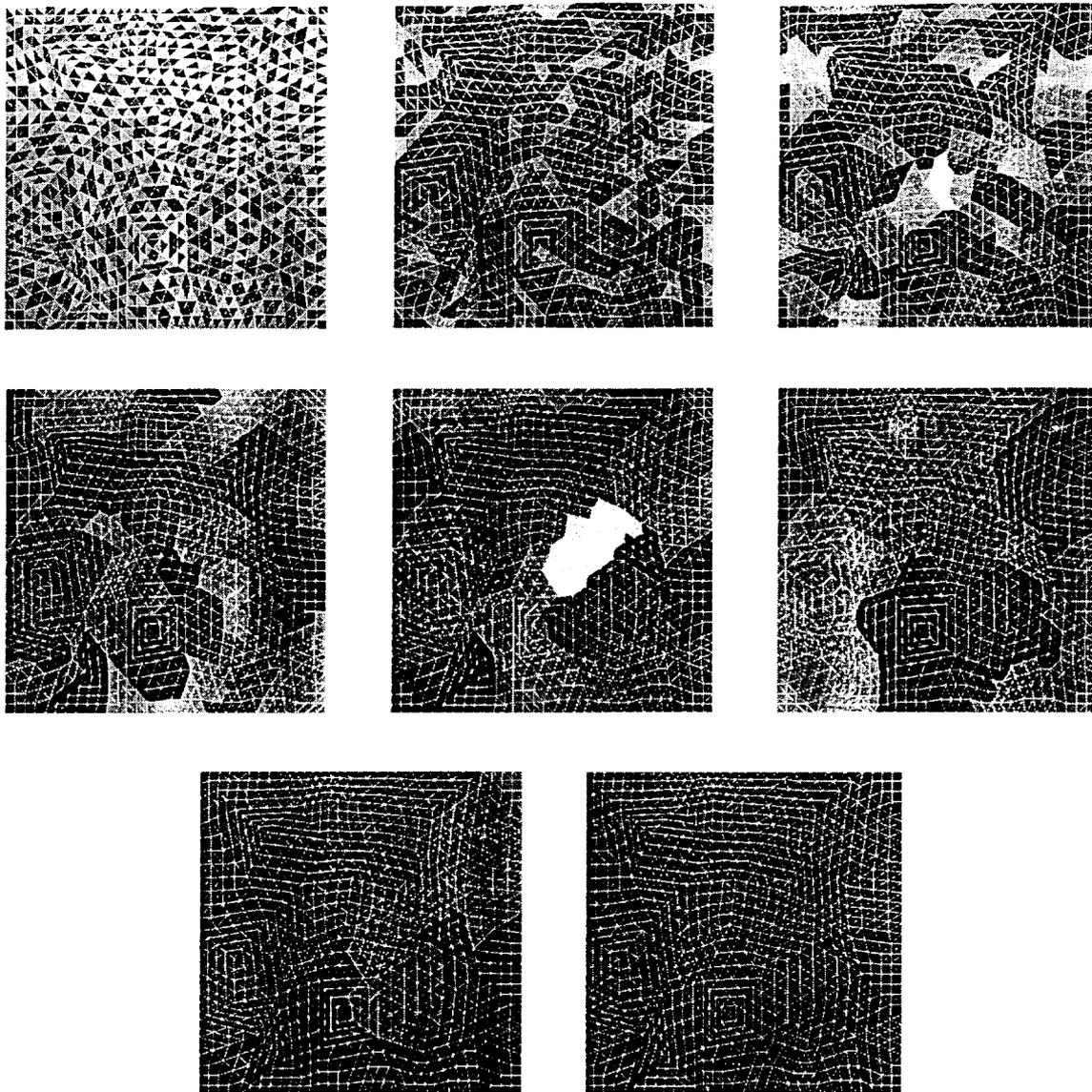


FIGURE 4. Sequence increasingly coarse elements, formed by element agglomeration.

8.1. An elliptic problem on a triangular element mesh. We apply the various interpolation rules to a second order elliptic PDE

$$(8.1) \quad -\nabla \cdot (A(x, y) \nabla u) = f(x, y) \quad \text{on } G$$

$$(8.2) \quad u(x, y) = g(x, y) \quad \text{on } \partial G$$

where  $G$  is the unit square. The matrix of diffusion coefficients includes functions with relatively benign characteristics— there is both spatial variability and jump discontinuity in the coefficients, but the jumps are of relatively small magnitude and the variation is mild. The discretization is by a finite-element method on an unstructured triangular mesh. The coarsening algorithm is one of element agglomeration. That is, the coarse grids are the vertices of coarse elements produced by an agglomeration algorithm proposed in [7]. Figure 4 displays the coarsening sequence for a typical

problem. Here the fine grid comprises 1600 elements, the first coarse grid has 382 elements, and the remaining grids have 93, 33, 15, 7, 3, and 1 elements. Table 1 gives the coarsening details for four different versions of this problem. It may be seen that

TABLE 1. *Coarsening history for the problem  $-\nabla \cdot A(x,y)\nabla u = f$  on an unstructured triangular fine grid. For each level of each problem size. “nz” is the number of nonzero entries in the operator matrix, “dofs” gives the number of degrees of freedom, and “elts” gives the number of finite elements in the agglomerated grid.*

level		No. of elements			
		25600	6400	1600	400
0	nz	90321	22761	5781	1491
	dofs	13041	3321	861	231
	elts	25600	6400	1600	400
1	nz	32898	9540	2602	1094
	dofs	4108	1152	330	114
	elts	6013	1427	382	76
2	nz	14305	4361	1397	470
	dofs	1507	451	143	50
	elts	1489	374	93	26
3	nz	7193	2098	634	199
	dofs	643	198	64	23
	elts	392	117	33	11
4	nz	3458	975	304	88
	dofs	302	91	32	12
	elts	158	47	15	5
5	nz	1580	453	126	36
	dofs	140	45	16	6
	elts	70	22	7	2
6	nz	714	188	46	16
	dofs	68	22	8	4
	elts	33	10	3	1
7	nz	274	84	16	
	dofs	30	12	4	
	elts	14	5	1	
8	nz	120	30		
	dofs	16	6		
	elts	7	2		
9	nz	42	16		
	dofs	8	4		
	elts	3	1		
10	nz	16			
	dofs	4			
	elts	1			

the number of elements decreases by about 75% at each coarsening for the first few

coarsenings, after which it decreases by about 50% per level. The number of nonzero entries in the matrix decreases by approximately 50% per level, while the number of degrees of freedom tends to decrease by 50-60% with each successive level.

For each of the four interpolation rules, the problem is solved using a preconditioned conjugate gradient method, where the preconditioning consists of a single  $V(1,1)$ -cycle of AMG, with a Gauß-Seidel smoother. The iteration is run until the residual is less than  $10^{-8}$  in norm. We report the results in Table 2. For each problem size we display, for each interpolation rule, the number of preconditioned CG iterations required to achieve the desired residual size and  $\varrho$ , the average convergence factor over the iterations.

TABLE 2. *CG convergence results; unstructured triangular fine grid; second order elliptic problem;  $V(1,1)$ -cycle MG, Gauß-Seidel smoother used as preconditioner.*

Interp. rule		400 elts	1600 elts	6400 elts	25600 elts
AMGe	iterations	14	16	21	23
	$\varrho$	0.115	0.172	0.252	0.289
$A$ -extension	iterations	13	15	19	20
	$\varrho$	0.118	0.158	0.218	0.247
$L_2$ -extension	iterations	13	16	19	21
	$\varrho$	0.119	0.161	0.227	0.249
quadratic funct. min.	iterations	13	15	19	19
	$\varrho$	0.105	0.152	0.222	0.231

Examination of the results reveals that all three of the extension methods,  $A$ -extension,  $L_2$ -extension, and quadratic functional minimization, perform at least as well on this problem as does AMGe. In some cases the performance of the extension methods is marginally better than AMGe. The amount of work entailed for the  $A$ -extension and the  $L_2$ -extension methods is comparable to that of AMGe, provided that the neighborhoods are selected to be of comparable size as the element neighborhoods (which is the case in these experiments). For the quadratic functional minimization the work is somewhat greater, but still comparable. The advantage of the element-free methods is, of course, that there is no requirement to have the actual individual stiffness matrices that are required in AMGe. For this experiment this represents a considerable savings in storage.

**8.2. Two dimensional elasticity, the thin beam.** We consider next the two dimensional plane-stress elasticity problem on a cantilevered beam, fixed at one end. The domain of the problem is  $G = (0, 1) \times (0, d)$  with  $d \leq 1$ . For  $d \ll 1$  this is the thin-beam problem. The problem is

$$\begin{aligned} u_{xx} + \frac{1-\nu}{2} u_{yy} + \frac{1+\nu}{2} v_{xy} &= f_1, \\ \frac{1+\nu}{2} u_{xy} + \frac{1-\nu}{2} v_{xx} + v_{yy} &= f_2, \end{aligned}$$

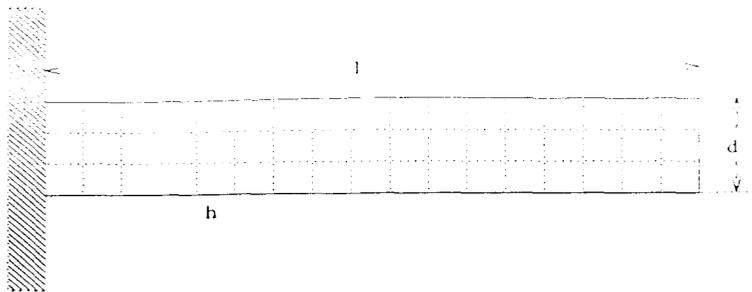


FIGURE 5. *The thin-beam elasticity problem domain. Homogeneous Dirichlet boundary conditions are applied at  $x = 0$ .*

where  $u$  and  $v$  are displacements in the  $x$  and  $y$  directions, respectively. This can be a difficult problem for standard multigrid methods, especially when the domain is long and thin. The problem is discretized using uniform square finite elements of size  $h$ . Nodal coarsening is used, with the coarse nodes being the vertices of elements created by the agglomeration algorithm from [7]. After certain levels of coarsening the algorithm agglomerates only along the  $x$ -direction.

We present results in both the thick beam ( $d = 1.0$ ) and thin beam ( $d = 0.05$ ) cases. For each case we present results for three sizes of the discretization parameter:  $h = 0.05, 0.025$ , and  $0.0125$  for the thick beam and  $h = 0.025, 0.0125$ , and  $0.00625$  for the thin beam. The coarsening histories of the agglomeration algorithm are shown in Table 3. Table 4 shows the results of the experiments for the beam problem. As in section 8.1, preconditioned conjugate gradient is used as the solver, with a single  $V(1,1)$ -cycle of AMG as the preconditioner, with a Gauß-Seidel smoother. For this problem we show the number of iterations required to achieve a residual norm less than  $10^{-8}$ , and also the convergence factor of the final iteration. For this problem we do not implement the quadratic minimization method described in section 6. That method is for scalar problems, while this problem is a system of PDEs. We use the standard AMGe method and compare it with the  $A$ - and  $L_2$ -extension methods described above. Our expectation is that AMGe should outperform the element-free methods, at least on the thin beam problem; this is the problem for which AMGe was originally developed. We observe, however, that for the thick beam problems the element-free methods both outperform AMGe. First, we note that it takes fewer iterations to reach the tolerance. It is also apparent that the element-free methods are more scalable, in that the number of iterations does not grow with the problem size. The AMGe method requires more iterations for larger problems.

For the thin beam problem, we observe the results we naturally expect. That is, AMGe outperforms the element-free methods, requiring fewer iterations. Further, AMGe appears to be more scalable on this problem than the extension methods. The  $L_2$ -extension method exhibits a distinct lack of scalability as the problem grows larger.

TABLE 3. *Coarsening history; structured rectangular fine grid; 2-d elasticity,  $d = 1$* 

level		Thick Beam $d = 1.0$			Thin Beam $d = 0.05$		
		$h = 0.050$	$h = 0.025$	$h = 0.0125$	$h = 0.025$	$h = 0.0125$	$h = 0.00625$
0	nz	14884	58564	232324	3388	12532	48100
	dofs	882	3362	13122	246	810	2898
1	nz	10440	40880	161760	1664	7328	30656
	dofs	264	924	3444	88	252	820
2	nz	4128	17248	70488	784	3744	10152
	dofs	84	264	924	44	132	252
3	nz	1000	4956	19056	384	1152	3816
	dofs	32	94	284	24	48	132
4	nz	256	1404	6128	144	384	1152
	dofs	16	38	104	12	24	48
5	nz	64	324	1668	64	144	384
	dofs	8	18	42	8	12	24
6	nz		144	576		64	144
	dofs		12	24		8	12
7	nz		64	144			64
	dofs		8	12			8
8	nz			64			
	dofs			8			

## 9. CONCLUSIONS

In this paper we propose a general rule for building interpolation weights in AMG, thus extending the applicability of AMG to more general settings than the traditional M-matrix case. The applications include elliptic problems on unstructured finite element grids, where both scalar problems and systems (like elasticity) are considered. The element-free AMGe method seems as competitive as the AMGe methods but entail much less overhead. The element information and the element matrices, in particular, are essential for the AMGe methods but are not required for element-free AMGe. If we assume more information is available (such as the rigid body modes in the case of elasticity) it may be incorporated into the construction of the extension mappings. Thus element-free AMGe can be made to reproduce the extra modes in the interpolation from their coarse values. This property is important in the AMG methods for elasticity problems (cf. [12]), and incorporating it into element-free AMGe is a subject of ongoing research.

## REFERENCES

- [1] A. BRANDT, *Generally highly accurate algebraic coarsening*, Elec. Trans. Num. Anal., 10 (2000), pp. 1–20.

TABLE 4. CG convergence results; structured rectangular fine grid; 2-d elasticity,  $d = 1$ ,  $V(1, 1)$ -cycle MG. Gauß–Seidel smoother used as preconditioner.

Thick Beam, $d = 1.0$				
Interp. rule		$h = 0.050$	$h = 0.025$	$h = 0.0125$
AMGe	iterations	16	18	20
	$\rho$	0.172	0.206	0.234
A-extension	iterations	12	12	12
	$\rho$	0.099	0.098	0.097
$L_2$ -extension	iterations	13	13	13
	$\rho$	0.101	0.102	0.104
Thin Beam, $d = 0.05$				
Interp. rule		$h = 0.025$	$h = 0.0125$	$h = 0.00625$
AMGe	iterations	17	18	19
	$\rho$	0.180	0.198	0.22
A-extension	iterations	20	23	22
	$\rho$	0.227	0.286	0.280
$L_2$ -extension	iterations	18	20	27
	$\rho$	0.203	0.243	0.254

- [2] A. BRANDT, S. F. MCCORMICK, AND J. W. RUGE, *Algebraic multigrid (AMG) for automatic multigrid solutions with application to geodetic computations*. Report, Inst. for Computational Studies, Fort Collins, Colo., October 1982.
- [3] ———, *Algebraic multigrid (AMG) for sparse matrix equations*, in Sparsity and Its Applications, D. J. Evans, ed., Cambridge University Press, Cambridge, 1984.
- [4] M. BREZINA, A. J. CLEARY, R. D. FALGOUT, V. E. HENSON, J. E. JONES, T. A. MANTEUFFEL, S. F. MCCORMICK, AND J. W. RUGE, *Algebraic multigrid based on element interpolation (AMGe)*. Submitted to the SIAM Journal on Scientific Computing, 1998.
- [5] A. J. CLEARY, R. D. FALGOUT, V. E. HENSON, AND J. E. JONES, *Coarse-grid selection for parallel algebraic multigrid*, in Proceedings of the Fifth International Symposium on Solving Irregularly Structured Problems in Parallel, vol. 1457 of Lecture Notes in Computer Science, Springer-Verlag, 1998, pp. 104–115.
- [6] A. J. CLEARY, R. D. FALGOUT, V. E. HENSON, J. E. JONES, T. A. MANTEUFFEL, S. F. MCCORMICK, G. N. MIRANDA, AND J. W. RUGE, *Robustness and scalability of algebraic multigrid*, SIAM Journal on Scientific Computing, 21 (2000), pp. 1886–1908.
- [7] J. E. JONES AND P. S. VASSILEVSKI, *Amge based on element agglomeration*, SIAM Journal on Scientific Computing, (to appear).
- [8] J. MANDEL, M. BREZINA, AND P. VANĚK, *Energy optimization of algebraic multigrid bases*, Computing, (to appear).
- [9] J. W. RUGE AND K. STÜBEN, *Algebraic multigrid (AMG)*, in Multigrid Methods, S. F. McCormick, ed., vol. 3 of Frontiers in Applied Mathematics, SIAM, Philadelphia, PA, 1987, pp. 73–130.
- [10] K. STÜBEN, *Algebraic multigrid (AMG): experiences and comparisons*, Appl. Math. Comput., 13 (1983), pp. 419–452.
- [11] ———, *A review of algebraic multigrid*. GMD Report 69, November 1999.
- [12] P. VANEK, M. BREZINA, AND R. TEZUAR, *Two-grid method for linear elasticity on unstructured meshes*, SIAM J. Sci. Comput., (to appear).

- [13] P. VANĚK, J. MANDEL, AND M. BREZINA, *Convergence of algebraic multigrid based on smoothed aggregation*. UCD/CCM Report 126. Center for Computational Mathematics, University of Colorado at Denver, February 1998. <http://www-math.cudenver.edu/ccmreports/rep126.ps.gz>.
- [14] W. L. WAN, T. F. CHAN, AND B. SMITH, *An energy-minimizing interpolation for robust multigrid methods*, SIAM J. Sci. Comput., (to appear).

# ROBUSTNESS AND SCALABILITY OF ALGEBRAIC MULTIGRID

ANDREW J. CLEARY\* , ROBERT D. FALGOUT\* , VAN EMDEN HENSON\* , JIM E. JONES\* , THOMAS A. MANTEUFFEL† , STEPHEN F. MCCORMICK† , GERALD N. MIRANDA‡ AND JOHN W. RUGE§

**Abstract.** Algebraic multigrid (AMG) is currently undergoing a resurgence in popularity, due in part to the dramatic increase in the need to solve physical problems posed on very large, unstructured grids. While AMG has proved its usefulness on various problem types, it is not commonly understood how wide a range of applicability the method has. In this study, we demonstrate that range of applicability, while describing some of the recent advances in AMG technology. Moreover, in light of the imperatives of modern computer environments, we also examine AMG in terms of algorithmic scalability. Finally, we show some of the situations in which standard AMG does not work well, and indicate the current directions taken by AMG researchers to alleviate these difficulties.

**Key words.** algebraic multigrid, interpolation, unstructured meshes, scalability

**1. Introduction.** Algebraic multigrid (AMG) was first introduced in the early 1980's [11, 8, 10, 12], and immediately attracted substantial interest [32, 28, 30, 29]. Research continued at a modest pace into the late 1980's and early 1990's [18, 14, 21, 25, 20, 26, 22]. Recently, however, there has been a major resurgence of interest in the field, for "classical" AMG as defined in [29], as well as for a host of other algebraic-type multilevel methods [3, 16, 34, 6, 2, 4, 5, 15, 33, 17, 35, 36, 37]. Largely, this resurgence in AMG research is due to the need to solve increasingly larger systems, with hundreds of millions or billions of unknowns, on unstructured grids. The size of these problems dictates the use of large-scale parallel processing, which in turn demands algorithms that scale well as problem size increases. Two different types of scalability are important. *Implementation scalability* requires that a single iteration be scalable on a parallel computer. Less commonly discussed is *algorithmic scalability*, which requires that the computational work per iteration be a linear function of the problem size and that the convergence factor per iteration be bounded below 1 with bound independent of problem size. This type of scalability is a property of the algorithm, independent of parallelism, but is a necessary condition before a scalable implementation can be attained.

Multigrid methods are well known to be scalable (both types) for elliptic problems on regular grids. However, many modern problems involve extremely complex geometries, making structured geometric grids extremely difficult, if not impossible, to use. Application code designers are turning in increasing numbers to very large unstructured grids, and AMG is seen by many as one of the most promising methods for solving the large-scale problems that arise in this context.

This study has four components. First, we examine the performance of "classical" AMG on a variety of problems having regular structure, with the intent of determining its robustness. Second, we examine the performance of AMG on the same suite of problems, but now with unstructured grids and/or irregular domains. Third, we study the algorithmic scalability of AMG by examining its performance on several of

---

\* Center for Applied Scientific Computing (CASC), Lawrence Livermore National Laboratory, Livermore, CA. Email: {cleary, ralgout, vhenson, jjones}@llnl.gov

† Department of Applied Mathematics, University of Colorado, Boulder, CO. Email: {tmanteuf, stevem}@boulder.colorado.edu

‡ USS Florida (SSBN-728), Naval Submarine Base, Silverdale, WA, Email: JerryTrish@aol.com

§ Front Range Scientific, Boulder, CO. Email: jruge@sobolev.Colorado.EDU

the problems using grids of increasing sizes. Finally, we introduce a new method for computing interpolation weights, and we show that in certain troublesome cases it can significantly improve AMG performance.

Our study differs from previous reports on the performance of AMG (e.g., [29, 30]) primarily by our examination of algorithmic scalability, our emphasis on unstructured grids, and the introduction of a new algorithm for computing interpolation weights. In Section 2, a description of some details of the AMG algorithm is given to provide an understanding of the results and later discussion. In Section 3, we present results of AMG applied to a range of symmetric scalar problems, using finite element discretizations on structured and unstructured 2D and 3D meshes. AMG is also tested on nonsymmetric problems, on both structured and unstructured meshes, and the results are presented in Section 4. A version of AMG designed for systems of equations is tested, with the focus on problems in elasticity. Results are discussed in Section 5. In Section 6, we introduce and report on tests of a new method for computing interpolation weights. We conclude with some remarks in Section 7.

**2. The Scalar AMG Algorithm.** We begin by outlining the basic principles and techniques that comprise AMG. Detailed explanations may be found in [29]. Consider a problem of the form

$$(1) \quad \mathbf{A}\mathbf{u} = \mathbf{f},$$

where  $A$  is an  $n \times n$  matrix with entries  $a_{ij}$ . For convenience, the indices are identified with grid points, so that  $u_i$  denotes the value of  $\mathbf{u}$  at point  $i$ , and the grid is denoted by  $\Omega = \{1, 2, \dots, n\}$ . In any multigrid method, the central idea is that error not eliminated by relaxation must be removed by coarse-grid correction. Applied to elliptic problems, for example, simple relaxations (Jacobi, Gauss-Seidel) reduce high frequency error components efficiently, but are very slow at removing smooth components. However, the smooth error that remains after relaxation can be approximated accurately on a coarser grid. This is done by solving the residual equation  $A\mathbf{e} = \mathbf{r}$  on a coarser grid, then interpolating the error back to the fine grid and using it to correct the fine-grid approximation. The coarse-grid problem itself is solved by a recursive application of this method. One iteration of this process, proceeding through all levels, is known as a multigrid cycle. In geometric multigrid, standard uniform coarsening and linear interpolation are often used, so the main design task is to choose a relaxation scheme that reduces errors the coarsening process cannot approximate. One purpose of AMG is to free the solver from dependence on geometry, so AMG instead fixes relaxation (normally Gauss-Seidel), and its main task is to determine a coarsening process that approximates error that this relaxation cannot reduce.

An underlying assumption in AMG is that smooth error is characterized by small residuals, that is,  $A\mathbf{e} \approx \mathbf{0}$ , which is the basis for choosing coarse grids and defining interpolation weights. For simplicity of discussion here, we assume that  $A$  is a symmetric positive-definite  $M$ -matrix, with  $a_{ii} > 0$ ,  $a_{ij} \leq 0$  for  $j \neq i$ , and  $\sum a_{ij} \geq 0$ . This assumption is made for convenience; AMG will frequently work well on matrices that are not  $M$ -matrices. To define any multigrid method, several components are required. Using superscripts to indicate level number, where 1 denotes the finest level so that  $A^1 = A$  and  $\Omega^1 = \Omega$ , the components that AMG needs are as follows:

1. "Grids"  $\Omega^1 \supset \Omega^2 \supset \dots \supset \Omega^M$ .
2. Grid operators  $A^1, A^2, \dots, A^M$ .
3. Grid transfer operators:  
Interpolation  $I_{k+1}^k, k = 1, 2, \dots, M - 1$ ,

Restriction  $I_k^{k+1}$ ,  $k = 1, 2, \dots, M - 1$ .

4. Relaxation scheme for each level.

Once these components are defined, the recursively defined cycle is as follows:

**Algorithm:**  $MV^k(\mathbf{u}^k, \mathbf{f}^k)$ . The  $(\mu_1, \mu_2)$  V-cycle.

If  $k = M$ , set  $\mathbf{u}^M = (A^M)^{-1}\mathbf{f}^M$ .

Otherwise:

Relax  $\mu_1$  times on  $A^k \mathbf{u}^k = \mathbf{f}^k$ .

Perform coarse grid correction:

Set  $\mathbf{u}^{k+1} = 0$ ,  $\mathbf{f}^{k+1} = I_k^{k+1}(\mathbf{f}^k - A^k \mathbf{u}^k)$ .

“Solve” on level  $k+1$  with  $MV^{k+1}(\mathbf{u}^{k+1}, \mathbf{f}^{k+1})$ .

Correct the solution by  $\mathbf{u}^k \leftarrow \mathbf{u}^k + I_{k+1}^k \mathbf{u}^{k+1}$ .

Relax  $\mu_2$  times on  $A^k \mathbf{u}^k = \mathbf{f}^k$ .

For this cycle to work efficiently, relaxation and coarse-grid correction must work together to effectively reduce all error components. This gives two principles that guide the choice of the components:

**P1:** *Error components not efficiently reduced by relaxation must be well approximated by the range of interpolation.*

**P2:** *The coarse-grid problem must provide a good approximation to fine-grid error in the range of interpolation.*

Each of these affects a different set of components: given a relaxation scheme, **P1** determines the coarse grids and interpolation, while **P2** affects restriction and the coarse grid operators. In order to satisfy **P1**, AMG takes an algebraic approach: relaxation is fixed, and the coarse grid and interpolation are automatically chosen so that the range of the interpolation operator accurately approximates slowly diminishing error components (which may not always appear to be “smooth” in the usual sense). **P2** is satisfied by defining restriction and the coarse-grid operator by the *Galerkin* formulation:

$$(2) \quad I_k^{k+1} = (I_{k+1}^k)^T \quad \text{and} \quad A^{k+1} = I_k^{k+1} A^k I_{k+1}^k.$$

When  $A$  is symmetric positive definite, this ensures that the correction from the exact solution of the coarse-grid problem is the best approximation in the range of interpolation [23], where “best” is meant in the  $A$ -norm: by  $\|\mathbf{v}\|_A \equiv \langle A\mathbf{v}, \mathbf{v} \rangle^{1/2}$ .

The choice of components in AMG is done in a separate preprocessing step:

**AMG Setup Phase:**

1. Set  $k = 1$ .
2. Partition  $\Omega^k$  into disjoint sets  $C^k$  and  $F^k$ .
  - (a) Set  $\Omega^{k+1} = C^k$ .
  - (b) Define interpolation  $I_{k+1}^k$ .
3. Set  $I_k^{k+1} = (I_{k+1}^k)^T$  and  $A^{k+1} = I_k^{k+1} A^k I_{k+1}^k$ .
4. If  $\Omega^{k+1}$  is small enough, set  $M = k + 1$  and stop. Otherwise, set  $k = k + 1$  and go to step 2.

Step 2 is the core of the AMG setup process. Since the focus is on coarsening a particular level  $k$ , such superscripts are omitted here and  $c$  and  $f$  are substituted for  $k + 1$  and  $k$  where necessary to avoid confusion. The goal of the setup phase is to choose the set  $C$  of coarse-grid points and, for each fine-grid point  $i \in F \equiv \Omega - C$ , small set  $C_i \subset C$  of interpolating points. Interpolation is then of the form:

where  $f_s$  is the number of flops required to coarsen a subgrid. Letting  $f_s = 90$  (this is representative of what appears in the PFMG code mentioned below), we have that  $\hat{P} \approx 207$ . For the PFMG algorithm, we set

$$K_\alpha \alpha = (c_1 + c_2 + L) P f_s \gamma,$$

which yields a  $\hat{P}$  that depends on  $n$ ,  $c_1$ , and  $c_2$ . However, we can bound  $\hat{P}$  as follows

$$(6/f_s)(\alpha/\gamma) \leq \hat{P} \leq (26/f_s)(\alpha/\gamma). \quad (7)$$

In the case of an isotropic problem, the smoothing cost per  $V$ -cycle for the PFMG algorithm is the same as for MG, hence the lower bound in (7). The upper bound is roughly a factor of four larger, so that  $\hat{P} \approx 898$  for the parameters being considered here. Note from (6) and (7) that  $\hat{P}$  depends strongly on the ratio of communication latency to computation speed.

This analysis also bears out in practice. In Figure 1, we present results from an MPI-implementation of PFMG run on an Intel Paragon. The problem solved was the anisotropic diffusion problem (1) with  $n = 40$ ,  $\varepsilon_1 = 1/10$ , and  $\varepsilon_2 = 1/100$ . The figure compares the cost of coarsening using approach A2 (labeled “Coarsen”) with the cost of a  $V$ -cycle. The time for A2 was not computed directly, but estimated by taking the overall setup time, and subtracting the setup time for the single processor run. The figure suggests that the cost of replicating the grid coarsening procedure is greater than the cost of a  $V(1,1)$  cycle when  $P$  is larger than about 500.

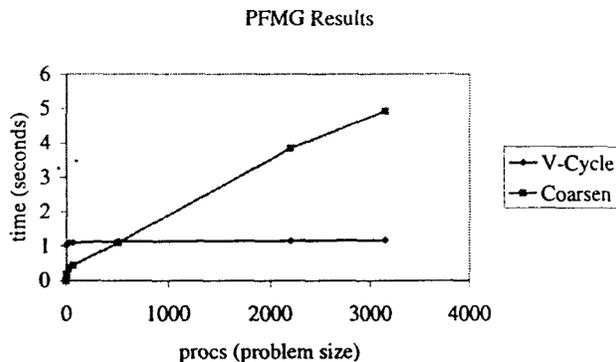


Fig. 1. PFMG results on an Intel Paragon comparing the cost of grid coarsening to the cost of a  $V$ -cycle.

### 3.2 Ghost Zones

The notion of *ghost zones* or *shadow zones* is commonly used in parallel linear solver codes, and is simply the extra “layer” of data needed from off-process to complete an on-process computation. The size of the ghost-zone

layer can vary depending on the algorithm implementation. We will consider here the use of a single layer of ghost zones in the library setting described earlier. Figure 2 illustrates (in 2D) the layout of data and ghost zones for two  $7 \times 7$  subgrids, and shows a typical communication pattern for a 5-point stencil computation. Note that, to simplify code, subgrid data and ghost-zone data are stored together as part of a single array in memory. To reduce the number of copies, this extra ghost-zone memory is always present in the vector data structure (note that ghost-zone memory is usually not persistent in unstructured-grid multigrid codes).

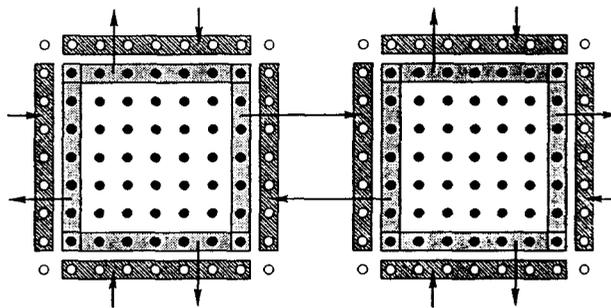


Fig. 2. Ghost zones and communications for a 5-point stencil and  $7 \times 7$  subgrids.

For the MG and PFMG algorithms, the storage overhead associated with ghost zones is quite acceptable. But, for more robust methods like SMG, ghost zone storage can be problematic. To see this, we can again use the models presented in Section 2. The coefficient multiplying  $\beta$  in each model also estimates the amount of ghost-zone storage used. In Figure 3, we plot this storage cost for the SMG algorithm relative to  $n^3$ , the cost of storing a vector. We see that the ghost-zone overhead is quite high, but we also note that the growth rate is moderate. That is, a  $\log_2 N$  dependence of the  $\beta$ -term in a model does not necessarily produce a ghost-zone memory overhead problem. For example, consider using alternating line relaxation in a full-coarsening multigrid method. Using a similar derivation as for SMG, it is easy to see that the ghost-zone storage cost is approximately  $6Ln^2$ , or about 30% that of SMG. In comparison, the overhead for PFMG for the problem described in Section 3.1 is about 0.6, and does not grow with  $P$ .

### 3.3 Mixed Programming Models

There is a recent trend to build large, parallel computers out of commodity parts. The largest such computers are clusters of shared memory processors (SMPs). In this section, we will discuss the use of mixed programming models for implementing parallel multigrid methods.

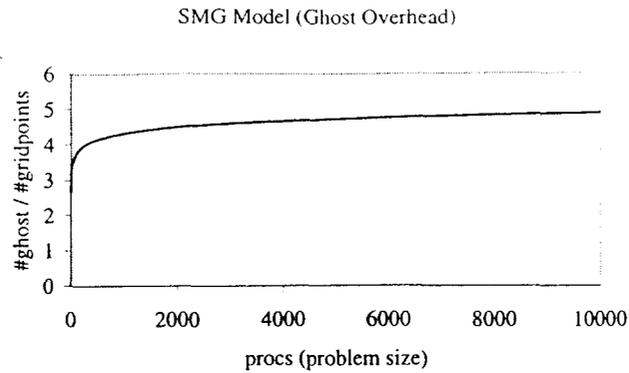


Fig. 3. SMG model illustrating relative storage costs of ghost zones.

Figure 4 illustrates two basic approaches for distributing (and computing on) subgrid data on a 4-processor SMP node. Pictured on the left (the *mixed model*) is one large subgrid with ghost layer (for communicating with other SMP nodes) and four regions of data, each assigned to different threads (these will usually be run on different processors). Pictured on the right (the *message-passing model*) are four subgrids with ghost layers, each subgrid assigned to different processes (again, these will usually be run on different processors).

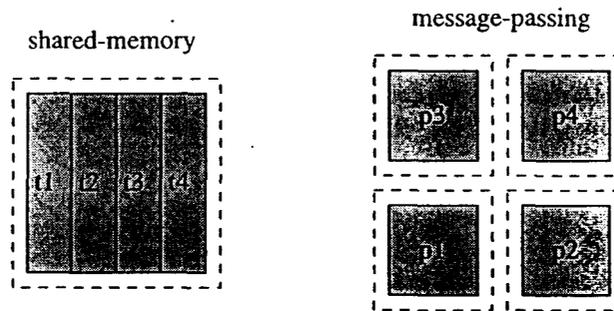


Fig. 4. Schematic of mixed model and message-passing model for a single 4-processor SMP node.

In theory, the mixed model has a couple of advantages over the pure message-passing model. The first advantage is a reduction in the number of messages going in and out of the SMP node. For example, for a 5-point stencil computation, the mixed model depicted in the figure requires 4 communications outside of the SMP and the message-passing model requires 8. The second advantage is the ghost-zone memory savings due to the fewer

and larger subgrids in the mixed model. In the figure, the ghost-zone memory savings is a factor of two. On SMPs with larger numbers of processors, the memory savings can be even more substantial.

Although the mixed model has these attractive features, our efforts to outperform the message-passing model have not yet succeeded in practice. We have developed two implementations of the mixed programming model, using MPI to do the message-passing in both cases. The first implementation uses POSIX threads, but we will discuss here only the second implementation, which uses OpenMP compiler directives. The approach taken was straightforward loop-level parallelism of the computational kernels in the code. Each of the kernels is a triply-nested loop over data associated with a subgrid. The OpenMP directives can only parallelize a single loop, so effective parallelism can only be achieved when the size of this loop is at least as large as the number of processors. Since multigrid methods—especially semicoarsening methods—produce grids of varying shapes and sizes, a fourth outer loop was added that explicitly decomposes the subgrid into roughly equal sized regions to be assigned to the different threads.

To be clear, consider the 2D example pictured on the left in Figure 4. Here, we have an outer loop as just described, but with only a doubly-nested inner loop. The outer loop has length four, and on each iteration, the inner loops iterate over the tall rectangular regions. If the outer loop is threaded using OpenMP, this means that each iteration is assigned to a different thread. Hence, the computations on each region in the figure are handled by different processors. The decomposition of the subgrid is done by simply subdividing the largest subgrid dimension by the number of threads being used.

In Figure 5, we show results comparing the MPI implementation to the mixed MPI-OpenMP implementation for conjugate gradient (CG) and CG with three different preconditioners: SMG, PFMG, and diagonal-scaling. We plot MPI time over MPI-OpenMP time. The MPI implementation is fastest in all cases.

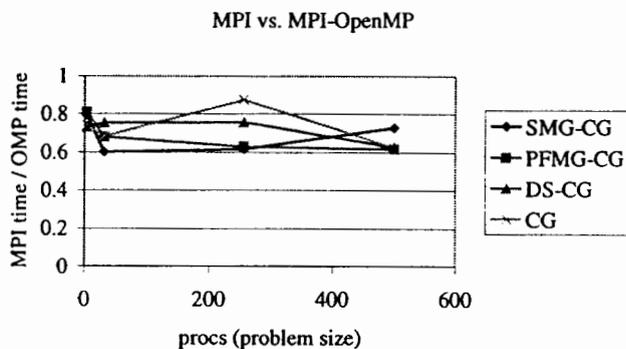


Fig. 5. Comparison of MPI and mixed MPI-OpenMP implementations of various solvers on an IBM SP2.

## 4 Conclusions

Extra care must be taken when developing codes for large-scale parallel architectures. Techniques commonly used for moderate-sized parallelism can be problematic for large-scale parallelism. Parallel performance models can provide useful implementation guidance, especially regarding the tradeoffs of replicating computations in order to reduce communications. On clusters of SMPs, mixed programming models have several advantages over straight message-passing, but these advantages are not yet born out in practice.

## References

1. S. F. Ashby and R. D. Falgout. A parallel multigrid preconditioned conjugate gradient algorithm for groundwater flow simulations. *Nuclear Science and Engineering*, 124(1):145–159, September 1996. Also available as LLNL Technical Report UCRL-JC-122359.
2. P. N. Brown, R. D. Falgout, and J. E. Jones. Semicoarsening multigrid on distributed memory machines. To appear in the SIAM Journal on Scientific Computing special issue on the Fifth Copper Mountain Conference on Iterative Methods. Also available as LLNL technical report UCRL-JC-130720, 1999.
3. W. D. Gropp and D. E. Keyes. Complexity of parallel implementation of domain decomposition techniques for elliptic partial differential equations. *SIAM J. Sci. Stat. Comput.*, 9:312–326, 1988.
4. J. E. Jones and S. F. McCormick. Parallel multigrid methods. In Keyes, Sameh, and Venkatakrishnan, editors, *Parallel Numerical Algorithms*, pages 203–224. Kluwer Academic, 1997.
5. S. Schaffer. A semi-coarsening multigrid method for elliptic partial differential equations with highly discontinuous and anisotropic coefficients. *SIAM J. Sci. Comput.*, 20(1):228–242, 1998.

# Language Interoperability for High-Performance Parallel Scientific Components

*N. Elliott, S. Kohn, B. Smolinski*

This paper was prepared for submittal to the  
International Symposium on Computing in Object-Oriented Parallel  
Environments, San Francisco, CA, September 29 – October 2, 1999

*U.S. Department of Energy*

Lawrence  
Livermore  
National  
Laboratory

**May 18, 1999**

DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

This is a preprint of a paper intended for publication in a journal or proceedings. Since changes may be made before publication, this preprint is made available with the understanding that it will not be cited or reproduced without the permission of the author.

This report has been reproduced  
directly from the best available copy.

Available to DOE and DOE contractors from the  
Office of Scientific and Technical Information  
P.O. Box 62, Oak Ridge, TN 37831  
Prices available from (423) 576-8401  
<http://apollo.osti.gov/bridge/>

Available to the public from the  
National Technical Information Service  
U.S. Department of Commerce  
5285 Port Royal Rd.,  
Springfield, VA 22161  
<http://www.ntis.gov/>

OR

Lawrence Livermore National Laboratory  
Technical Information Department's Digital Library  
<http://www.llnl.gov/tid/Library.html>

# Language Interoperability for High-Performance Parallel Scientific Components

Noah Elliott, Scott Kohn, Brent Smolinski

Center for Applied Scientific Computing, Lawrence Livermore National Laboratory,  
Livermore, CA.

**Abstract.** With the increasing complexity and interdisciplinary nature of scientific applications, code reuse is becoming increasingly important in scientific computing. One method for facilitating code reuse is the use of components technologies [16, 17, 9], which have been used widely in industry. However, components have only recently worked their way into scientific computing [2, 1, 11, 18]. Language interoperability is an important underlying technology for these component architectures. In this paper, we present an approach to language interoperability for a high-performance parallel, component architecture being developed by the Common Component Architecture (CCA) group.<sup>1</sup> Our approach is based on Interface Definition Language (IDL) techniques[6]. We have developed a Scientific Interface Definition Language (SIDL), as well as bindings to C and Fortran. We have also developed a SIDL compiler and run-time library support for reference counting, reflection, object management, and exception handling (Babel). Results from using Babel to call a standard numerical solver library (written in C) from C and Fortran show that the cost of using Babel is minimal, where as the savings in development time and the benefits of object-oriented development support for C and Fortran far outweigh the costs.

## 1 Introduction

Component technologies and component programming methodologies are beginning to work their way into the scientific community [2, 1, 11, 18] in the hopes of facilitating code reuse. One group developing a component architecture for high-performance, parallel computing is the Common Component Architecture group. An integral part to this component architecture is a mechanism for language interoperability. All components that operate within a component architecture should adhere to a standard behavior, which includes being able to easily interoperate with software written in other languages. With the proliferation of languages used for numerical simulation in recent years, like C, C++, Fortran 90, Fortran 77, Java, and Python, language interoperability can be a huge barrier to developing components, as well as developing reusable scientific applications and libraries.

---

<sup>1</sup> The CCA group consists of representatives from DOE laboratories and academia working towards the specification of a component architecture for high-performance scientific computing

Getting the many languages used in scientific computing to interoperate can be a difficult problem for developers. For both component and library developers, the choice of implementation language may severely limit the reuse of their software. Without language interoperability, users of components may be required to adopt the language of the component for future applications development, even though better alternatives may exist. If language interoperability is desired, component developers and users would be forced to write “glue” code that mediates data representations and calling mechanisms between languages. However, this approach is labor-intensive and in many cases does not provide seamless language integration across the various calling languages. Fortran 90 is a particular challenge for language interoperability, since Fortran 90 calling conventions vary widely from compiler to compiler.

### 1.1 Pairwise Approaches

There have been attempts at automatically generating glue code to support calls among a small set of targeted languages. For example, the SWIG package [3] reads C and C++ header files and generates glue code so that these routines can be called from scripting languages such as Python. Pyffle [19] is similar to SWIG except that it provides seamless integration of Python and C++. The problem with these approaches is that they either don't provide two-way interoperability between the scripting language and the target language, or all calls between languages must occur through the scripting environment, which makes them inappropriate for a high-performance component architecture. For instance, if a simulation package written in C wanted to call a numerical solver package written in Fortran 77 the package would have to make the call through the scripting environment. This would be much too inefficient for general use in scientific computing. These methods are not general enough to support a high-performance component architecture.

Foreign invocation libraries, such as *Java Native Interface* [14], have been used to handle interoperability between two targeted languages. For instance, the *Java Native Interface* defines a set of library routines that enables Java code to interoperate with applications and libraries written in C and C++. The problem with this type of approach is that given  $N$  languages,  $O(N^2)$  different software packages would be needed to get all the languages to interoperate. Again, this is not general enough to support a high-performance component architecture.

### 1.2 IDL Approach

One interoperability mechanism used successfully by the distributed systems and components community [16, 13, 17, 20] is based on the concept of an Interface Definition Language or IDL. The IDL is a new “language” that describes the calling interfaces to software packages written in standard programming languages such as C, Fortran, or Java. Given an IDL description of the interface, IDL compilers automatically generate the glue code necessary to call that software component from other programming languages.

This approach shows promise, however, current IDL implementations are not sufficient for specifying interfaces to single-program multiple-data (SPMD) type of components. First, standard IDLs such as those defined by CORBA and COM do not include basic scientific computing data types such as complex numbers or block style dynamic multidimensional arrays. Second, all of these approaches do not provide support for high-performance same address space function calls for all the programming languages needed in scientific computing. Our goal was to make the overhead of calls through the SIDL about as expensive as the invocation of a C++ virtual function. Third, some of these approaches don't have support for true multiple inheritance (e.g. COM does not support multiple inheritance and implementation inheritance is done with composition or aggregation, which can be computationally expensive or difficult to implement), and those that do have support use a limited object model (e.g. CORBA does not support method overriding and their implementation of multiple inheritance is prone to method name collisions). It is important that an IDL supports true multiple inheritance to allow specification of standards for numerical library interfaces, like the Equation Solver Interface (ESI) specification [10].

We have used an IDL approach for handling language interoperability in a scientific computing environment. We have developed a Scientific IDL (SIDL) as well as a run-time environment (Babel) that implements bindings to SIDL and provides support necessary for a component architecture, like reflection. Currently SIDL has bindings to C and Fortran 77. Babel implements those bindings on Solaris and AIX, with plans to port them to most major platforms. Preliminary efforts have shown that SIDL is expressive enough for scientific computing and that the binding implementations are fast.

This paper is organized as follows. Section 2 describes the features of SIDL that are necessary to support high-performance parallel computing. Section 3 describes the bindings of SIDL to C and Fortran 77, as well as Babel run-time environment, which includes a SIDL compiler and library support. Section 4 gives the results from wrapping a standard solver library with Babel and calling it from both C and Fortran. Finally, we conclude in Section 5 with an analysis of the lessons learned while wrapping *hypre* and identification of future research and additions to Babel.

## 2 Scientific Interface Definition Language

For an IDL approach to work in the scientific domain, the IDL must be sufficiently expressive to represent the abstractions and data types common in scientific computing, such as dynamic multidimensional arrays and complex numbers. Additionally, the IDL must have an object model that supports true multiple inheritance. This is necessary for satisfying the CCA component architecture specification as well as interface standardization efforts like those being implemented by the ESI. The IDL should also provide error handling mechanisms which are robust and efficient. Unfortunately, no standard IDL currently exists that supports all of these, since most IDLs have been designed for operating systems [7, 8]

or for distributed client-server computing in the business domain [13, 17, 20]. However, SIDL does borrow heavily from the CORBA IDL [17] and the Java programming language [12]. Some of the features SIDL provides are an object model similar to Java, language constructs necessary for scientific computing like complex numbers and dynamic multi-dimensional arrays, and an error handling mechanism that is a cross between Java and CORBA's exception handling mechanisms. Also, implicit constructs in SIDL allow SIDL implementation environments, like Babel, to provide reflection capabilities, which is a necessary feature for component architectures.

## 2.1 SIDL Object Model

Currently, interfaces and classes are the only two user defined types in SIDL. SIDL adopts the same object model as the Java programming language. The Java object model is advantageous because it is well defined, where other models, like those used in C++ and CORBA, are not as well defined. For instance, in C++, a class can inherit from multiple non-abstract classes. This poses a problem if any two or more of the parent classes have method(s) with the same signature. Java avoids this problem by only allowing single implementation inheritance and multiple interface inheritance.

All methods are equivalent in semantics to C++ virtual functions. Methods can be overridden by child classes, which means the methods in all the parent classes and interfaces, which have the same signature as the method in the child class, will be defined by that method in the child class. Methods can also be declared **abstract**, **final**, or **static**. An **abstract** method is purely declarative and has no implementation provided for it. When a method is declared **abstract**, the class also becomes **abstract**. All methods of interfaces are **abstract**. A **final** method is one which can not be overridden by child classes. We include the **final** construct to allow implementations of the SIDL bindings to perform optimizations by eliminating a lookup in a class's virtual function table. A **static** method is also **final**, with additional semantics. **Static** methods are invoked through a class, not an instance of a class. They are the closest thing to "global" methods in SIDL. We include the **static** construct to ease wrapping of non-object oriented language libraries and components.

Every class belongs to a nested package scope. Packages in SIDL are similar to namespaces in C++ and packages in Java. The package construct is used to create nested SIDL name space scopes. It is the only SIDL construct that creates a new name scope. Packages help prevent global naming collisions of classes and interfaces.

## 2.2 Scientific Data Types

Most IDL's, like those used in COM and CORBA, do not support all the types needed in scientific computing. For instance, both COM and CORBA's IDLs do not support complex numbers nor *block style*, *dynamic multidimensional* arrays. CORBA only supports static multidimensional arrays and sequences,

where COM only support ragged dynamic multidimensional arrays. In addition to the standard types like *int*, *char*, *bool*, *string*, and *double*, we have included *dcomplex*, *fcomplex*, and *array*. *dcomplex* is a complex number of type double. *fcomplex* is a complex number of type float. The *array* type has both a *type* specification and a *dimensions* specification. The *type* specification tells what type of elements the array contains and the *dimensions* specification tells how many dimensions are in the array. A SIDL *array* is the same as a Fortran block style array.

### 2.3 Exception Handling

Component architectures need robust mechanisms for error handling that can work across languages. For instance, COM requires all synchronous methods to return an error code and all asynchronous methods to return void. The mechanism for COM is not robust and requires a lot of run-time support to gain meaningful results, as with an exception mechanism found in Java. CORBA uses an exception mechanism where an environment variable is passed as the last argument in an argument list in a method and exceptions are set in that environment variable. We use a mechanism very similar to CORBA except that exceptions are not defined as structures, as they are in CORBA, but rather as objects, as they are in Java. All exceptions in SIDL are objects that inherit from *Throwable*. Also, we are exploring using a static environment variable, which would allow exceptions to be thrown without explicitly passing an environment variable as a method argument. Of course implementations of this model will have to be thread safe since they will be used in parallel applications.

### 2.4 Reflection

SIDL has constructs that allow support tools to implement reflection capabilities, which is necessary for components (e.g. CCA components). Recall that SIDL's object model is very similar to Java. SIDL also borrows its introspection capabilities from Java. For instance, like Java, all SIDL objects implicitly inherit from *Object*. *Object* has a method *getClass* which returns a *Class* object. This *Class* object contains information about a particular object's methods, fields, and constructors, which can be queried and invoked at run-time. Every object has a *Class* object associated with it that contains information on its methods, fields, and constructors. Given this, SIDL implementation tools, like Babel, can provide reflection capabilities by implementing SIDL's introspection specification.

## 3 Bindings and Implementation

This sections discusses the bindings of C and Fortran 77 to SIDL, as well as the implementation of those in the Babel run-time environment. This discussion is of only the more challenging aspects of developing the bindings and implementation. See [15] for a complete specification of SIDL and its bindings to C and Fortran 77.

### 3.1 Bindings to C and Fortran 77

Mapping SIDL onto C and Fortran 77 posed some interesting challenges. For instance, mapping SIDL objects into C and Fortran 77 objects was not altogether obvious since neither language has object oriented features. Also, mapping complex numbers to C as well as mapping the SIDL array syntax to the two languages, posed some challenges as well. Besides these, the mappings of SIDL to C and Fortran 77 were fairly straight forward.

For C, SIDL objects are mapped to opaque structure pointers. In Fortran an object is mapped to an integer. Of course a run-time environment that implements these bindings will have to provide library support that can translate an integer representation of an object to the actual object, in order to get access to that object's data and methods (this is done by the Babel run-time environment). A method is invoked by passing the reference to the object, whether it be an integer in Fortran or an opaque structure pointer in C, as the first argument in the argument list.

Complex numbers in C are mapped to a structure with two elements. The first element is the real part of the number and the second part is the imaginary part. Arrays are mapped to structures in C that contain three elements. The first element is a single dimensional array that contains the lower bounds for each of the dimensions. The second element is the same as the first, except it contains the upper bounds. The third element is a pointer to the data. In Fortran, arrays are simply mapped to the corresponding array representation in Fortran. Bounds for the dimensions need to be specified explicitly in Fortran, since Fortran does not have structures.

### 3.2 Implementing the Babel Run-Time Environment

Most of the effort in developing the Babel compiler and run-time was in implementing the object model, namely: the virtual function tables, the object lookup table, reference counting, dynamic type casting, the exception handling mechanism, and reflection capabilities. The Babel run-time is implemented in C and the compiler is written in Java, however, the "glue" code that is generated from the compiler is in C. All of the object support is contained in the "glue" code and the run-time library. For instance, every object has a skeleton associated with it. The skeleton contains the implementation of the object, including the virtual function table (which is implemented like a static C++ virtual function table), constructors, destructors, support for dynamic type casting, etc... The run-time library contains support for reference counting, the object lookup mechanisms (which is necessary for supporting objects in Fortran), and the exception handling mechanism. The reflection capability is supported through both the skeleton and the run-time library.

One of the goals while developing Babel was to make function calls made through Babel fast. We were able to limit C to C function calls to one extra function call and one lookup. Calls between C and Fortran 77 required two extra function calls and one lookup, and Fortran 77 to Fortran 77 calls require

three extra function calls and one lookup. The extra function calls between languages are needed to translate between the different signatures. Babel does take advantage of the `static` and `final` constructs in SIDL by eliminating a function table lookup to functions of those types.

#### 4 Results from Wrapping *hypre*

As a test case, we used Babel to create new interfaces for the *hypre* semicoursening multigrid solver (SMG) [4], a linear solver that is part of the *hypre* preconditioner library [5]. *hypre* is a library of parallel solvers for large, sparse linear systems being developed at Lawrence Livermore National Laboratory's Center for Applied Scientific Computing. The library currently consists of over 30,000 lines of C code, and it has 94 encapsulated user-interface functions. To test Babel we created a new interface (*hypre* is written in C, with a C interface provided by the authors) for both C and Fortran 77 using Babel, and ran similar test drivers using the two Babel generated interfaces and the original C interface already provided by the library. We then compared the results from all three.

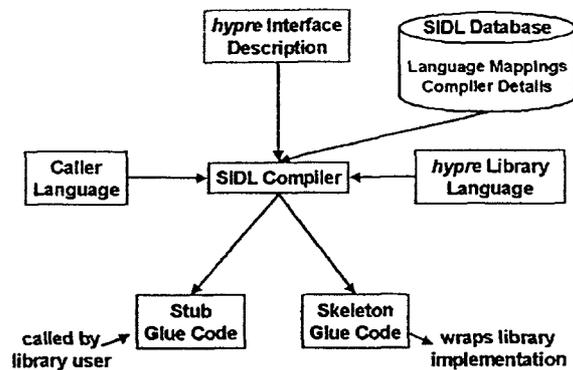


Fig. 1. Wrapping *hypre*

Wrapping *hypre* with Babel took three steps. The first step was to write a description of the existing interface in SIDL, which was done by two people, one who was familiar with SIDL and another who was familiar with the *hypre* library. The second step was to run the Babel compiler with the interface description as input to create all of the "glue-code" for each class (see Fig. 1). Since the signatures for the library functions were different from those in the virtual function tables, we also had to manually write the calls to the *hypre* functions into the library skeleton generated by the Babel compiler. This was a somewhat mundane task, but it required only one line of code per function, and it needed to be done only once, as the same skeleton code was used for both the C and Fortran (as well as for all other language bindings). Manually editing of

the skeleton code would not be necessary if the library used naming conventions and calling sequences that complied with the Babel specification (e.g., prepend every function call with an impl\_). Once the function calls were manually added, the new C interface was complete, and then the Fortran interface was created almost instantly by running the compiler once more with different options to create the Fortran stub code. The final step was to compile and link the drivers with the skeletons, stubs, and the *hypre* library.

We rewrote an existing SMG test driver to test the efficacy of the new interfaces. The driver uses SMG to solve Laplace's equation on a 3-D rectangular domain with a 7-point stencil. First, all calls in the driver to the *hypre* library were replaced with the new C interfaces that Babel created. Then we wrote a new Fortran driver that sets up exactly the same problem using the same algorithm and calls the same *hypre* functions via the new Fortran interface. Fig. 2 shows a portion of the *hypre* interface written in SIDL and 3 shows portions of both the C and Fortran drivers that call the *hypre* library through Babel.

```

package hypre {
  class stencil {
    stencil NewStencil(in int dim, in int size);
    int SetStencilElement(in int index, inout array<int> offset);
  };
  class grid {
    grid NewGrid(in mpi_com com, in int dimension);
    int SetGridExtents(inout array<int> ilower, inout array<int> iupper);
  };
  class vector {
    vector NewVector(in mpi_com com, in grid g, in stencil s);
    int SetVectorBoxValues(inout array<int> ilower,
      inout array<int> iupper, inout array<double> values);
    ...
  };
  class matrix { /* matrix member functions omitted in this figure */ };
  class smg_solver {
    int Setup(inout matrix A, inout vector b, inout vector x);
    int Solve(inout matrix A, inout vector b, inout vector x);
    ...
  };
};

```

Fig. 2. SIDL for *hypre*.

Both new drivers ran with no change in numerical results. We compared the efficiency of the new C and Fortran drivers to the original C driver. The drivers that used the Babel wrappers solved large problems both sequentially and in parallel on 216 processors, with no noticeable effect (less than 1%) on the speed of execution. The overhead added by Babel is negligible when compared to the overhead of the numerical kernel of the library.

In all, this took less than an afternoon to wrap and run *hypre* on both Solaris and AIX using both a C and Fortran 77 driver. To put this in perspective, there was an effort by the *hypre* team to wrap *hypre* by hand, making it callable from

C Test Code	Fortran 77 Test Code
<pre> hypr_vector b, x; hypr_matrix A; hypr_smg_solver solver; hypr_stencil s;  b = hypr_vector_NewVector(com, grid, s); ... x = hypr_vector_NewVector(com, grid, s); ... A = hypr_matrix_NewMatrix(com, grid, s); ...  solver = hypr_smg_solver_new(); hypr_smg_solver_SetMaxItr(solver, 10); hypr_smg_solver_Solve(solver, &amp;A, &amp;b, &amp;x); hypr_smg_solver_Finalize(solver); </pre>	<pre> integer b, x integer A integer solver integer s  b = hypr_vector_NewVector(com, grid, s) ... x = hypr_vector_NewVector(com, grid, s) ... A = hypr_matrix_NewMatrix(com, grid, s) ...  solver = hypr_smg_solver_new() hypr_smg_solver_SetMaxItr(solver, 10) hypr_smg_solver_Solve(solver, A, b, x) hypr_smg_solver_Finalize(solver) </pre>

Fig. 3. Sample test code.

Fortran on a Solaris platform, that took over one week for one person to do. Even after they finished wrapping it, they had to redo the effort when they ported it to another platform.

## 5 Lessons Learned and Future Work

Our experience using Babel to create new interfaces for *hypr* shows that Babel is an effective tool to support language interoperability for high-performance, parallel scientific computing. While it is not difficult to use for an existing library such as *hypr*, Babel can be easier to use if a library, or component, is designed and written from the beginning with Babel naming conventions in mind. Calls to the library, or component, will also be faster, if these conventions are followed, since there will be one less function call. Calls through Babel can be streamlined even further by declaring methods `final` or `static`, where possible. This will eliminate a virtual function table lookup. Also, developers who use non-object oriented languages can take advantage of the object support that Babel provides.

In the future we will develop bindings for C++, Java, Fortran 90, and Python and implement those bindings in Babel. We will also explore various component composition and introspection models for scientific computing, in conjunction with the CCA, and develop the appropriate library implementations in Babel to support them.

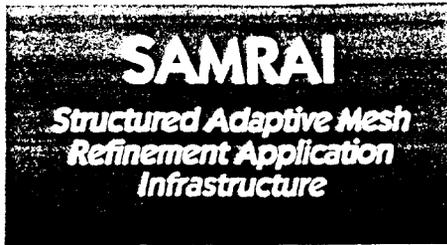
## References

1. R. Armstrong, D. Gannon, A. Geist, K. Keahey, S. Kohn, L. McInnes, S. Parker, and B. Smolinski, *Toward a common component architecture for high performance scientific computing*, 1999.

2. S. Balay, B. Gropp, L. C. McInnes, and B. Smith, *A microkernel design for component-based numerical software systems*, in Proceedings of the First Workshop on Object Oriented Methods for Inter-operable Scientific and Engineering Computing, 1998.
3. D. M. Beazley and P. S. Lomdahl, *Building flexible large-scale scientific computing applications with scripting languages*, in The 8th SIAM Conference on Parallel Processing for Scientific Computing, Minneapolis, MN, March 1997.
4. P. Brown, R. Falgout, and J. Jones, *Semicoarsening multigrid on distributed memory machines*, in SIAM Journal on Scientific Computing special issue on the Fifth Copper Mountain Conference on Iterative Methods, 1999.
5. E. Chow, A. Cleary, and R. Falgout, *Design of the hypre preconditioner library*, in Proceedings of the First Workshop on Object Oriented Methods for Inter-operable Scientific and Engineering Computing, 1998.
6. A. Cleary, S. Kohn, S. Smith, and B. Smolinski, *Language interoperability mechanisms for high-performance applications*, in Proceedings of the First Workshop on Object Oriented Methods for Inter-operable Scientific and Engineering Computing, 1998.
7. G. Eddon and H. Eddon, *Inside Distributed COM*, Microsoft Press, Redmond, WA, 1998.
8. E. Eide, J. Lepreau, and J. L. Simister, *Flexible and optimized IDL compilation for distributed applications*, in Proceedings of the Fourth Workshop on Languages, Compilers, and Run-time Systems for Scalable Computers, 1998.
9. R. Englander, *Developing Java Beans*, O'Reilly, 1997.
10. Equations Solver Interface Forum. See <http://z.ca.sandia.gov/esi/>.
11. D. Gannon, R. Bramley, T. Stuckey, J. Villacis, J. Balasubramanian, E. Akman, F. Breg, S. Diwan, and M. Govindaraju, *Component architectures for distributed scientific problem solving*, 1998.
12. J. Gosling and K. Arnold, *The Java Programming Language*, Addison-Wesley Publishing Company, Inc., Menlo Park, CA, 1996.
13. B. Janssen, M. Spreitzer, D. Larner, and C. Jacobi, *ILU Reference Manual*, Xerox Corporation, November 1997. Available at <ftp://ftp.parc.xerox.com/pub/ilu/ilu.html>.
14. JAVA SOFT, *Java Native Interface Specification*, May 1997.
15. S. Kohn and B. Smolinski, *Component interoperability architecture: A proposal to the common component architecture forum. in preparation*, 1999.
16. MICROSOFT CORPORATION, *Component Object Model Specification (Version 0.9)*, October 1995. Available at <http://www.microsoft.com/oledev/olecom/title.html>.
17. OBJECT MANAGEMENT GROUP, *The Common Object Request Broker: Architecture and Specification*, February 1998. Available at <http://www.omg.org/corba>.
18. S. Parker, D. Beazley, and C. Johnson, *The SCIRun Computational Steering Software System*, E. Arge, A.M. Bruaset, and H.P. Langtangen (Eds.), Modern Software Tools in Scientific Computing, Birkhauser Press, 1997.
19. Paul Dubois, personal communication. See [http://xfiles.llnl.gov/CXX\\_Objects/cxx.htm](http://xfiles.llnl.gov/CXX_Objects/cxx.htm).
20. J. Shirley, W. Hu, and D. Magid, *Guide to Writing DCE Applications*, O'Reilly & Associates, Inc., Sebastopol, CA, 1994.

This article was processed using the  $\LaTeX$  macro package with LLNCS style

Work performed under DOE at LLNL under contract No. W-7405-Eng-48.



### Technology

SAMRAI is an object-oriented software framework for structured adaptive mesh refinement (AMR) research. SAMRAI provides computational scientists with general and extensible support for rapid prototyping and development of parallel structured AMR applications. The primary goal of the SAMRAI effort is to facilitate numerical method and solution algorithm development for AMR applications that require high-performance computing hardware.

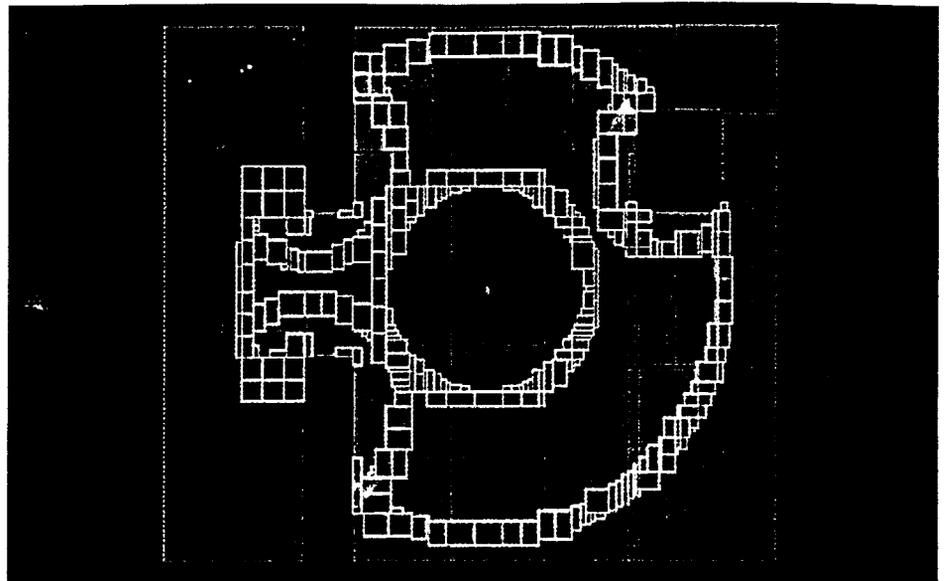


Figure 1. Adaptive mesh refinement concentrates computational effort in areas of interest, such as near the shock fronts in this hydrodynamics simulation.

Developments in high-performance computing hardware make it possible to simulate large three-dimensional problems that model increasingly complex chemical and physical processes. However, the numerical resolution required to capture the phenomena represented by mathematical models still makes such computations very expensive. In many interesting science and engineering applications, the most important features of the simulation occur in localized regions of the computational domain. Uniformly fine computational meshes that resolve the local phenomena may be unnecessarily fine outside the regions of interest. As a result, such uniform grid simulations can be inefficient or even prohibitively expensive.

Structured AMR provides a systematic way to focus computational resources (CPU time and memory) by employing varying degrees of local spatial and temporal resolution. Thus, AMR is an important technology needed to support large-scale, physically and numerically well-resolved, three-dimensional simulations.

### Emerging AMR Application Domains

Structured AMR has proved to be a useful simulation technology for many fluid dynamics applications (Figure 1). SAMRAI is being developed as a substantial generalization of existing AMR technology. In addition to supporting more traditional AMR applications, SAMRAI is designed to explore new problem areas and new AMR solution algorithms. New applications include, but are not limited to, problems modeled by coupled systems of partial differential equations that exhibit complicated combinations of hyperbolic, elliptic, and parabolic behavior (such as radiation hydrodynamics, flow and transport in porous media, combustion, and reactive transport), neutron transport, hybrid models that combine discrete and continuum numerical models, and Arbitrary Lagrangian-Eulerian (ALE) integration methods.

The application of structured AMR to such problems gives rise to many interesting algorithmic, numerical, and computer science research questions. These issues are related to mathematical model approximations in an AMR

setting, adaptive integration methods, load balancing for distributed memory parallel computing, and software framework design and development. SAMRAI currently supports a variety of computational science projects. Through collaborations with researchers at DOE laboratories and universities, we are investigating the application of AMR technology to some of the application domains mentioned previously.

### The SAMRAI Framework

Numerical algorithms for AMR problems are complex and require a substantial amount of software infrastructure. However, many software components used in AMR codes are similar across a broad spectrum of applications. These common components can be incorporated into a single general-purpose application framework. SAMRAI is being developed for two purposes: first, to codify existing AMR software support into a flexible, extensible, parallel development framework; and second, to apply structured AMR technology to new problem domains and to develop alternative

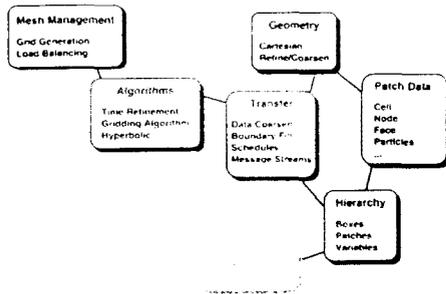


Figure 2. The SAMRAI framework consists of a collection of software components that greatly simplify the development of structured AMR applications.

adaptive solution algorithms for more traditional applications.

Since AMR applications involve complex, dynamically changing data communication patterns, it is particularly challenging on parallel computers. From its inception, SAMRAI was designed to facilitate sophisticated numerical algorithm development in a parallel computing environment. Three main software design points SAMRAI addresses are: first, support for a wide range of complex data structures on an adaptively-refined mesh, including arbitrary user-defined types; second, algorithmic flexibility, extensibility, and software reuse; and third, general software support for parallel application development.

SAMRAI facilitates rapid prototyping of various application code implementation alternatives by freeing developers from low-level data structure and algorithm management. An application developer views SAMRAI as a collection of software packages and classes that may be combined in complex ways to build an application (Figure 2). We use object-oriented design techniques so that many fundamental SAMRAI components may be enhanced without changing the underlying framework source code or re-compiling the library. In particular, software components may be specialized and extended through C++ class derivation. We are also addressing

interoperability issues that will allow applications built using SAMRAI to take advantage of other software packages, such as linear and nonlinear solver libraries, including PETSc, KINSOL, and *hypra*.

SAMRAI's framework structure provides flexibility to explore a wide range of AMR applications. Its design helps to reduce code development, encourages interoperability in application software, and simplifies the learning curve for new adaptive computational methods. Finally, SAMRAI provides robust support for parallelism to insulate communication operations from application code without impeding performance.

### SAMRAI Applications and Framework Validation

In collaboration with researchers, we are using SAMRAI across a diverse range of applications. We are collaborating with academic researchers to employ SAMRAI in an application that couples fluid dynamics models (including turbulence and reactive chemistry) to container dynamics models to study firespread problems. Another SAMRAI application code combines plasma fluid simulation with a model for laser light propagation. Other projects include simulation of solar winds in the Earth's magnetosphere, radiation-hydrodynamics using ALE integration methods, shear band formation in granular materials, flow and transport in porous media, neutron transport, and multi-physics problems that combine continuum and discrete particle models.

Each application emphasizes the combination of different numerical models and solution techniques in the context of structured AMR. For example, the firespread application uses both implicit and explicit integration methods for fluid calculations that are coupled to particle-like methods to model solid

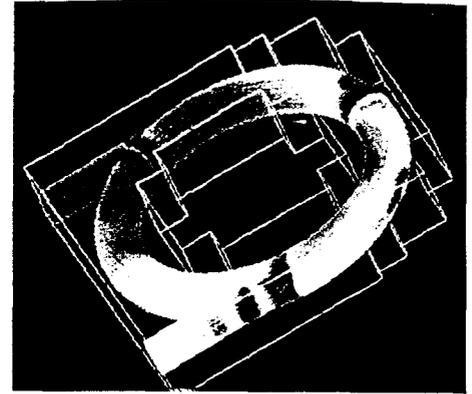


Figure 3. Embedded boundary capabilities enable structured AMR applications to capture complex geometry in an efficient manner.

containers. The solar wind problem combines magneto-hydrodynamics equations with a dipole field model of the Earth's ionosphere.

In addition to producing interesting computational science research, these applications serve as a validation of the SAMRAI software architecture, as they require substantial data structure and algorithmic flexibility. The applications produce a variety of simulation data on an adaptive mesh, including particles and embedded interfaces (Figure 3), as well as various forms of array-based data. Also, these complex solution algorithms combine sophisticated numerical methods to treat different aspects of each problem. Finally, incorporation of linear and nonlinear solution software tests SAMRAI's ability to interoperate with independently developed solver libraries.

For additional information about the SAMRAI project, visit the Web site at <http://www.llnl.gov/CASC/SAMRAI> or contact:

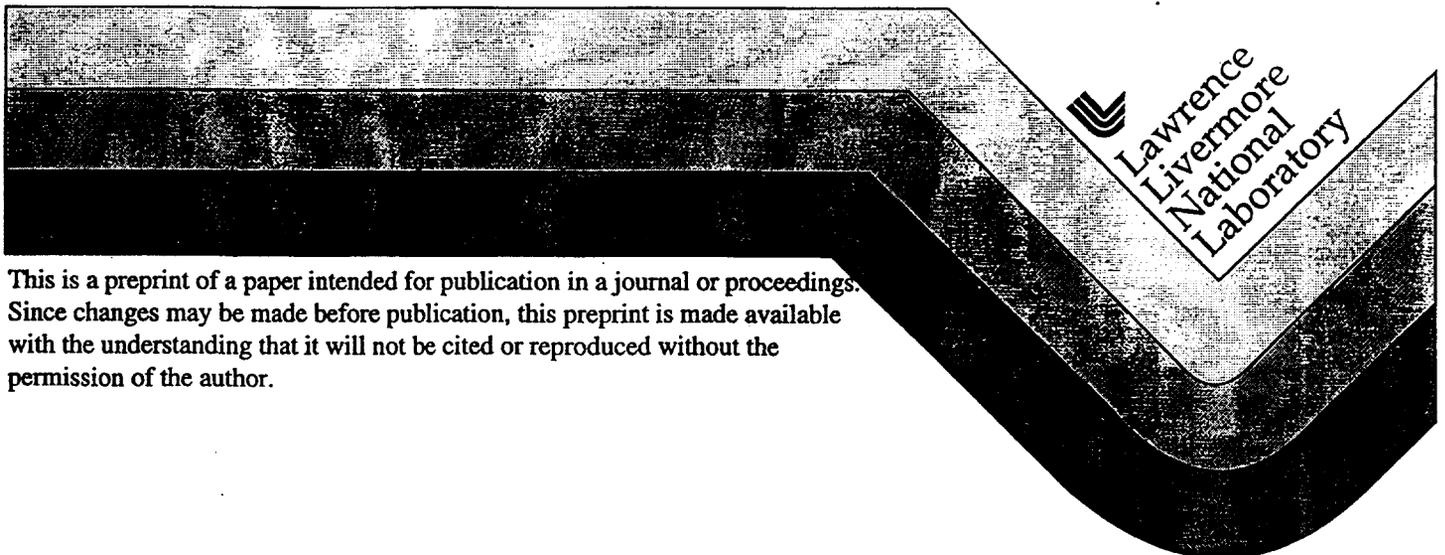
Richard Hornung,  
(925) 422-5097, [hornung@llnl.gov](mailto:hornung@llnl.gov), or  
Scott Kohn,  
(925)422-4022, [skohn@llnl.gov](mailto:skohn@llnl.gov).

## The Use Of Object-Oriented Design Patterns In the SAMRAI Structured AMR Framework

R. D. Hornung  
S. R. Kohn

This paper was prepared for submittal to  
Society for Industrial & Applied Mathematics  
Workshop on Object- Oriented Methods for  
Interoperable Scientific and Engineering Computing  
Yorktown Heights, NY  
October 21-23, 1998

October 17, 1998



#### DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

# The Use of Object-Oriented Design Patterns in the SAMRAI Structured AMR Framework\*

Richard D. Hornung<sup>†</sup>

Scott R. Kohn<sup>†</sup>

## Abstract

We describe the use of object-oriented design patterns in the implementation of a flexible structured adaptive mesh refinement software framework called SAMRAI. We present five common patterns—*Smart Pointers*, *Singleton*, *Abstract Factory*, *Strategy*, and *Chain of Responsibility*—that have greatly simplified framework development. These design patterns have enabled the decomposition of complex algorithms into smaller, more manageable, decoupled components that may be reused across a variety of applications.

## 1 Introduction

The design and implementation of quality, high-performance numerical software frameworks is difficult. Framework designers must address issues of algorithm complexity, evolving requirements in a research environment, and software reuse within the targeted application domain. Many modern numerical algorithms, such as structured adaptive mesh refinement methods [3, 4], consist of many complex numerical components involving sophisticated time integration methods, various geometry descriptions, time interpolation, spatial refinement and coarsening, and linear and nonlinear solvers. These numerical components interact in complex ways that must be captured in the design of the software framework. Finally, numerical frameworks are usually developed in tandem with research projects in algorithms and applications; thus, the framework software must be designed to evolve as computational scientists improve their understanding of application domains and the associated numerical methods.

In this paper, we address some of these design issues in the context of a parallel structured adaptive mesh refinement (SAMR) framework called SAMRAI. Object-oriented techniques and design patterns [8] have been valuable tools for the high-level organization of the SAMRAI software architecture. They have enabled us to isolate various functional parts of complex algorithms into different framework components so that applications can be built from smaller algorithmic “building blocks.” As a result, we provide a flexible software framework that simplifies the management of inherently complex SAMR algorithms and is being applied to diverse SAMR applications.

This paper is organized as follows. We begin with a brief overview of the SAMRAI framework and the basic SAMR methodology. Section 3 describes five different design

---

\*This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract number W-7405-Eng-48.

<sup>†</sup>Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94551, <http://www.llnl.gov/CASC/SAMRAI>, [hornung@llnl.gov](mailto:hornung@llnl.gov), [skohn@llnl.gov](mailto:skohn@llnl.gov).

patterns—*Smart Pointers* [7], *Singleton*, *Abstract Factory*, *Strategy*, and *Chain of Responsibility* [8]—used in the SAMRAI framework. Finally, Section 4 discusses the usefulness of design patterns and object-oriented techniques within the SAMRAI framework and for the general computational science community.

## 2 The SAMRAI Framework

Structured adaptive mesh refinement has shown *great potential* as a numerical simulation methodology for a variety of applications in computational fluid dynamics [2, 1, 12], laser-plasma interactions [6], radiation transport [11], porous media [10], and materials [9, 13, 14]. However, SAMR methods are not widely used by the scientific computing community. The primary reason for this is that SAMR codes are complex and require a substantial amount of software infrastructure to support productive application development. Fortunately, many software components are common across diverse application domains and may be incorporated into a general-purpose framework that supports a broad range of applications.

SAMRAI is a C++ object-oriented framework that provides computational scientists with general and extensible software support for prototyping and developing parallel SAMR applications. The primary goal of the SAMRAI effort is to explore the use of SAMR technology in new problem domains and to develop new numerical and algorithmic approaches for more traditional SAMR applications. SAMRAI provides an overarching software framework that orchestrates the various processes involved in a complex numerical simulation. SAMR algorithms can usually be decomposed into smaller, simpler constituent parts such as algorithmic components, data structures, and numerical routines. In the process of building a new application with SAMRAI, computational scientists select the appropriate numerical and algorithm components from the framework and supply only those operations that are specific to their application. Thus, the computational scientist leverages a large simulation code base and only specializes certain components as needed for his application.

A full description of SAMR algorithms is well beyond the scope of this paper. However, we will provide a brief overview of the basic SAMR approach as an aid to understanding the algorithmic and software issues in the remainder of this paper. The SAMR approach, introduced by Berger, Oliger, and Colella [3, 4], represents simulation data using a hierarchy of nested levels of spatial and temporal mesh refinement. This hierarchy dynamically adapts to follow interesting features in the evolving simulation and focus computer resources in these localized portions of the computational domain.

A SAMR hierarchy consists of a number of levels. All computational cells at a particular level in the hierarchy represent the same mesh resolution. Each level consists of a collection of *patches*, each of which is a logically rectangular collection of computational cells. A patch contains data that represent simulation quantities in the region of the simulation domain defined by the patch region. The level with the coarsest mesh resolution defines an abstract, global integer index space. Then, each successively finer level is a refinement of a portion of the next coarser index space. The organization of the computational mesh into a hierarchy of levels of patches allows data communication and computation to be expressed as geometrically-simple, efficient operations. Consequently, the SAMR methodology is used to construct application codes from a set of computational tasks, each of which is defined in terms of operations on mesh patches, organized in a highly structured fashion.

In the remainder of this paper, we discuss object-oriented techniques used to implement two of SAMRAI's design goals. First, SAMRAI must support a wide range of complex data

structures on SAMR patches, including arbitrary user-defined types. Second, SAMRAI must provide flexible and extensible algorithm support for a variety of SAMR applications. One important design constraint is that SAMRAI must enable new applications development and support new user-defined data types without requiring changes to the underlying framework source code or recompilation of the libraries.

### 3 Design Patterns in SAMRAI

Design patterns are specific solutions to common, recurring software engineering problems. Each pattern codifies a general solution technique by providing a problem description, the solution pattern, and a list of consequences resulting from the application of the pattern. In practical terms, design patterns describe the configuration of a small set of objects whose cooperative behavior solves a software design problem. There are several useful books that describe design patterns [5, 8, 15]; our discussion follows that of Gamma *et al.* [8] most closely.

We have found design patterns to be very useful in solving some important design problems during the construction of the SAMRAI software architecture. Some of these patterns are covered in detail in the following five sections. We begin each section with a discussion of a design problem encountered in the SAMRAI software framework. We then describe the design pattern selected to solve that particular design problem and the consequences of that decision.

Section 3.1 describes the *Smart Pointer* pattern that simplifies the management of dynamically allocated memory and provides safe dynamic type casting. The *Singleton* pattern (Section 3.2) defines a single point of contact for objects shared among various components. The *Abstract Factory* creational pattern (Section 3.3) enables SAMRAI to support new user-defined patch data types without requiring modifications to the framework. Finally, the *Strategy* (Section 3.4) and *Chain of Responsibility* (Section 3.5) behavioral patterns are used in SAMRAI to decouple various framework components and thus obtain greater reuse of fundamental algorithm pieces.

#### 3.1 Smart Pointers

In this section, we describe two typical problems in the SAMRAI framework that are solved through the use of *Smart Pointer* [7] techniques: (1) safe dynamic type casting and (2) memory management for shared objects. The need for safe dynamic type casting is illustrated by the following example. As described in Section 3.3, all SAMRAI patch data types share a common base class called `PatchData`:

```
class PatchData : public ... {
    void copy(const PatchData& source) = 0;
    ...
};
```

The concrete data types that are instantiated on SAMRAI patches—such as `CellData<double>` or `EmbeddedBoundaryData`—inherit the signature for `copy()` declared in `PatchData`. However, concrete classes often require the type of the `copy()` argument to be the same as the concrete class, not any arbitrary `PatchData` object. For example, it would probably not make sense to copy `EmbeddedBoundaryData` into `CellData<double>`. Unfortunately, there is no way to enforce this through the C++ type system at compile-time. Although templates are often used in similar cases to ensure type safety, they are not sufficient for complex applications that must access data through abstract base classes.

Run-time type safety can be achieved through the use of run-time dynamic type casting. In this case, dynamic type casting of the argument source within the `copy()` implementation returns a pointer to the object if the cast is valid and `NULL` otherwise. Although dynamic type-safe casting is part of the C++ standard, it is not yet supported by all C++ compilers.

Another common problem solved through *Smart Pointers* is memory management for shared objects. In this case, many framework objects maintain pointers to a shared object instance that must be deallocated when all references to it disappear. Since ownership of this shared object cannot be uniquely established, the application cannot easily determine when to deallocate it. For example, SAMRAI patches typically share a pointer to a patch descriptor object. Moreover, patches are created and destroyed dynamically during mesh refinement. The memory allocation problem is solved with reference counting smart pointers that track the number of references to an object and then delete the pointed-to object when the number of references decrements to zero.

**3.1.1 Pattern Description** The *Smart Pointer* pattern is a common C++ pattern [7]. It consists of two parts: a templated `Pointer` class that manages the object reference counting and a collection of classes that support run-time safe type casting. All pointed-to objects are required to inherit from a common base class and provide a small number of functions to implement the type conversions.

**3.1.2 Consequences** The use of the *Smart Pointer* pattern within SAMRAI has greatly simplified the management of dynamic memory allocation; multiple objects may share pointers to the same object and the smart pointers guarantee that there will be no memory leaks. The type-safe dynamic casting ensures that type errors will be caught at run-time.

The primary disadvantage of the *Smart Pointer* approach is that it introduces a common base class for all pointed-to classes. While not a burden when writing new code, it is esthetically unappealing to force otherwise unrelated classes to inherit from a common base class, since it introduces extraneous coupling in the software architecture.

## 3.2 Singleton Classes

Many classes in the SAMRAI framework may be instantiated only once, with that single instance shared by various framework components. For example, a `VariableDatabase` object contains information about the variables used in a computational simulation (e.g., pressure, density, or velocity). The database must be accessible to all algorithm components to extract information about the variables and their roles in the simulation. Traditionally, such shared objects were implemented using global variables; however, global variables do not ensure only one instance of a class and they do not allow extension by subclassing. Instead, we implement shared objects such as `VariableDatabase` using the *Singleton* creational pattern as described in Gamma *et al.* [8]

**3.2.1 Pattern Description** The *Singleton* pattern ensures that a class will have only one instance and provides global, well-defined access to that instance. The class may be extended through inheritance. Then, clients may use the subclass object without changes to their code.

In SAMRAI, the `VariableDatabase` encapsulates its single instance and maintains strict controls over access to this instance. It declares a `getDatabase()` static member function that returns a pointer to the single database instance. In addition, the constructor and destructor of the class are protected to ensure that only the database and its subclasses

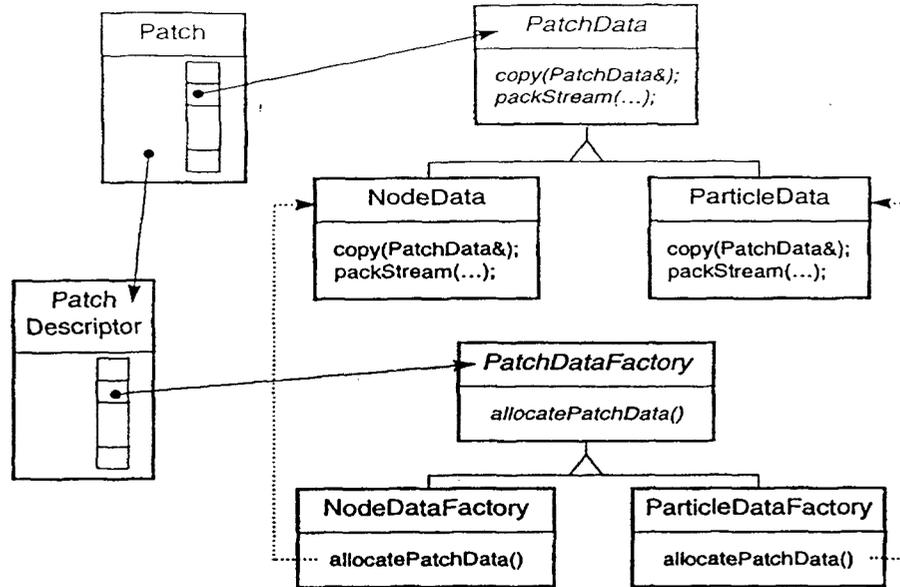


FIG. 1. The Abstract Factory pattern manages the allocation of data for the SAMRAI patch hierarchy. As illustrated by the dotted lines, subclasses of PatchData are created by associated subclasses of PatchDataFactory. This diagram follows the OMT (Object Modeling Technique) notation [8].

may create instances of the database.

**3.2.2 Consequences** The *Singleton* pattern provides a more flexible alternative to the use of global variables. The name space remains cleaner and applications may use extensions of a basic singleton object, even at run-time. A singleton can be extended through standard class derivation and any client can use the subclass without needing changes in its own code.

### 3.3 Abstract Factory

Recall that one of the primary considerations in the design of SAMRAI was the need to support complex user-defined data on an SAMR patch hierarchy. The patches in an SAMR application may contain data such as cell-centered arrays of doubles, node-centered arrays of integers, or user-defined collections of particles. These patch data types are manipulated by the SAMRAI framework, which manages allocation, deallocation, data copying, and marshaling and unmarshaling of data for communication between processors.

We believe that SAMRAI users should not modify the framework software or recompile the libraries to add new data types, as such practices violate sound software engineering principles. Thus, the framework cannot know the concrete class types for user-defined patch data, since these classes may be designed and implemented long after the framework has been compiled. In this case, how can the framework allocate user-defined data? Clearly, SAMRAI cannot execute new for concrete types that do not exist at compile-time. The solution to this problem is the *Abstract Factory* creational pattern.

**3.3.1 Pattern Description** The *Abstract Factory* pattern defines an approach for creating families of related objects without specifying their concrete classes [8]. This

pattern does so through two related inheritance hierarchies. The first hierarchy is rooted in an *abstract product* class that declares the interface for all objects to be created by the pattern. These product objects are created by *factory* objects in a second hierarchy.

Figure 1 illustrates how this pattern is implemented in the SAMRAI framework. The SAMRAI Patch is a container class for all patch data types that exist in some rectangular region of index space. All patch data types inherit from an abstract PatchData class and define a set of required routines such as copy() and packStream() (used for interprocessor communication). Each Patch has a smart pointer to a PatchDescriptor that contains the factory objects needed to make the concrete patch data. Then, to create an instance of a PatchData object, the Patch consults the PatchDescriptor and asks the appropriate PatchDataFactory to allocate a PatchData instance. In particular, the allocatePatchData() function returns the concrete PatchData instance.

**3.3.2 Consequences** The *Abstract Factory* pattern separates concrete object creation and declaration by encapsulating the responsibility for creating product objects. Use of this pattern enhances software flexibility and extensibility since concrete product classes (such as NodeData in Figure 1) never appear in the framework code. Thus, new product classes can be added after the framework has been compiled and archived into a library.

There are two drawbacks to *Abstract Factory* pattern. First, every new product class requires the definition of two new classes—the product class and the factory class. Second, some form of dynamic safe type casting is needed to obtain concrete class references. For example, although it is sufficient for the Patch container class to manipulate data as abstract PatchData objects, user-defined numerical routines will need to extract data from the patch and process that data using the concrete class interface. The cast from abstract product to concrete product requires some form of run-time type checking such as that described in Section 3.1.

### 3.4 The Strategy Pattern

SAMR applications involve sophisticated algorithms that can be decomposed into smaller constituent parts. These parts include algorithms for sequentially advancing a set of SAMR patch levels, integrating single patch levels, dynamically changing the mesh, and numerical routines defined on individual patches. A primary goal of SAMRAI is to provide a flexible algorithmic framework that encapsulates components such as these so that they may be reused in different SAMR applications when appropriate.

Developing a flexible algorithmic framework is difficult. The most important research challenge is discovering how complex algorithms may be factored into their constituent parts. Then, the specific behavior of each component must be determined and appropriate interfaces must be defined between the different pieces. Ideally, each individual algorithmic part may be replaced or enhanced without adversely influencing the behavior of the other components. If this separation is attained, it is relatively easy to combine existing software components to construct a complete SAMR algorithm. While we are still grappling with these issues in the development of SAMRAI, we believe that the approach outlined here demonstrates substantial progress.

The *Strategy* design pattern is the primary object-oriented design technique that we use to encapsulate algorithms and define reusable interfaces between software components. Next, we illustrate our use of this pattern by describing the decomposition of a standard SAMR algorithm into its primary components.

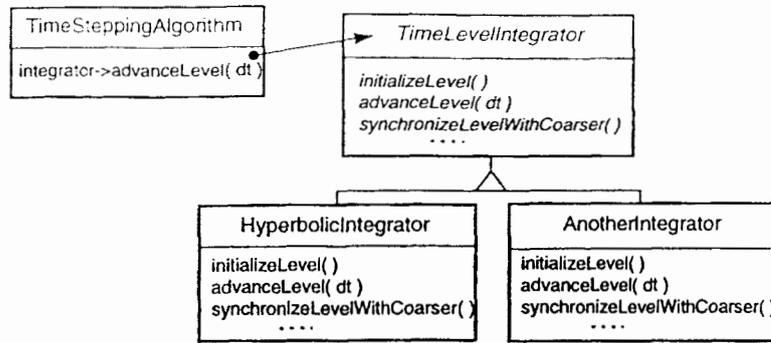


FIG. 2. SAMRAI uses Strategy to define a family of time-dependent integration algorithms.

**3.4.1 Pattern Description** The intent of the *Strategy* pattern is to define and encapsulate families of algorithmic components to make them interchangeable through common interfaces. Consequently, this pattern is well-suited to our concerns. An concise example of the basic form of the *Strategy* pattern is illustrated in Figure 2.

In SAMRAI, a `TimeSteppingAlgorithm` class controls a sequence of timesteps that advances the levels in an SAMR hierarchy. While this class is fairly general, the routines that advance data on the individual levels are specific to each application. When a `TimeSteppingAlgorithm` object is created, it is configured with a suitable level integration algorithm object. The level integration class may be supplied by the framework; for example, the `HyperbolicIntegrator` class is provided for systems of hyperbolic conservation laws. Otherwise, another integrator class must be implemented (e.g., `AnotherIntegrator`). Each level integrator class is derived from the `TimeLevelIntegrator` abstract base class and must satisfy the interface defined by that class. The `TimeSteppingAlgorithm` object maintains a pointer to the abstract type. Thus, it knows nothing of the specific, concrete level integration process.

Figure 3 shows multiple *Strategy* patterns, including a particular instance of the pattern in Figure 2, combined to form a complex algorithm from simpler components. The configuration represents a common SAMR algorithm for treating hydrodynamics applications, such as the Euler equations of gas dynamics, with explicit timestepping [3, 4].

At the top algorithmic level, the `TimeSteppingAlgorithm` class controls the overall SAMR scheme. It is configured with `HyperbolicIntegrator` and `RichardsonExtrapolation` objects, which supply routines to advance the data and dynamically adjust the mesh, respectively. Consistent with the *Strategy* pattern, the timestepping algorithm knows only the abstract types `TimeLevelIntegrator` and `GriddingAlgorithm`.

The *Strategy* pattern is repeated in the design of `RichardsonExtrapolation` and `HyperbolicIntegrator`. Concrete subclasses of `MeshGenerator` and `LoadBalancer` (not shown) provide routines that create box regions and load balance the patches on a new patch level. The `EulerPatchModel` class supplies numerical routines for the Euler equations on a single patch in the mesh hierarchy. Both `HyperbolicIntegrator` and `RichardsonExtrapolation` invoke functions in `EulerPatchModel` (e.g., numerical flux computation, conservative difference, select cells for refinement, etc.), but they are independent of the specific routines. That is, the `HyperbolicIntegrator` has a pointer to `HyperbolicPatchModel` and `RichardsonExtrapolation` has a pointer to

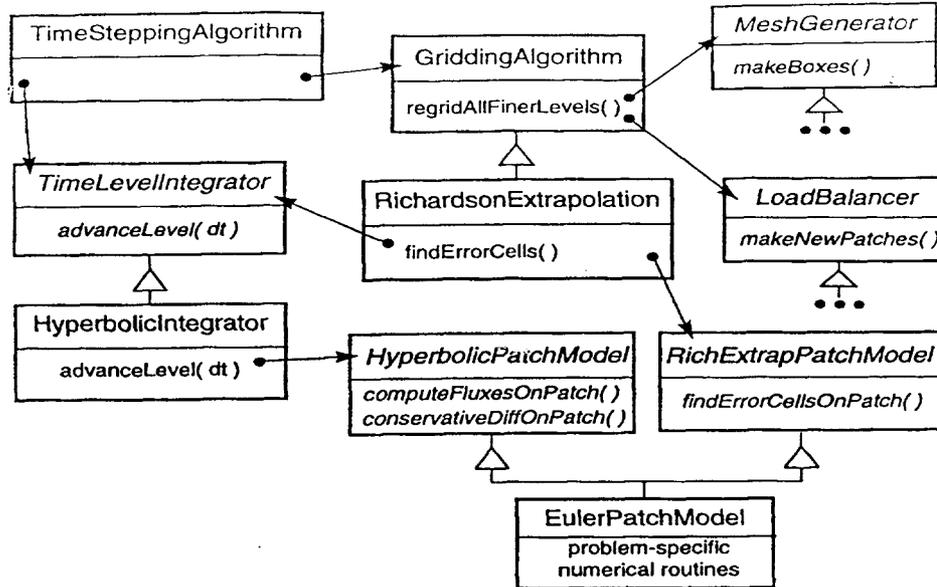


FIG. 3. Multiple instances of the Strategy pattern are combined in SAMRAI to build a complex AMR solution from simpler components.

RichExtrapPatchModel. The EulerPatchModel class, derived from both of these abstract base classes, implements functions declared in both of their interfaces.

**3.4.2 Consequences** The *Strategy* pattern provides a useful degree of algorithmic encapsulation in SAMRAI. Using common interfaces to characterize families of related algorithmic components, a system may be configured to perform a wide range of behaviors. This type of “plug-and-play” interoperability is advantageous for several reasons. First, it frees application programmers from unnecessary, redundant code implementation and reduces development time. Second, it promotes the exploration of different algorithmic choices within a single application. Third, it increases software reuse within the framework, which facilitates testing, maintenance, and extensibility of the architecture.

The encapsulation forced by the *Strategy* pattern is a valuable alternative to large, overly-complex classes that can occur through the abuse of inheritance. For instance, the design in Figure 2 could have been implemented by inheriting from TimeSteppingAlgorithm directly. The result would be several larger, more complicated classes that differ in level integration procedures, but have much timestepping code in common. Although decoupling the algorithm components slightly increases function call overheads, the cost is negligible at the high algorithmic level at which we use the *Strategy* pattern.

### 3.5 Chain of Responsibility

Data motion between SAMR hierarchy patches requires time interpolation, coarsening, and refinement operators that depend on problem geometry, the type of patch data, and the centering of patch data. The SAMRAI parallel communication routines are defined in terms of abstract operator and geometry base classes to decouple them from the details of the particular geometry or concrete operators used in an application. The association between

a patch data type and its concrete operators is managed through the SAMRAI geometry classes, which are responsible for cataloging the operators for a particular patch data type.

As users define new patch data types for their applications, they must also provide the required operators for these types. However, the SAMRAI geometry classes cannot know the concrete types of these new operators, since they were defined after the compilation of the SAMRAI framework. Thus, the geometry classes require an extensible lookup mechanism that allows the definition of new operators for user-defined patch data types. This particular design problem is solved by the *Chain of Responsibility* design pattern.

**3.5.1 Pattern Description** The *Chain of Responsibility* pattern avoids coupling the sender of a request to any potential request receiver by giving multiple objects handlers an opportunity to handle the request. Our implementation of this pattern follows Gamma [8].

To obtain an operator, an algorithm object queries the geometry object for the operators associated with a particular patch data type. The geometry object passes the request to the chain of handlers it owns. The request is forwarded along the chain until the correct operator handler is found. This handler then returns a pointer to the desired operator. The correct operator handler is found when the patch data type of the request matches the patch data type of the handler, where the type equality is determined using the dynamic casting facilities described in Section 3.1.

**3.5.2 Consequences** There are several advantages to using the *Chain of Responsibility* pattern for the operator lookup. First, this pattern reduces the coupling between patch data types, operators, and the algorithms that use them since these objects have no explicit knowledge of each other's concrete types. The same mechanism may be used for arbitrary patch data types and operators without changing any of the algorithm code. Second, the system is sufficiently flexible so that new concrete operator handlers (thus, new operators) may be added to the chain at run time. In particular, there is no need to use conditional statements or enumerated types that cannot be extended without recompilation. Third, the use of the dynamic cast mechanism ensures type safety. That is, an operator cannot be associated with a patch data type if the patch data type is not of the type supported by the operator handler.

There are some disadvantages to the *Chain of Responsibility* pattern. In particular, the pattern requires the implementation of an operator class and handler class for each concrete operator. The number of classes can be reduced by bundling operators together within larger classes and using conditionals to choose the correct behavior. However, the overall amount of source code required for this bundled implementation is about the same. In most applications, each chain is traversed only once for each variable. Once an operator is found and a pointer to its instance is returned, the operator may be called directly through the pointer. No future use of the chain is required. We believe that the general flexibility that we achieve using the chain mechanism far outweighs these drawbacks for our framework.

## 4 Summary and Conclusions

Object-oriented design patterns have been very useful in the design and development of the SAMRAI structured adaptive mesh refinement software architecture. By using patterns such as *Abstract Factory*, *Strategy*, and *Chain of Responsibility*, we have simplified the management of complex SAMR algorithms. Consequently, design patterns have enabled us meet two of our most important design goals: flexible, extensible algorithm support for a wide range of SAMR applications and generic support for arbitrary patch data types.

When considering the adoption of object-oriented techniques, the scientific computing

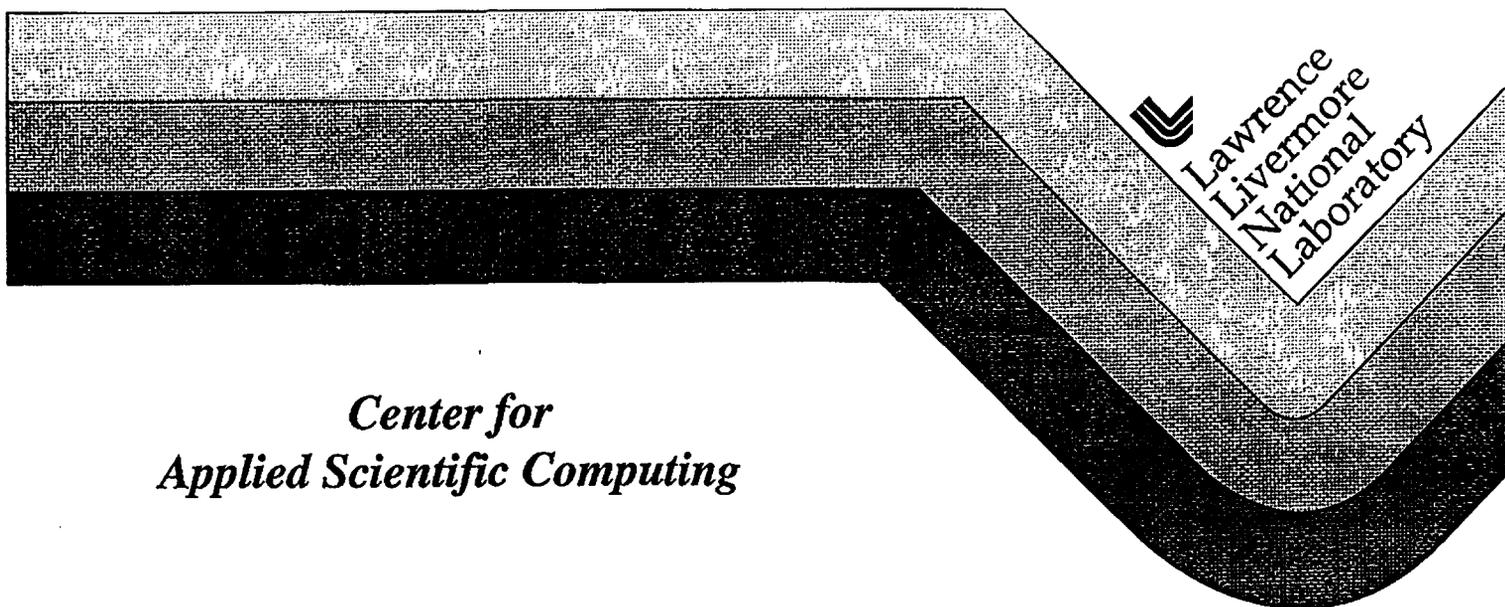
community has often focused on implementation and performance issues associated with “low-level” classes such as vectors, matrices, arrays, and C++ STL containers. While these abstractions are useful, we feel that object-oriented design offers the most benefit at the higher levels of a numerical software architecture. Object-oriented techniques enable the decomposition of complex algorithms into smaller, more manageable pieces that are suitable for a variety of applications. They promote code and algorithm reuse and also facilitate testing and management of software framework components. Most importantly, object-oriented patterns support more productive application construction by allowing rapid exploration of new algorithms that are built from both existing and new components.

## References

- [1] M. Aftosmis, M. Berger, and J. Melton, *Adaptation and surface modeling for cartesian mesh methods*, in Proceedings of the 12th AIAA Computational Fluid Dynamics Conference, San Diego, CA, June, 1995, 1995. AIAA Paper 95-1725.
- [2] A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. L. Welcome, *A conservative adaptive projection method for the variable density incompressible navier-stokes equations*, Tech. Rep. LBNL-39075, Lawrence Berkeley National Laboratory, Berkeley, CA, 1996.
- [3] M. J. Berger and P. Colella, *Local adaptive mesh refinement for shock hydrodynamics*, Journal of Computational Physics, 82 (1989), pp. 64–84.
- [4] M. J. Berger and J. Olinger, *Adaptive mesh refinement for hyperbolic partial differential equations*, Journal of Computational Physics, 53 (1984), pp. 484–512.
- [5] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *A System of Patterns*, John Wiley and Sons, New York, NY, 1996.
- [6] P. Colella, M. Dorr, and D. Wake, *Numerical simulation of plasma fluid equations using locally refined grids*, Tech. Rep. UCRL-JC-129913, Lawrence Livermore National Laboratory, Livermore, CA, 1998. submitted to J. Comp. Phys.
- [7] J. Coplien, *Advanced C++: Programming Styles and Idioms*, Addison-Wesley Publishing Co., Menlo Park, CA, 1992.
- [8] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of reusable Object-Oriented Software*, Addison-Wesley Publishing Co., Menlo Park, CA, 1995.
- [9] F. X. Garaizar and J. A. Trangenstein, *Adaptive mesh refinement and front tracking for shear bands in an antiplane shear model*. to appear in *SIAM Journal on Scientific Computing*, 1998.
- [10] R. D. Hornung and J. A. Trangenstein, *Adaptive mesh refinement and multilevel iteration for flow in porous media*, Journal of Computational Physics, 136 (1997), pp. 522–545.
- [11] J. P. Jessee, L. H. Howell, W. A. Fieveland, P. Colella, and R. B. Pember, *An adaptive mesh refinement algorithm for the discrete ordinates method*, in Proceedings of the 1996 National Heat Transfer Conference, Houston, TX, August 3–6, 1996, 1996.
- [12] R. I. Klein, J. B. Bell, R. B. Pember, and T. Kelleher, *Three dimensional hydrodynamic calculations with adaptive mesh refinement of the evolution of rayleigh taylor and richtmyer meshkov instabilities in converging geometry: Multi-mode perturbations*, in Proceedings of the 4th International Workshop on Physics of Compressible Turbulent Mixing, 1993.
- [13] S. Kohn, J. Weare, E. Ong, and S. Baden, *Software abstractions and computational issues in parallel structured adaptive mesh methods for electronic structure calculations*, in Proceedings of the Workshop on Structured Adaptive Mesh Refinement Grid Methods, Minneapolis, MN, March 1997, Springer-Verlag.
- [14] J. A. Trangenstein, *Adaptive mesh refinement for wave propagation in nonlinear solids*, SIAM J. Sci. Stat. Comput., 16 (1995), pp. 819–839.
- [15] J. Vlissides, *Pattern Hatching: Design Patterns Applied*, Addison-Wesley Publishing Co., Menlo Park, CA, 1998. Software Patterns Series.

# Coarse-Grid Selection for Parallel Algebraic Multigrid

A.J. Cleary  
R. Falgout  
V.E. Henson  
J.E. Jones



*Center for  
Applied Scientific Computing*

#### DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

#### PREPRINT

This is a preprint of a paper that was submitted to the Irregular 98 Conference, Berkeley, CA, August 9-11, 1998. This preprint is made available with the understanding that it will not be cited or reproduced without the permission of the authors.

# Coarse-grid Selection for Parallel Algebraic Multigrid

Andrew J. Cleary, Robert D. Falgout, Van Emden Henson, Jim E. Jones

Center for Applied Scientific Computing, Lawrence Livermore National Laboratory,  
Livermore, CA

**Abstract.** The need to solve linear systems arising from problems posed on extremely large, unstructured grids has sparked great interest in parallelizing algebraic multigrid (AMG). To date, however, no parallel AMG algorithms exist. We introduce a parallel algorithm for the selection of coarse-grid points, a crucial component of AMG, based on modifications of certain parallel independent set algorithms and the application of heuristics designed to insure the quality of the coarse grids. A prototype serial version of the algorithm is implemented, and tests are conducted to determine its effect on multigrid convergence, and AMG complexity.

## 1 Introduction

Since the introduction of algebraic multigrid (AMG) in the 1980's [4, 2, 3, 5, 19, 16, 18, 17] the method has attracted the attention of scientists needing to solve large problems posed on unstructured grids. Recently, there has been a major surge of interest in the field, due in large part to the need to solve increasingly larger systems, with hundreds of millions or billions of unknowns. Most of the current research, however, focuses either on improving the standard AMG algorithm [9, 7], or on dramatic new algebraic approaches [20, 6]. Little research has been done on parallelizing AMG. The sizes of the modern problems, however, dictate that large-scale parallel processing be employed.

Methods for parallelizing geometric multigrid methods have been known for some time [10], and most of the AMG algorithm can be parallelized using existing technology. Indeed, much of the parallelization can be accomplished using tools readily available in packages such as PETSc or ISIS++. But, the heart of the AMG setup phase includes the coarse-grid selection process, which is inherently sequential in nature.

In this note we introduce a parallel algorithm for selecting the coarse-grid points. The algorithm is based on modifications of parallel *independent set* algorithms. Also, we employ heuristics designed to insure the quality of the coarse grids. A prototype serial code is implemented, and we examine the effect the algorithm has on the multigrid convergence properties.

In Section 2 we outline the basic principles of AMG. Section 3 describes our parallelization model and the underlying philosophy, while the details of the parallel algorithm are given in Section 4. Results of numerical experiments with the serial prototype are presented and analyzed in Section 5. In Section 6 we make concluding remarks and indicate directions for future research.

## 2 Algebraic Multigrid

We begin by outlining the basic principles of AMG. Detailed explanations may be found in [17]. Consider a problem of the form  $A\mathbf{u} = \mathbf{f}$ , where  $A$  is an  $n \times n$  matrix with entries  $a_{ij}$ . For AMG, a “grid” is simply a set of indices of the variables, so the original grid is denoted by  $\Omega = \{1, 2, \dots, n\}$ . In any multigrid method, the central idea is that error not eliminated by relaxation is eliminated by solving the residual equation  $A\mathbf{e} = \mathbf{r}$  on a coarser grid, then interpolating  $\mathbf{e}$  and using it to correct the fine-grid approximation. The coarse-grid problem itself is solved by a recursive application of this method. Proceeding through all levels, this is known as a multigrid cycle. One purpose of AMG is to free the solver from dependence on geometry (which may not be easily accessible, if it is known at all). Hence, AMG fixes a relaxation method, and its main task is to determine a coarsening process that approximates error the relaxation cannot reduce.

Using superscripts to indicate level number, where 1 denotes the finest level so that  $A^1 = A$  and  $\Omega^1 = \Omega$ , the components that AMG needs are: “grids”  $\Omega^1 \supset \Omega^2 \supset \dots \supset \Omega^M$ ; grid operators  $A^1, A^2, \dots, A^M$ , interpolation operators  $I_{k+1}^k, k = 1, 2, \dots, M-1$ , restriction operators  $I_k^{k+1}, k = 1, 2, \dots, M-1$ , and a relaxation scheme for each level. Once these components are defined, the recursively defined multigrid cycle is as follows:

**Algorithm:**  $MV^k(\mathbf{u}^k, \mathbf{f}^k)$  The  $(\mu_1, \mu_2)$  V-cycle  
 If  $k = M$ , set  $\mathbf{u}^M = (A^M)^{-1}\mathbf{f}^M$   
 Otherwise:  
   Relax  $\mu_1$  times on  $A^k\mathbf{u}^k = \mathbf{f}^k$   
   Perform coarse grid correction  
   Set  $\mathbf{u}^{k+1} = \mathbf{0}, \mathbf{f}^{k+1} = I_k^{k+1}(\mathbf{f}^k - A^k\mathbf{u}^k)$   
   “Solve” on level  $k+1$  with  $MV^{k+1}(\mathbf{u}^{k+1}, \mathbf{f}^{k+1})$   
   Correct the solution by  $\mathbf{u}^k \leftarrow \mathbf{u}^k + I_{k+1}^k\mathbf{u}^{k+1}$   
   Relax  $\mu_2$  times on  $A^k\mathbf{u}^k = \mathbf{f}^k$

For this to work efficiently, two principles must be followed

**P1:** *Errors not efficiently reduced by relaxation must be well-approximated by the range of interpolation*

**P2:** *The coarse-grid problem must provide a good approximation to fine-grid error in the range of interpolation*

AMG satisfies P1 by automatically selecting the coarse grid and defining interpolation, based solely on the algebraic equations of the system. P2 is satisfied by defining restriction and the coarse-grid operator by the *Galerkin* formulation [14]:

$$I_k^{k+1} = (I_{k+1}^k)^T \quad \text{and} \quad A^{k+1} = I_k^{k+1}A^kI_{k+1}^k \quad (1)$$

Selecting the AMG components is done in a separate preprocessing step

AMG Setup Phase:

- 1 Set  $k = 1$
- 2 Partition  $\Omega^k$  into disjoint sets  $C^k$  and  $F^k$ 
  - (a) Set  $\Omega^{k+1} = C^k$
  - (b) Define interpolation  $I_{k+1}^k$
- 3 Set  $I_k^{k+1} = (I_{k+1}^k)^T$  and  $A^{k+1} = I_k^{k+1} A^k I_{k+1}^k$
- 4 If  $\Omega^{k+1}$  is small enough, set  $M = k + 1$  and stop. Otherwise, set  $k = k + 1$  and go to step 2

## 2.1 Selecting Coarse Grids and Defining Interpolation

Step 2 is the core of the AMG setup process. The goal of the setup phase is to choose the set  $C$  of coarse-grid points and, for each fine-grid point  $i \in F \equiv \Omega - C$ , a small set  $C_i \subset C$  of interpolating points. Interpolation is then of the form

$$(I_{k+1}^k \mathbf{u}^{k+1})_i = \begin{cases} \mathbf{u}_i^{k+1} & \text{if } i \in C, \\ \sum_{j \in C_i} \omega_{ij} \mathbf{u}_j^{k+1} & \text{if } i \in F \end{cases} \quad (2)$$

We do not detail the construction of the interpolation weights  $\omega_{ij}$ , instead referring the reader to [17] for details.

An underlying assumption in AMG is that smooth error is characterized by small residuals, that is,  $A\mathbf{e} \approx \mathbf{0}$ , which is the basis for choosing coarse grids and defining interpolation weights. For simplicity of discussion here, assume that  $A$  is a symmetric positive-definite  $M$ -matrix, with  $a_{ii} > 0$ ,  $a_{ij} \leq 0$  for  $j \neq i$ , and  $\sum a_{ij} \geq 0$ .

We say that point  $i$  depends on point  $j$  if  $a_{ij}$  is “large” in some sense, and hence, to satisfy the  $i$ th equation, the value of  $u_i$  is affected more by the value of  $u_j$  than by other variables. Specifically, the set of dependencies of  $i$  is defined by

$$S_i \equiv \left\{ j \neq i \mid -a_{ij} \geq \alpha \max_{k \neq i} (-a_{ik}) \right\}, \quad (3)$$

with  $\alpha$  typically set to be 0.25. We also define the set  $S_i^T \equiv \{j \mid i \in S_j\}$ , that is, the set of points  $j$  that depend on point  $i$ , and we say that  $S_i^T$  is the set of influences of point  $i$ .

A basic premise of AMG is that relaxation smoothes the error in the direction of influence. Hence, we may select  $C_i = S_i \cap C$  as the set of interpolation points for  $i$ , and adhere to the following criterion while choosing  $C$  and  $F$ :

**P3:** For each  $i \in F$ , each  $j \in S_i$  is either in  $C$  or  $S_j \cap C_i \neq \emptyset$

That is, if  $i$  is a fine point, then the points influencing  $i$  must either be coarse points or must themselves depend on the coarse points used to interpolate  $u_i$ .

The coarse grid is chosen to satisfy two criteria. We enforce P3 in order to insure good interpolation. However, we wish to keep the size of the coarse-grid as small as possible, so we desire that

P4:  $C$  is a maximal set with the property that no  $C$ -point influences another  $C$ -point

It is not always possible to enforce both criteria. Hence, we enforce P3 while using P4 as a guide in coarse-point selection.

AMG employs a two-pass process, in which the grid is first “colored”, providing a tentative  $C/F$  choice. Essentially, a point with the largest number of influences (“influence count”) is colored as a  $C$  point. The points depending on this  $C$  point are colored as  $F$  points. Other points influencing these  $F$  points are more likely to be useful as  $C$  points, so their influence count is increased. The process is repeated until all points are either  $C$  or  $F$  points. Next, a second pass is made, in which some  $F$  points may be recolored as  $C$  points to ensure that P3 is satisfied. Details of the coarse-grid selection algorithm may be found in [17], while a recent study of the efficiency and robustness of the algorithm is detailed in [7].

Like many linear solvers, AMG is divided into two main phases, the *setup* phase and the *solve* phase. Within each of these phases are certain tasks that must be parallelized to create a parallel AMG algorithm. They are

- Setup phase:
  - Selecting the coarse grid points,  $\Omega^{k+1}$
  - Construction of interpolation and restriction operators,  $I_{k+1}^k, I_k^{k+1}$
  - Constructing the coarse-grid operator  $A^{k+1} = I_k^{k+1} A^k I_k^{k+1}$
- Solve phase:
  - Relaxation on  $A^k \mathbf{u}^k = \mathbf{f}^k$
  - Calculating the residual  $\mathbf{r}^k \leftarrow \mathbf{f}^k - A^k \mathbf{u}^k$
  - Computing the restriction  $\mathbf{r}^{k+1} = I_k^{k+1} \mathbf{r}^k$
  - Interpolating and correcting  $\mathbf{u}^k \leftarrow \mathbf{u}^k + I_{k+1}^k \mathbf{u}^{k+1}$

### 3 Parallelization Model

In this work we target massively parallel distributed memory architectures, though it is expected that the method will prove useful in other settings, as well. Currently, most of the target platforms support shared memory within clusters of processors (typically of size 4 or 8), although for portability we do not utilize this feature. We assume explicit message passing is used among the processors, and implement this with MPI [15]. The equations and data are distributed to the processors using a *domain-partitioning* model. This is natural for many problems of physics and engineering, where the physical domain is partitioned by subdomains. The actual assignment to the processors may be done by the application code calling the solver, by the gridding program, or by a subsequent call to a graph partitioning package such as Metis [12]. The domain-partitioning strategy should not be confused with *domain decomposition*, which refers to a family of solution methods.

We use object-oriented software design for parallel AMG. One benefit of this design is that we can effectively employ kernels from other packages, such as

PETSc [1] in several places throughout our code. Internally, we focus on a *matrix object* that generalizes the features of “matrices” in widely-used packages. We can write AMG-specific routines once, for a variety of matrix data structures, while avoiding the necessity of reinventing widely available routines, such as matrix-vector multiplication.

Most of the required operations in the solve phase of AMG are standard, as are several of the core operations in the setup phase. We list below the standard operations needed by AMG:

- *Matrix-vector multiplication*: used for residual calculation, for interpolation, and restriction (both use rectangular matrices, restriction multiplies by the transpose). Some packages provide all of the above, while others may have to be augmented, although the coding is straightforward in these cases.
- *Basic iterative methods*: used for the smoothing step. Jacobi or scaled Jacobi are most common for parallel applications, but any iterative method provided in the parallel package could be applied.
- *Gathering/scattering processor boundary equations*: used in the construction of the interpolation operators and in the construction of coarse-grid operators via the Galerkin method. Each processor must access “processor-boundary equations” stored on neighboring processors. Because similar functionality is required to implement additive Schwarz methods, parallel packages implementing such methods already provide tools that can be modified to fulfill this requirement.

## 4 Parallel Selection of Coarse Grids

Designing a parallel algorithm for the selection of the coarse-grid points is the most difficult task in parallelizing AMG. Classical AMG uses a two-pass algorithm to implement the heuristics, P3 and P4, that assure grid quality and control grid size. In both passes, the algorithm is inherently sequential. The first pass can be described as:

- 1) Find a point  $j$  with maximal measure  $w(j)$ . Select  $j$  as a  $C$  point.
  - 2) Designate neighbors of  $j$  as  $F$  points, and update the measures of other nearby points, using heuristics to insure grid quality.
- Repeat steps 1) and 2) until all points are either  $C$  or  $F$  points.

This algorithm is clearly unsuitable for parallelization, as updating of measures occurs after each  $C$  point is selected. The second pass of the classical AMG algorithm is designed to enforce P3, although we omit the details and refer the reader to [17]. We can satisfy P3 and eliminate the second pass through a simple modification of step 2)

Further, we may allow for parallelism by applying the following one-pass algorithm. Begin by performing step 1) globally, selecting a *set* of  $C$  points,  $D$ , and then perform step 2) locally, with each processor working on some portion of the set  $D$ . With different criteria for selecting the set  $D$ , and armed with

various heuristics for updating the neighbors in 2), a family of algorithms may be developed. The overall framework is:

```

Input the  $n \times n$  matrix  $A^k$  (level  $k$ )
Initialize
   $F = \emptyset, C = \emptyset$ 
   $\forall i \in \{1 \dots n\},$ 
     $w(i) \leftarrow$  initial value
Loop until  $|C| + |F| = n$ 
  Select an independent set of points  $D$ 
   $\forall j \in D:$ 
     $C = C \cup j$ 
     $\forall k$  in set local to  $j$ , update  $w(k)$ 
    if  $w(k) = 0, F = F \cup k$ 
End loop

```

#### 4.1 Selection of the set $D$

For the measure  $w(i)$ , we use  $|S_i^T| + \sigma(i)$ , the number of points influenced by the point  $i$  plus a random number in  $(0, 1)$ . The random number is used as a mechanism for breaking ties between points with the same number of influences. The set  $D$  is then selected using a modification of a parallel maximal independent set algorithm developed in [13, 11, 8].

A point  $j$  will be placed in the set  $D$  if  $w(j) > w(k)$  for all  $k$  that either influence or depend on  $j$ . By construction, this set will be independent. While our implementation selects a maximal set of points possessing the requisite property, this is not necessary, and may not be optimal. An important observation is that this step can be done entirely in parallel, provided each processor has access to the  $w$  values for points with influences that cross its processor boundary.

#### 4.2 Updating $w(k)$ of neighbors

Describing the heuristics for updating  $w(k)$  is best done in terms of graph theory. We begin by defining  $S$ , the auxiliary *influence matrix*:

$$S_{ij} = \begin{cases} 1 & \text{if } j \in S_i, \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

That is,  $S_{ij} = 1$  only if  $i$  depends on  $j$ . The  $i$ th row of  $S$  gives the dependencies of  $i$  while the  $i$ th column of  $S$  gives the influences of  $i$ . We can then form the directed graph of  $S$ , and observe that a directed edge from vertex  $i$  to vertex  $j$  exists only if  $S_{ij} \neq 0$ . Notice that the directed edges point in the direction of dependence. To update the  $w(k)$  of neighbors, we apply the following pair of heuristics:

**P5:** Values at  $C$  points are not interpolated, hence, neighbors that influence a  $C$  point are less valuable as potential  $C$  points themselves

**P6:** If  $k$  and  $j$  both depend on  $c$ , a given  $C$  point, and  $j$  influences  $k$ , then  $j$  is less valuable as a potential  $C$  point, since  $k$  can be interpolated from  $c$

The details of how these heuristics are implemented are:

$$\begin{array}{ll}
 \forall c \in D, & \\
 \forall j \mid S_{cj} \neq 0, & \text{P5:} \\
 \quad w(j) \leftarrow w(j) - 1 & \text{(each } j \text{ that influences } c) \\
 \quad S_{cj} \leftarrow 0 & \text{(decrement the measure)} \\
 & \text{(remove edge } cj \text{ from the graph)} \\
 \forall j \mid S_{jc} \neq 0 & \text{P6:} \\
 \quad S_{jc} \leftarrow 0 & \text{(each } j \text{ that depends on } c), \\
 \quad \forall k \mid S_{kj} \neq 0, & \text{(remove edge } jc \text{ from the graph)} \\
 \quad \text{if } S_{kc} \neq 0 & \text{(each } k \text{ that } j \text{ influences),} \\
 \quad \quad w(j) \leftarrow w(j) - 1 & \text{(if } k \text{ depends on } c), \\
 \quad \quad S_{kj} \leftarrow 0 & \text{(decrement the measure)} \\
 & \text{(remove edge } kj \text{ from the graph)}
 \end{array}$$

The heuristics have the effect of lowering the measure  $w(k)$  for a set of neighbors of each point in  $D$ . As these measures are lowered, edges of the graph of  $S$  are removed to indicate that certain influences have already been taken into account. Frequently the step  $w(j) \rightarrow w(j) - 1$  causes  $\lfloor w(j) \rfloor = 0$ . When this occurs  $j$  is flagged as an  $F$  point.

Once the heuristics have been applied for all the points in  $D$ , a global communication step is required, so that each processor has updated  $w$  values for all neighbors of all their points. The entire process is then repeated.  $C$  points are added by selecting a new set,  $D$ , from the vertices that still have edges attached in the modified graph of  $S$ . This process continues until all  $n$  points have either been selected as  $C$  points or  $F$  points.

## 5 Numerical Experiments

To test its effect on convergence and algorithmic scalability, we include a serial implementation of the parallel coarsening algorithm in a standard sequential AMG solver. Obviously, this does not test parallel efficiency, which must wait for a full parallel implementation of the entire AMG algorithm.

Figure 1 shows the coarse grid selected by the parallel algorithm on a standard test problem, the 9-point Laplacian operator on a regular grid. This test is important because the grid selected by the standard sequential AMG algorithm

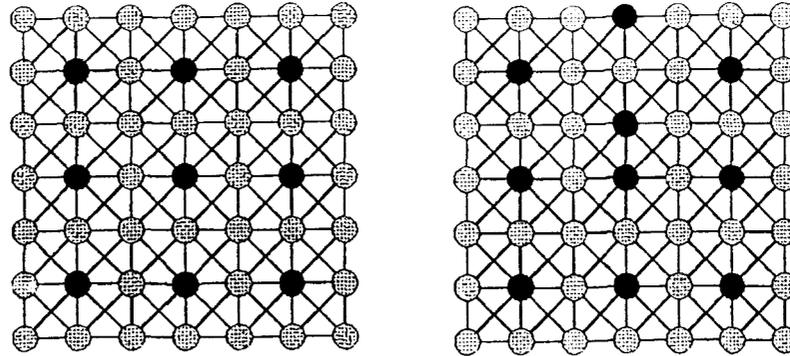


Fig. 1 Coarse grids for the structured-grid 9-point Laplacian operator. The dark circles are the  $C$  points. Left: Grid selected by the standard algorithm. Right: Grid selected by the parallel algorithm.

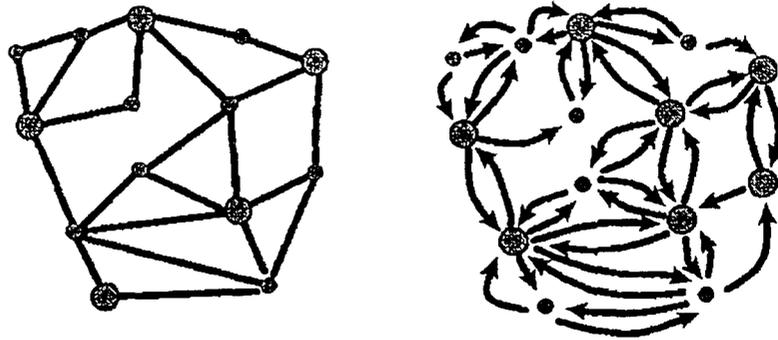


Fig. 2 Coarse grids for an unstructured grid. The large circles are the  $C$  points. Left: Grid selected by the standard algorithm. Right: Grid selected by the parallel algorithm. Graph connectivity is shown on the left, while the full digraph is shown on the right.

is also the optimal grid used in geometric multigrid for this problem. Examining many such test problems on regular grids, we find that the parallel coarsening algorithm typically produces coarse grids with 10–20% more  $C$  points than the sequential algorithm. On unstructured grids or complicated domains, this increase tends to be 40–50%, as may be seen in the simple example displayed in Figure 2.

The impact of the parallel coarsening algorithm on convergence and scalability is shown in two figures. Figure 3 shows the convergence factor for the 9-point Laplacian operator on regular grids ranging in size from a few hundred to nearly a half million points. Several different choices for the smoother and the parameter  $\alpha$  are shown. In Figure 4 the same tests are applied to the 9-point Laplacian operator for anisotropic grids, where the aspect ratios of the under-

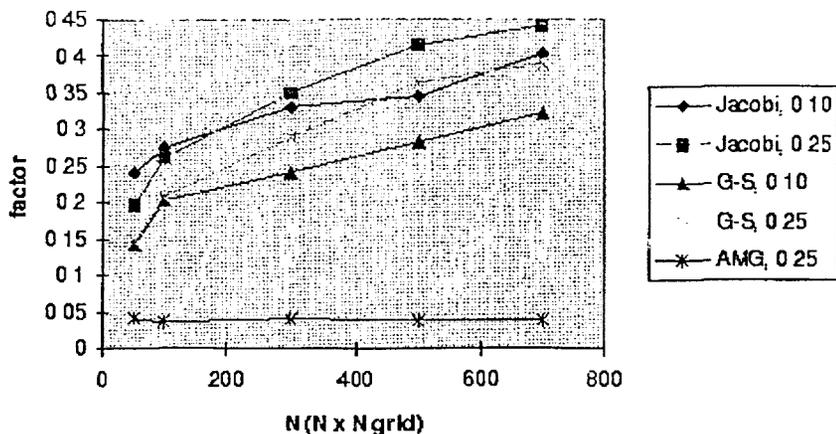


Fig 3 Convergence factors for parallel AMG for the 9-point Laplacian

lying quadrilateral finite elements are extremely high. In both figures, we see that convergence factors for the grids chosen by the parallel algorithm are significantly larger than standard AMG (shown as "AMG" in Figure 3, not shown in Figure 4), although the parallel algorithm still produce solutions in a reasonable number of iterations. Of more concern is that the convergence factors do not scale well with increasing problem size. We believe that this may be caused by choosing too many coarse grid points at once, and that simple algorithmic modifications mentioned below may improve our results.

Figure 5 shows the grid and operator complexities for the parallel algorithm applied to the 9-point Laplacian operator. Grid complexity is the total number of grid points, on all grids, divided by the number of points on the original grid. Operator complexity is the total number of non-zeros in all operators  $A^1, A^2$ , divided by the number of non-zeros in the original matrix. Both the grid and operator complexities generated using by the parallel algorithm are essentially constant with increasing problem size. While slightly larger than the complexities of the sequential grids, they nevertheless appear to be scalable.

The framework described in Section 4 permits easy modification of the algorithm. For example, one may alter the choice of the set  $D$  of  $C$  points. We believe that the convergence factor degradation shown in our results may be due to selecting too many coarse grid points. One possibility is to choose the minimal number of points in  $D$ , that is, one point per processor. This amounts to running the sequential algorithm on each processor, and there a number of different ways to handle the interprocessor boundaries. One possibility is to coarsen the processor boundary equations first, using a parallel MIS algorithm, and then treat each domain independently. Another option is to run the sequential algorithm on each processor ignoring the nodes on the boundary, and then patch up the

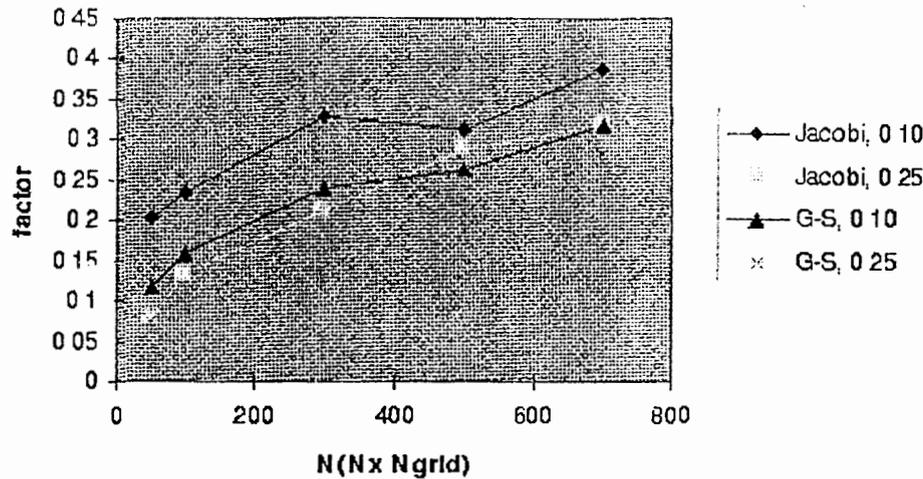


Fig. 4 Convergence rates for parallel AMG for the anisotropic grid problem

grids on the processor boundaries

## 6 Conclusions

Modern massively parallel computing requires the use of scalable linear solvers such as multigrid. For unstructured-grid problems, however, scalable solvers have not been developed. Parallel AMG, when developed, promises to be such a solver. AMG is divided into two main phases, the setup phase and the solve phase. The solve phase can be parallelized using standard techniques common to most parallel multigrid codes. However, the setup phase coarsening algorithm is inherently sequential in nature.

We develop a family of algorithms for selecting coarse grids, and prototype one member of that family using a sequential code. Tests with the prototype indicate that the quality of the selected coarse grids are sufficient to maintain constant complexity and to provide convergence even for difficult anisotropic problems. However, convergence rates are higher than for standard AMG, and do not scale well with problem size. We believe that this degradation may be caused by choosing too many coarse grid points at once, and that simple algorithmic modifications may improve our results. Exploration of these algorithm variants is the subject of our current research.

## References

- 1 S. BALAY, W. GROPP, L. C. MCINNIS, AND B. SMITH, *Petsc 2.0 user's manual*, Tech. Rep. ANL-95/11, Argonne National Laboratory, Nov. 1995.

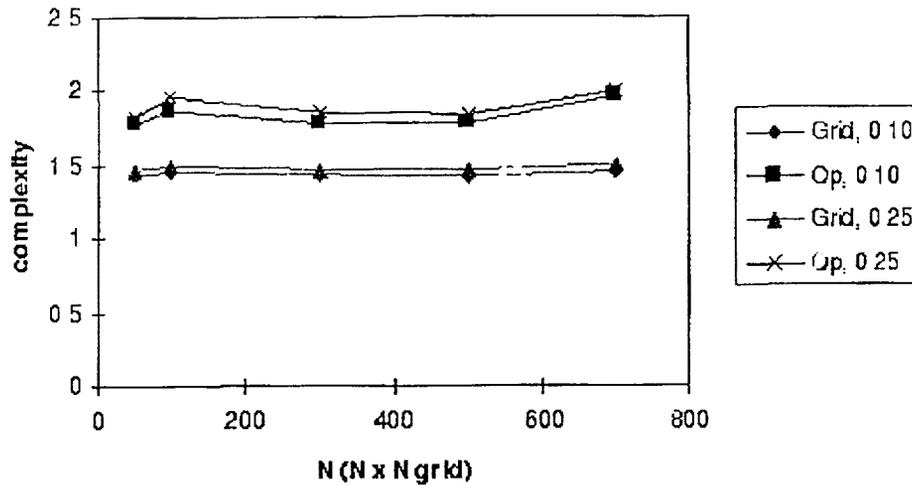


Fig. 5 Operator complexity for parallel AMG on example problem

- 2 A BRANDT, *Algebraic multigrid theory The symmetric case*, in Preliminary Proceedings for the International Multigrid Conference, Copper Mountain, Colorado, April 1983
- 3 A BRANDT, *Algebraic multigrid theory: The symmetric case*, Appl Math Comput, 19 (1986), pp 23-56
- 4 A BRANDT, S F MCCORMICK, AND J W RUGE, *Algebraic multigrid (AMG) for automatic multigrid solutions with application to geodetic computations* Report, Inst for Computational Studies, Fort Collins, Colo, October 1982
- 5 ———, *Algebraic multigrid (AMG) for sparse matrix equations*, in Sparsity and Its Applications, D J Evans, ed, Cambridge University Press, Cambridge, 1984
- 6 M BREZINA, A J CLEARY, R D FALGOUT, V E HENSON, J E JONES, T A MANTEUFFEL, S F MCCORMICK, AND J W RUGE, *Algebraic multigrid based on element interpolation (AMGe)* Submitted to the SIAM Journal on Scientific Computing special issue on the Fifth Copper Mountain Conference on Iterative Methods, 1998
- 7 A J CLEARY, R D FALGOUT, V E HENSON, J E JONES, T A MANTEUFFEL, S F MCCORMICK, G N MIRANDA, AND J W RUGE, *Robustness and scalability of algebraic multigrid* Submitted to the SIAM Journal on Scientific Computing special issue on the Fifth Copper Mountain Conference on Iterative Methods, 1998
- 8 R K GJERTSEN, JR, M T JONES, AND P E PLASSMAN, *Parallel heuristics for improved, balanced graph colorings*, Journal of Parallel and Distributed Computing, 37 (1996), pp 171-186
- 9 G GOLUBOVICI AND C POPA, *Interpolation and related coarsening techniques for the algebraic multigrid method*, in Multigrid Methods IV, Proceedings of the Fourth European Multigrid Conference, Amsterdam, July 6-9, 1993, vol 116 of ISNM, Basel, 1994, Birkhäuser, pp 201-213

- 10 J E JONES AND S F MCCORMICK, *Parallel multigrid methods*, in *Parallel Numerical Algorithms*, D E Keys, A H Sameh, and V Venkatakiishnan, eds , Dordrecht, Netherlands, 1997, Kluwer Academic Publications
- 11 M T JONES AND P E PLASSMAN, *A parallel graph coloring heuristic*, *SIAM Journal on Scientific Computing*, 14 (1993), pp 654–669
- 12 G KARYPIS AND V KUMAR, *A coarse-grain parallel multilevel k-way partitioning algorithm*, in *Proceedings of the 8th SIAM Conference on Parallel Processing for Scientific Computing*, 1997
- 13 M LUBY, *A simple parallel algorithm for the maximal independent set problem*, *SIAM Journal on Computing*, 15 (1986), pp 1036–1053
- 14 S F MCCORMICK, *Multigrid methods for variational problems: general theory for the V-cycle*, *SIAM J Numer Anal* , 22 (1985), pp 634–643
- 15 MPI FORUM, *MPI: A message-passing interface standard*, *International J Supercomputing Applications*, 8(3/4) (1994), pp 654–669
- 16 J W RUGE AND K STÜBEN, *Efficient solution of finite difference and finite element equations by algebraic multigrid (AMG)*, in *Multigrid Methods for Integral and Differential Equations*, D J Paddon and H Holstein, eds , The Institute of Mathematics and its Applications Conference Series, Clarendon Press, Oxford, 1985, pp 169–212
- 17 ———, *Algebraic multigrid (AMG)*, in *Multigrid Methods*, S F McCormick, ed , vol 3 of *Frontiers in Applied Mathematics*, SIAM, Philadelphia, PA, 1987, pp 73–130
- 18 K STÜBEN, *Algebraic multigrid (AMG) experiences and comparisons*, *Appl Math Comput* , 13 (1983), pp 419–452
- 19 K STÜBEN, U TROTTENBERG, AND K WITSCH, *Software development based on multigrid techniques*, in *Proc IFIP-Conference on PDE Software, Modules, Interfaces and Systems*, B Enquist and T Smedsaas, eds , Sweden, 1983, Söderköping
- 20 P VANĚK, J MANDEL, AND M BREZINA, *Algebraic multigrid based on smoothed aggregation for second and fourth order problems*, *Computing*, 56 (1996), pp 179–196

# SAMRAI

## Structured Adaptive Mesh Refinement Applications Infrastructure

### Technology

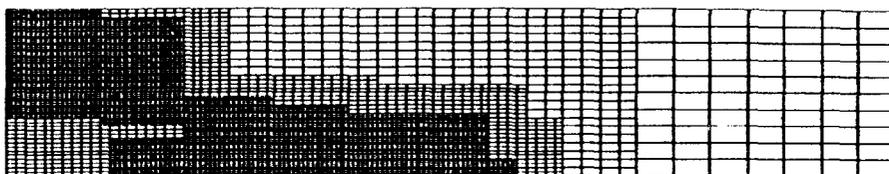
SAMRAI is an object-oriented code framework that provides computational scientists with general and extensible software support for rapid prototyping and development of parallel structured adaptive mesh refinement (AMR) applications. The primary goal of the SAMRAI effort is to facilitate numerical and algorithmic exploration in AMR applications that require high performance computing hardware.

**N**umerical simulations, especially those involving complex physical models and large spatial domains, are very expensive. In many applications of interest to science and engineering, the most important features of the physical processes occur in localized regions of the computational domain. Uniformly fine computational grids with sufficient resolution to capture these local phenomena may be unnecessarily fine outside the regions of interest. As a result, such uniform grid simulations can be inefficient or even prohibitively expensive.

Structured AMR provides a systematic way to focus computer resources (CPU time and memory) in local regions of a computational domain by employing varying degrees of spatial and temporal resolution. As such, AMR is an important technology needed to support large-scale, physically and numerically well-resolved, three-dimensional simulations.



aqueous phase saturation



adaptive mesh configuration

Fig. 1. Adaptive mesh refinement concentrates computational effort near the aqueous phase front in this polymer flooding simulation.

### Emerging AMR Application Domains

Structured AMR has been applied successfully to the numerical solution of systems of partial differential equations associated with fluid dynamics applications. SAMRAI is being developed to support the extension of traditional structured AMR methodology to new problem areas (see Figs. 1 and 2). These include, but are not limited to, problems modeled by tightly coupled systems of hyperbolic and elliptic or parabolic partial differential equations (such as hydrodynamics coupled with radiation diffusion, flow and transport in porous media, and combustion), neutron transport, hybrid models that combine vastly different numerical methods (e.g., discrete and continuum), and Arbitrary Lagrangian Eulerian (ALE) integration methods.

The application of AMR to these non-traditional problem areas gives rise to many interesting algorithmic, numerical, and computer science research questions. For example, the equations of radiation hydrodynamics—which simulate the transport of radiation and its interaction with matter via radiation emission and absorption—couple hyperbolic hydrodynamics with

parabolic radiation-diffusion. Developing radiation hydrodynamics AMR applications requires new solvers, new time integration methods, and extended data structure support.

In collaboration with scientists at LLNL, other DOE laboratories, and academia, we are exploring open research questions associated with the application of AMR to these non-traditional problem areas.

### The SAMRAI Framework

Building AMR applications is difficult and requires substantial software and algorithmic development. Experience has shown that AMR applications require substantially more complicated programming and longer development time than codes employing simpler uniform computational grids. AMR can be particularly challenging on parallel computers, since the programmer is responsible for managing and orchestrating complicated communication patterns among the processors of the machine. The SAMRAI framework facilitates the rapid prototyping of various design alternatives by freeing scientists from low-level data structure and algorithm management, and other implementation details.

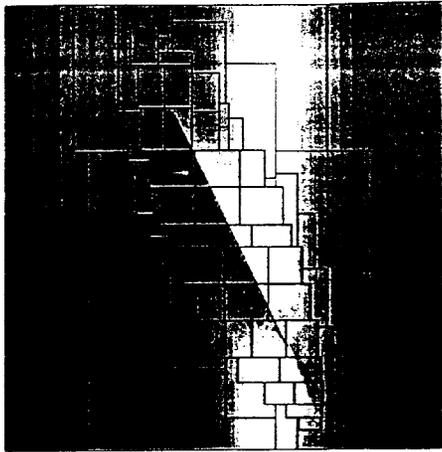


Fig. 2. AMR and front tracking techniques have been used to study the formation of shear bands in granular materials.

The SAMRAI framework must be applicable to a broad range of structured AMR applications; therefore, the design of the software architecture reflects an emphasis on generality and extensibility. Object-oriented design techniques have been applied throughout so that data structures and algorithms can be specialized and extended through derivation. In particular, SAMRAI's algorithmic framework can be adapted for a particular AMR application by derivation from SAMRAI's abstract classes and defining application-specific operations (see Fig. 3). Also, problem-dependent data types are incorporated easily by exploiting SAMRAI's flexible data structure abstractions.

In summary, SAMRAI reduces code duplication, encourages interoperability of application software, and simplifies the learning curve for new computational methods. SAMRAI's object-oriented architecture provides the flexibility to address a wide range of AMR applications. The advantages of this approach include reduced code development time and broader, more in-depth research into numerical methods for AMR applications.

### SAMRAI Applications and Framework Validation

In collaboration with scientists at LLNL and academia, we are using SAMRAI to develop applications in the areas of computational fluid dynamics, shear band formation in granular flow, flow and transport in porous media, and multi-physics problems. In addition to producing interesting scientific research, these applications serve as a validation of the SAMRAI software architecture since they require substantial data structure and algorithmic flexibility.

One particularly interesting application is multi-phase reactive flow and transport in porous media. The model equations contain general mass balance for the chemical species, general non-isothermal energy balance, and a wide variety of chemical reactions. The nonlinear coupling of equations in this system requires highly sophisticated linear and nonlinear solution techniques. The disparate length and time scales over which the chemical and physical mechanisms operate, along with the transient nature of the important phenomena, suggest that some form of AMR is necessary to treat the problem successfully. Developing

appropriate adaptive solution algorithms for this problem area is a challenging test for the SAMRAI algorithmic framework. Complicated domain geometry, including subsurface fractures and faults, and the need to represent different sets of primary variables in different regions of the physical domain (i.e., different sets of phases, species, and reactions) will motivate future development of SAMRAI framework.

Other applications presently targeted for SAMRAI development include radiation hydrodynamics simulations, for which hydrodynamics equations of hyperbolic character are coupled to parabolic equations modeling radiation diffusion, and high explosives simulations in which various processes (chemical, thermal, hydrodynamic, etc.) interact at different length and time scales.

For additional information about the SAMRAI project, contact Xabier Garaizar (510-423-1521, garaizar@llnl.gov), Richard Hornung (510-422-5097, hornung@llnl.gov), or Scott Kohn (510-422-4022, skohn@llnl.gov).

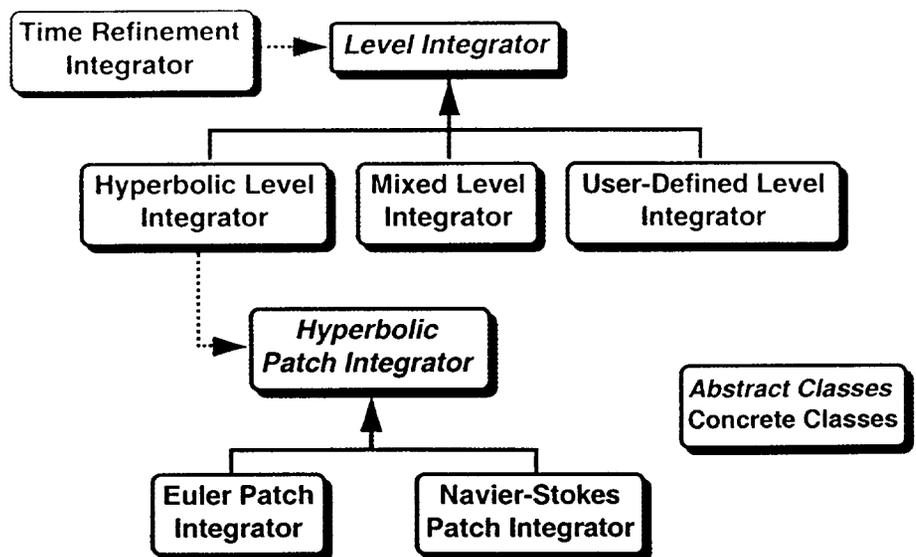


Fig. 3. The SAMRAI software framework supports a variety of AMR application domains through a careful object-oriented design.

UCRL-JC-127598

PREPRINT

# Software Abstractions and Computational Issues in Parallel Structured Adaptive Mesh Methods for Electronic Structure Calculations

S. Kohn  
J. Weare  
E. Ong  
S. Baden

This paper was prepared for submittal to the  
*Workshop on Structured Adaptive Mesh Refinement Grid Methods*  
*Minneapolis, MN*  
*March 12-13, 1997*

May 1997

The logo for Lawrence Livermore National Laboratory, featuring a stylized 'L' symbol and the text 'Lawrence Livermore National Laboratory' arranged in a chevron shape.

Lawrence  
Livermore  
National  
Laboratory

This is a preprint of a paper intended for publication in a journal or proceedings.  
Since changes may be made before publication, this preprint is made available with  
the understanding that it will not be cited or reproduced without the permission of the  
author.

#### DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

# SOFTWARE ABSTRACTIONS AND COMPUTATIONAL ISSUES IN PARALLEL STRUCTURED ADAPTIVE MESH METHODS FOR ELECTRONIC STRUCTURE CALCULATIONS\*

SCOTT KOHN<sup>†</sup>, JOHN WEARE<sup>‡</sup>, ELIZABETH ONG<sup>§</sup> AND SCOTT BADEN<sup>¶</sup>

**Abstract.** We have applied structured adaptive mesh refinement techniques to the solution of the LDA equations for electronic structure calculations. Local spatial refinement concentrates memory resources and numerical effort where it is most needed, near the atomic centers and in regions of rapidly varying charge density. The structured grid representation enables us to employ efficient iterative solver techniques such as conjugate gradient with FAC multigrid preconditioning. We have parallelized our solver using an object-oriented adaptive mesh refinement framework.

**Key words.** Structured Adaptive Mesh Refinement, Electronic Structure Calculations, LDA, Parallel Framework, Object Oriented Design.

**1. Introduction.** Electronic structure calculations attempt to accurately model the chemical properties of important materials through computer simulation. These computational methods complement traditional “wet” laboratory experiments. They help scientists to understand and predict the chemistry and structure of complex compounds. Simulations can provide insight into chemical behavior and material structure that is often unavailable from experiments; such insight can be used to guide the design of new classes of materials with desired properties.

Computations at the quantum mechanical level require the solution to some approximation of Schrödinger’s equation. The direct solution of Schrödinger’s equation is currently computationally intractable except for the smallest of molecules, since problem size scales exponentially with the number of electrons in the system. One common first-principles approximation—and the one we use here—is the Local Density Approximation (LDA) of Kohn and Sham [19].

Over the past thirty years, computational scientists have developed various approaches to solving the LDA equations. The most common and successful techniques in use today include Fast Fourier Transform (FFT) methods that expand the LDA equations using a planewave basis set [23]

---

\* This work has been supported by the NSF (ASC-9503997, ASC-9520372, and CCR-9403864), AFOSR (F49620-94-1-0286), ONR (N00014-93-1-0152 and N00014-91-J-1835), the UCSD School of Engineering, the San Diego Supercomputer Center, and by LLNL under DOE contract W-7405-Eng-48.

<sup>†</sup> Center for Applied Scientific Computing (CASC), Lawrence Livermore National Laboratory (LLNL), Livermore, CA.

<sup>‡</sup> Department of Chemistry and Biochemistry, University of California at San Diego (UCSD), La Jolla, CA.

<sup>§</sup> Department of Mathematics, UCSD, currently visiting CASC.

<sup>¶</sup> Department of Computer Science, UCSD.

and LCAO (Local Combination of Atomic Orbitals) methods that use a Gaussian basis set [2]. Other computational techniques include finite difference methods on uniform grids [9,6], wavelets [10,25], finite elements with  $p$ -refinement (but not spatial refinement) [27], and adaptive coordinate methods [13,14] that locally deform a logically rectangular mesh.

We are primarily interested in studying aperiodic systems that exhibit multiple length scales and therefore require local spatial refinement [8]. Examples of such systems include metal-carbon clusters or molecules with loosely bound, diffuse electrons. Ideally, our basis set should adapt to local changes in the electronic charge density, such as near atomic centers. Although LCAO methods support a form of local refinement, they do not scale well with increasing system size and can be inefficient when coupled to molecular dynamics. Planewave methods do not readily support local adaptivity since Fourier basis functions cover all space; consequently, local changes are propagated throughout the entire computational domain. The adaptive coordinate method has been somewhat successful in supporting spatial adaptivity; however, it is limited in the amount of local refinement since large mesh deformations can result in numerical instabilities.

To address the limitations of current simulation techniques, we have developed a prototype LDA code based on structured adaptive mesh refinement techniques using a finite element basis set. Adaptive methods nonuniformly place computational effort and memory in those portions of the problem domain with the highest error. Using our adaptive approach, we have studied systems with very short length scales that would have been difficult or infeasible with a uniform grid method.

In this paper, we describe some of the computational and numerical issues surrounding structured adaptive mesh refinement methods for electronic structure calculations. In particular, the software infrastructure needed to support these applications can become quite complex, especially on parallel computers. We also present computational results for some simple diatomic systems. Although our adaptive implementation is not yet competitive with the more mature planewave methods, we have identified changes that will improve the accuracy and competitiveness of the adaptive approach.

This paper is organized as follows. Section 2 introduces the LDA equations, and Section 3 describes the numerical methods employed in solving the LDA equations on a structured adaptive grid hierarchy. Section 4 provides an overview of the software framework and parallelization techniques. Section 5 evaluates our approach for a few diatomic molecules with known properties. Finally, Section 6 summarizes this work and highlights new research directions.

**2. The LDA Equations.** In the Local Density Approximation, the electronic wavefunctions are given by the solution of the nonlinear Kohn-

Sham eigenvalue equations

$$(2.1) \quad \mathcal{H}\psi_i = \epsilon_i\psi_i,$$

where the symmetric, indefinite LDA Hamiltonian  $\mathcal{H}$  is

$$(2.2) \quad \mathcal{H} = \left( \frac{-\nabla^2}{2} + V_{ext} + V_H(\rho) + V_{xc}(\rho) \right).$$

Atomic units are assumed throughout this paper. The electronic charge density is  $\rho(x) = \sum_{i=1}^N \|\psi_i(x)\|^2$ , and  $N$  is the number of occupied electron orbitals describing the system. The electronic charge density can be interpreted as giving the spatial distribution of the total electron charge. The eigenvectors (or wavefunctions)  $\psi_i$  are orthonormal and  $\epsilon_i$  is an eigenvalue. In general, we require the lowest  $N$  eigenvalues and associated eigenvectors. For a typical system of interest,  $N$  is a few tens to a few hundreds, and the number of basis functions used to describe each  $\psi_i$  is on the order of a quarter million.

The first term in the Hamiltonian operator represents the kinetic energy of a wavefunction.  $V_{ext}$  describes the interactions between an electron and the nuclear ions. The Hartree potential  $V_H$  models electron-electron repulsion and is the solution to the free-space or infinite domain Poisson problem

$$(2.3) \quad \nabla^2 V_H(x) = -4\pi\rho(x), V_H(x) = 0 \text{ as } \|x\| \rightarrow \infty.$$

$V_{xc}$  is the electron exchange-correlation functional and depends only on the local charge density. In our implementation, we use the correlation parameterization obtained by Vosko, Wilk, and Nusair [26]. Both  $V_H$  and  $V_{xc}$  are functions of the electron density, which, in turn, depends on the eigenfunctions; thus, the Kohn-Sham eigenvalue problem must be solved self-consistently.

The length scale difficulties in the LDA equations arise in the accurate representation of the external potential term  $V_{ext}$ , given by

$$(2.4) \quad V_{ext}(x) = \sum_a \frac{Z_a}{\|x - X_a\|},$$

where the sum is over the atoms in the system, and atom  $a$  has nuclear charge  $Z_a$  and position  $X_a$ . We have solved these “all-electron” problems with the  $\frac{1}{r}$  singularity in  $V_{ext}$  using our adaptive code but find that inner core electrons, which play little—if any—role in chemical bonding, create stiffness and conditioning problems for the discrete eigenvalue equations.

The core electron  $\frac{1}{r}$  singularities can be removed without much loss of accuracy by replacing the Coulomb attraction of the atomic centers using separable pseudopotentials [15,16]. For each species of atom  $a$ , we define a collection of pseudopotentials  $V_l^a$  and corresponding pseudowavefunctions

$u_l^a$  that solve the single-atom LDA equations for the valence electrons alone. The number of pseudopotentials in the expansion roughly depends on the type of bonding behavior associated with the atom; typically, three or four pseudopotentials are sufficient to approximate each atom. The  $V_{ext}$  term in the Hamiltonian then becomes

$$(2.5) \quad V_{ext}(x)\psi(x) = \sum_a V_{local}^a(x)\psi(x) + \sum_a \sum_{l,m} G_{l,m}^a u_{l,m}^a(x) \Delta V_l^a(x)$$

$$(2.6) \quad \Delta V_l^a(x) = V_l^a(x) - V_{local}^a(x)$$

$$(2.7) \quad G_{l,m}^a = \frac{\int u_{l,m}^a(x) \Delta V_l^a(x) \psi(x) dx}{\int u_{l,m}^a(x) \Delta V_l^a(x) u_{l,m}^a(x) dx}$$

$V_{local}^a$  is the local ionic pseudopotential and is typically chosen as the pseudopotential  $V_l^a$  with the largest quantum number  $l$ .

The application of pseudopotentials significantly softens  $V_{ext}$ ; however, depending on the types of atoms in the molecule,  $V_{ext}$  may still be too stiff for uniform grid methods. Pseudopotentials may be softened, but softening can introduce artificial physics. Our adaptive approach has been motivated by the need to accurately describe atoms such as oxygen or transition metals with stiff pseudopotentials.

Note that we have presented the LDA equations assuming a restricted spin formulation; that is, all up-spin electrons are paired with down-spin electrons. In many molecules, however, spins are not paired and the Local Spin Density (LSD) equations must be used instead. The LSD equations are similar to the restricted spin equations given above except that the exchange-correlation functional  $V_{xc}$  is now a function of two densities,  $\rho_\uparrow$  and  $\rho_\downarrow$ , corresponding to the two types of electrons.

**3. Computational Approach.** We seek to accurately model molecules containing atoms with steep pseudopotential representations, such as oxygen, fluorine, or transition metals (see Section 5). To do so requires some form of local spatial refinement about the atomic center to capture the rapidly varying pseudopotentials.

Structured adaptive mesh refinement methods solve partial differential equations using a hierarchy of nested, locally structured grids. All grids at the same level of the hierarchy have the same mesh spacing, but each successive level has higher spatial resolution than the ones preceding it, providing a more accurate representation of the solution (see Figure 3.1). Structured adaptive mesh techniques were originally developed for computational fluid dynamics [4,3].

We have implemented an LDA application using the techniques of structured adaptive mesh refinement methods. Although the data representations are similar, our eigenvalue problem has very different mathematical

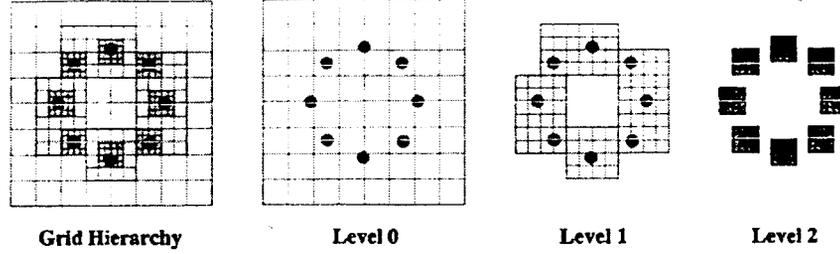


FIG. 3.1. Three levels of a structured adaptive mesh hierarchy. The eight dark circles represent regions of high error, such as atomic centers in a materials design application. The mesh spacing of each level is half of the previous coarser level.

properties than computational fluid dynamics and therefore requires different discretizations and solver algorithms. The following Sections cover our discretization approach and numerical algorithms in detail.

**3.1. Finite Element Discretization.** We discretize the Kohn-Sham equations (Eq. 2.1) using the finite element method, which for our application has a variety of advantages over competing discretization techniques. Finite elements readily admit local adaptivity. Finite element basis functions are very localized in space, interacting only with their immediate neighbors, and therefore do not suffer from the scaling problems of LCAO methods that use Gaussian basis sets. Finally, the finite element approach provides a consistent framework for defining operators across coarse-fine grid interfaces in adaptive grid hierarchies, as opposed to finite difference or finite volume discretizations that can result in nonsymmetric operators with complex Kohn-Sham eigenvalues.

The finite element approach expands a function  $f(x)$  in a basis of  $M$  functions  $\{\phi_j(x)\}$  with coefficient weights  $\alpha_j$

$$f(x) = \sum_{j=1}^M \alpha_j \phi_j(x).$$

All spatially varying quantities in the LDA equations—including the wavefunctions  $\psi_i$ , the charge density  $\rho$ , and the potentials  $V_{ext}$ ,  $V_H$ , and  $V_{xc}$ —are represented by their discrete expansion coefficients as indicated above. The Kohn-Sham equations are discretized using a Ritz formulation, resulting in the discrete nonlinear eigenvalue problem

$$\frac{1}{2} \int \nabla \phi_j \cdot \nabla \psi_i + \int \phi_j (V_{ext} + V_H(\rho) + V_{xc}(\rho)) \psi_i = \epsilon_i \int \phi_j \psi_i, \quad j = 1, \dots, M$$

where the unknowns are the coefficients in the expansion of  $\psi_i$ . Note that we have shown only one wavefunction  $\psi_i$  and one eigenvalue  $\epsilon_i$  to simplify the notation; the full Kohn-Sham equations involve a set of  $N$   $\psi_i$  coupled

through the charge density  $\rho$ . Our current code uses a 3d trilinear basis element  $\phi_j$  and approximates the rightmost two integrals in the above equation using the mid-point integration rule. The Hartree equation (Eq. 2.3) and the pseudopotential equations (Eqs. 2.5 through 2.7) are discretized in a similar manner.

Numerical computations on structured adaptive meshes consist of local array-based calculations on refinement patches and “fix-up” computations on the boundaries of the patches. Since computations are over structured domains, there is no need to explicitly create and store a sparse matrix. Instead, all operations are performed matrix-free. For example, the code to compute the discrete Laplacian operator on the interior of a grid patch uses the standard second order finite element stencil; however, the form of the stencil on coarse-fine grid interfaces becomes more complex. The following Section describes how to manage the computation at interfaces between coarse and fine grids.

**3.2. Grid Interfaces.** Our composite grid hierarchy uses node-centered refinement, as this is the natural centering for a second order linear finite element discretization. One difficulty with node-centered refinement on adaptive grids is that not all grid values are true degrees of freedom; rather, some grid points are “slaved” to the values of other nodes in the hierarchy. Here we refer to these grid points as “slave nodes.”

As shown in Figure 3.2, there are two types of slave nodes. In the first case, the slave node adjoins a coarse grid cell. To maintain continuity across grid interfaces, the value of the slave node must match the value at the edge of the coarse grid cell. For linear elements, the fine grid slave node is linearly interpolated from the two adjacent grid points. The second type of slave node exists wherever two fine grid cells overlap. In this case, the same degree of freedom is represented on two different refinement patches. One of the replicated values must be designated as the “real” value (the black point in the Figure) and the other as the “slave” (the grey point). To differentiate between slave and free nodes, we impose an ordering on all grid patches at one level of the refinement hierarchy. Degrees of freedom on lower numbered patches are designated real nodes and all overlapping nodes on higher numbered grids are marked as slave nodes. The numerical algorithm must ensure that slave nodes remain consistent with their corresponding degrees of freedom.

Recall that the calculation of an operator (e.g, the Laplacian) on the interior of a grid patch uses the standard uniform grid stencil; however, something special must be done on interfaces between coarse and fine grids. To compute the operator at coarse grid points, it is sufficient to inject nodal values from the fine grid into the coarse grid and then apply the uniform stencil. The computation on the fine grid interface is more complicated, especially in three dimensions. There are many interface cases to consider (see Figure 3.3 for two examples), and it would be tedious to catalog the

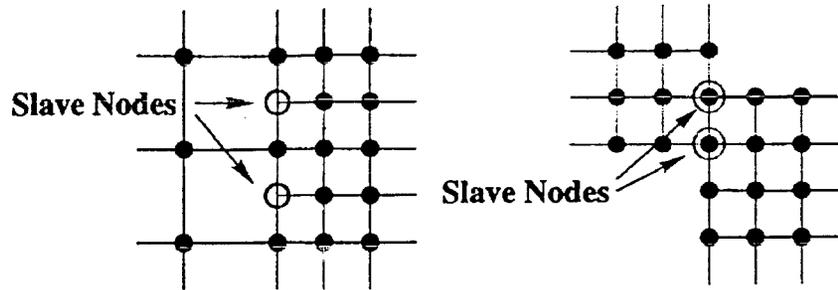


FIG. 3.2. *Slave nodes are mesh points that do not represent true degrees of freedom; rather, they are “slaved” to the values of other nodes. The values of the fine grid slave nodes on the left are determined by the finite element basis functions on the coarse grid. For linear elements, slave node values are linearly interpolated from neighboring mesh points. On the right, one degree of freedom is represented on two different fine patches. In this case, the nodes on the black patch are chosen as the true degrees of freedom and the grey nodes are “slaved” to the black nodes.*

appropriate stencils for these various cases.

Instead, we exploit the variational nature of the finite element formulation to compute the operator on the exterior of fine grid patches. The algorithm is as follows.

1. Whenever a new refinement level is created in the hierarchy, the boundary points of each new fine grid patch are tagged as to the refinement status of the cells adjacent to it. For example, the center node in Figure 3.3a would be tagged to indicate that the cells to the northwest and southwest are coarse cells. Likewise, the center node in Figure 3.3b would have northwest, southwest, and southeast tagged.
2. To compute the operator on the boundaries of the fine patch, first add a ghost cell boundary layer two cells wide to each patch. Fill the ghost cell region with interpolated coarse grid data. Then copy into the ghost cells data from adjacent fine grid cells, overwriting coarse grid data where there is overlap.
3. Apply the uniform grid stencil to the interior of the patch along with the one ghost cell layer surrounding it.
4. Finally, iterate over all the points on the boundary of the fine grid patch. Update the operator value by adding in the appropriately weighted values from all surrounding nodes that are not true degrees of freedom on the fine grid. The weights are determined from the finite element basis functions (see Figure 3.3).

Using this approach, there is no need to catalog all the various types of stencils at the interfaces. Instead, a relatively simple procedure can be used to compute stencil values directly. The only additional bookkeeping is a flag for each point on the boundary indicating the type of refinement

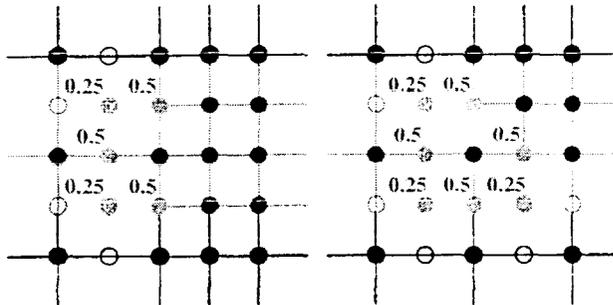


FIG. 3.3. To compute the composite grid operator at grid interfaces, we first grow the fine grid by two ghost cells and then apply the uniform grid stencil on the expanded domain. We then fix the values on the boundaries of the original patch using weighted sums of neighboring values. Shown here are the weights for the center node in the Figure; the weights are determined by the numerical values of the bilinear finite element at the neighboring nodes.

for the cells surrounding it.

**3.3. Eigenvalue Solver.** The Kohn-Sham equations (Eq. 2.1 and 2.2) pose a nonlinear eigenvalue problem. Standard eigenvalue solvers such as Lanczos [12] are not appropriate, since the Hamiltonian may change during the solution procedure due to the nonlinear coupling through the electronic charge density. Using a naive algorithm, such as steepest descent, would require extremely small step sizes (to guarantee convergence) dictated by the smallest length scales in the system, resulting in numerous iterations and unacceptably long solution times. Therefore, we use an eigenvalue solver technique developed by Longsine and McCormick called Simultaneous Rayleigh Quotient Minimization with subspace diagonalization [20].

The basic idea behind this approach is to take iterative steps that minimize the Rayleigh Quotient

$$RQ(\psi) = \frac{\int \psi \mathcal{H} \psi}{\int \psi \psi}$$

where  $\mathcal{H}$  is the Hamiltonian (Eq. 2.2) of the Kohn-Sham equations. The algorithm begins by freezing the nonlinear terms in the Hamiltonian. It then cycles through the wavefunctions in turn. For each wavefunction  $\psi_i$ , it takes a few iterations of the form  $\psi_i^{new} \leftarrow \psi_i + \alpha d$ , where  $\alpha$  minimizes the Rayleigh Quotient for that wavefunction,

$$\min_{\alpha} RQ(\psi_i + \alpha d),$$

assuming all other wavefunctions are fixed. Under the assumption that the Hamiltonian operator is approximately linear about the location  $\psi_i$ , the

method can compute the step size  $\alpha$  efficiently without a nonlinear search. The step directions  $d$  are generated via a CG-like process. After working through all wavefunctions, the solver performs a subspace diagonalization that accelerates the overall convergence of the method.

Our Rayleigh Quotient solver is essentially a band-by-band conjugate gradient solver, similar to other methods used in the materials science community. Unfortunately, these methods suffer from ill-conditioning problems with additional levels of refinement. We are actively pursuing a multilevel preconditioning technique to reduce the dependence on the number of refinement levels and therefore speed convergence. We are considering either a multigrid preconditioner [7] or a multilevel nodal basis preconditioner [5]. Experiments by Sung, Ong, and Weare [24] for planewave methods show the effectiveness of multilevel preconditioners for the eigenvalue equations.

**3.4. Hartree Equations.** Recall from Section 2 that the LDA Hartree potential  $V_H$  is the solution to a free-space Poisson equation

$$(3.1) \quad \nabla^2 V_H(x) = -4\pi\rho(x), V_H(x) = 0 \text{ as } \|x\| \rightarrow \infty.$$

$V_H$  is a function of the electron charge density  $\rho$ , which depends in turn on the wavefunctions  $\psi_i$ . Thus,  $V_H$  must be recalculated many times during the eigenvalue solution procedure. There are two parts to computing  $V_H$ : (1) obtaining the Dirichlet boundary conditions on a finite computational domain, and (2) solving the resulting boundary value problem on a nonuniform grid hierarchy.

**3.4.1. Free-Space Boundary Conditions.** Fast numerical methods such as multigrid require a finite computational domain  $\Omega$  with boundary conditions  $g(x)$  on  $\partial\Omega$ :

$$(3.2) \quad \nabla^2 V_H(x) = -4\pi\rho(x), V_H(x) = g(x) \text{ on } \partial\Omega.$$

Therefore, we must find a fast and accurate scheme for computing the boundary values on  $\partial\Omega$  that would arise from free-space boundary conditions on an infinite domain. We would prefer a method that scales as  $O(N)$  since our multigrid solver scales linearly with the number of unknowns.

We can evaluate the potential on the boundaries of the computational domain through a direct numerical integration of the Green's function

$$(3.3) \quad g(x) = -4\pi \int_{y \in \Omega} \frac{\rho(y)}{\|x - y\|} dy, \quad \forall x \in \partial\Omega.$$

However, this approach scales as  $O(N^{\frac{3}{2}})$ . To reduce the computational cost to  $O(N)$ , we have developed a method that employs a multipole-like approximation due to Anderson [1]. Instead of evaluating the Green's integral for each of the  $O(N^{\frac{3}{2}})$  boundary points in  $\partial\Omega$ , we only evaluate it at a small, constant number of points located on a sphere that encloses

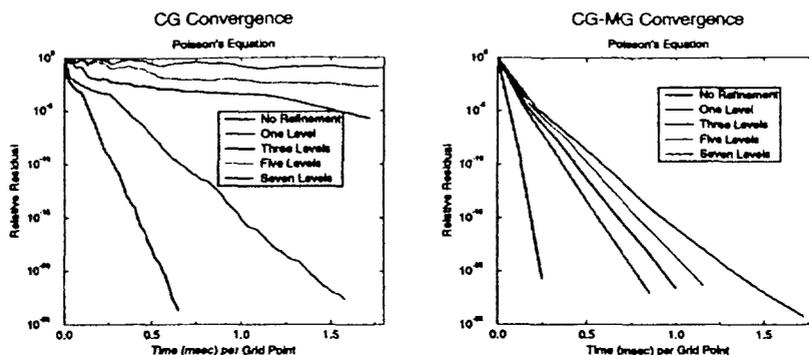


FIG. 3.4. *Preconditioning becomes essential with increasing local refinement. These graphs compare the convergence of (a) an nonpreconditioned conjugate gradient method with (b) a multigrid-preconditioned conjugate gradient solver as the number of levels of adaptive refinement is varied. The number of iterations is approximately proportional to the time per grid point.*

the boundary of the computational domain. Using the potentials at these locations, we then approximate the true boundary values using Anderson's multipole expansion formula. With little loss in accuracy, this approach reduces the overall computational cost from  $O(N^{\frac{3}{2}})$  to  $O(N)$ . The current method employs only 72 evaluation points but provides accuracy through the first eight multipole moments.

Our approximation is justified when the boundaries of the computational domain are "well-separated" from the support of the electronic charge density  $\rho$ . In practice, the boundaries are expanded so that each dimension of the computational domain is approximately twice the size of the support of the charge density. Uniformly spaced grids would therefore require eight times more storage. However, because we employ nonuniform grid refinements, we can represent this "expanded" area using a very coarsely spaced grid with little additional memory storage overhead.

**3.4.2. FAC Multigrid Solver.** One of the difficulties of solving the Hartree equations on a composite grid hierarchy is that the condition number of the discrete Kohn-Sham equations is dependent on the number of levels of refinement. As shown in Figure 3.4a, iterative methods such as nonpreconditioned conjugate gradient require twice as many iterations to converge with each new level of adaptive refinement (assuming a mesh refinement factor of two). Typical adaptive mesh computations such as the ones presented in Section 5 need up to six levels of adaptive refinement, resulting in perhaps sixteen times more iterations for a naive solver. Thus, practical and efficient implementations of the adaptive method require sophisticated numerical algorithms and scalable preconditioners.

Adaptive codes represent PDEs on a hierarchy of grids at different length scales; thus, it would seem appropriate to develop a multigrid-like

solver that could exploit this multiscale information and speed convergence. The multigrid method is a highly efficient and practical solver for many elliptic partial differential equations. Multigrid is optimal in the sense that it converges in a constant number of iterations independent of the size of the linear system of equations.

We have implemented a multigrid preconditioner to accelerate the computation of the Hartree potential (2.3). We use a variant of multigrid for structured adaptive mesh hierarchies called FAC (Fast Adaptive Composite) [21]. The advantage of FAC over competing adaptive multigrid methods is that it provides a consistent framework for applying the composite grid operator at interfaces between fine and coarse grids.

Figure 3.4b illustrates the performance of our Hartree solver with the FAC preconditioner. (Although we could use FAC by itself without CG, the conjugate gradient wrapper provides some extra robustness to the iterative solver.) Preconditioning significantly reduces the time to solution, especially for adaptive mesh hierarchies with many refinement levels. For example, for an adaptive mesh with five levels of refinement, the FAC solver reduces the Hartree residual by more than twenty orders of magnitude in the same time that the standard conjugate gradient method reduces it by only two orders of magnitude.

**4. Software Framework and Parallelism.** Structured adaptive mesh applications are difficult to implement on parallel architectures because they rely on dynamic, complicated data structures with irregular communication patterns. On message passing platforms, the programmer must explicitly manage data distribution across the processor memories and orchestrate interprocessor communication. Such implementation difficulties soon become unmanageable and can obscure the mathematics behind the algorithms.

To simplify the implementation of our application, we have developed an object-oriented adaptive mesh software infrastructure in C++ that provides high-level support for structured adaptive mesh applications. The main components of our framework are illustrated in Figure 4.1. MPI [22] is a basic message passing coordination and communication substrate. We have used KeLP [11] to parallelize work at one level of the mesh hierarchy. KeLP provides powerful mechanisms that manage data decomposition and interprocessor communication for irregular block-structured applications running on parallel architectures. KeLP adds very little execution-time overhead to MPI but can significantly reduce the bookkeeping complexity for dynamic block-structured codes. On top of KeLP, we have built functionality to support collections of levels arranged in an adaptive mesh hierarchy. Finally, the LDA application layer defines problem-specific classes such as molecule descriptions and energy functionals.

**4.1. Parallelization Approach.** Typically, parallelism in structured adaptive mesh applications lies across patches at a particular level of the

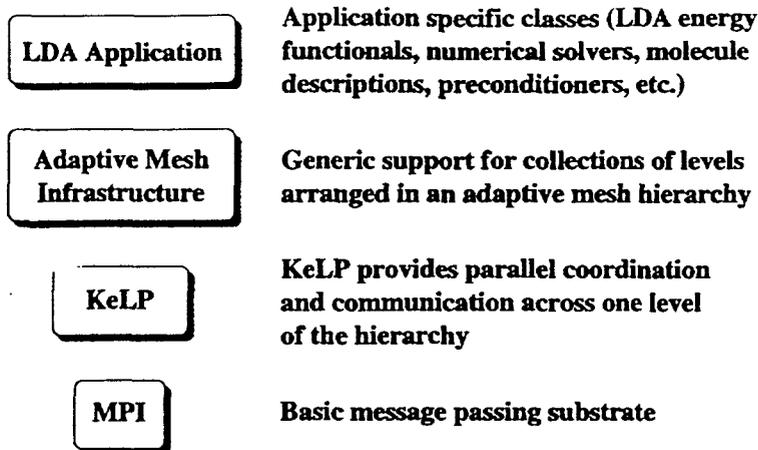


FIG. 4.1. Our LDA application support framework consists of three layers of C++ classes built on top of the MPI message passing system.

grid hierarchy. For example, the FAC multigrid method (see Section 3.4.2) cycles through levels sequentially but can compute in parallel across the refinement patches at each level. Thus, a natural data decomposition assigns each patch to a single processor. This is KeLP's data decomposition model.

Parallel loops over patches are executed as follows. First, the framework uses KeLP's communication schedule building mechanisms to describe the data motion that must take place to satisfy data dependencies. These descriptions of data motion can be quite complex. For example, the proper management of the slave nodes shown in Figure 3.2b implies that the slave values must not be included in data dependence computations since they do not represent actual degrees of freedom. If these slave nodes were communicated, their values might overwrite valid data. Next, this description is executed, forcing communication between processors via MPI's message passing mechanisms. The actual communication of data is managed by KeLP and is invisible to the programmer. Finally, now that data dependencies between patches have been satisfied, the patches have been decoupled and computation may proceed in parallel.

For our LDA application, the workload associated with each refinement region is directly proportional to the size of the region. Our simple but effective load balancing algorithm calculates the approximate average workload to be assigned to each processor and then recursively divides each patch until it is equal to or smaller than this average workload size. This guarantees that larger patches will be evenly divided across processors. However, we do not allow patches to get too small; although small patches may reduce load imbalance, they introduce additional interprocessor com-

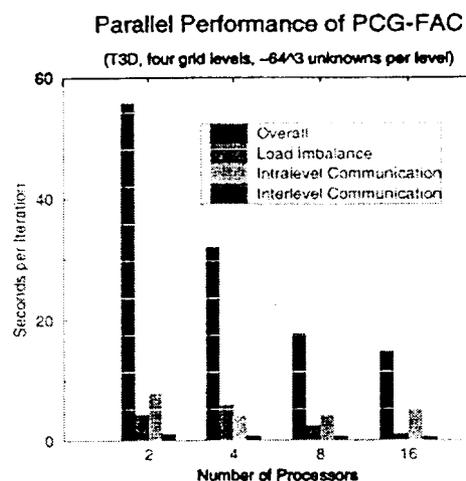


FIG. 4.2. Parallel performance timings on the Cray T3D for one iteration of the preconditioned conjugate gradient Hartree solver with an FAC preconditioner. The hierarchy consists of four levels, each with approximately  $64^3$  grid points. The problem could not be run on one node due to memory constraints.

munication.

After the patches have been subdivided, they are assigned “preferences” to processors based on the amount of overlap between the fine grid patch and the patches on the next coarser level. The patches are then sorted by decreasing size and bin-packed to processors. If possible, patches are assigned to their preferred processors; otherwise, they are assigned to any undersubscribed processor. More details concerning the load balancing and processor assignment algorithms can be found elsewhere [17,18].

**4.2. Parallel Performance.** In this Section, we present parallel performance results for the Hartree solver in the LDA equations. Recall from Section 3.4.2 that this entails solving Poisson’s equation on an adaptive grid hierarchy using conjugate gradient along with an FAC multigrid preconditioner. We did not gather parallel performance results for the entire LDA application since we have not yet studied sufficiently large molecular systems to justify parallel computation. In fact, the LDA solver should scale better than the Hartree solver alone, since much of the other work in the LDA solver, such as orthonormalization and the computation of  $V_{xc}$ , requires little communication.

Figure 4.2 shows the parallel performance of our Hartree solver test case on the Cray T3D. The application was compiled with full optimization. The hierarchy consists of a base level and three levels of refinement with approximately  $64^3$  unknowns per level. The numbers are reported for one iteration of the CG solver, which includes several applications of the composite grid operator, two FAC preconditioning cycles, and a number

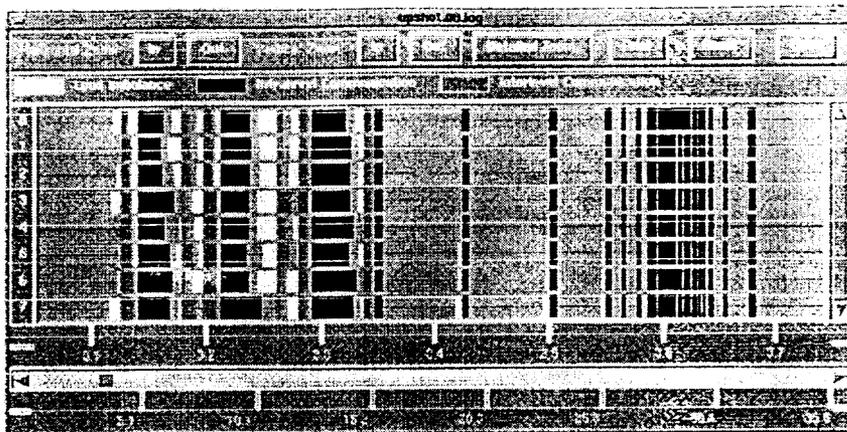


FIG. 4.3. A snapshot of the PCG-FAC Hartree computation on eight processors of the Cray T3D. Portions of the timeline without a filled box represent numerical computation. This picture was generated using Upshot, a parallel program visualization tool.

of inner product evaluations. The times are divided into four categories: overall execution time, time lost due to load imbalance, intralevel communication among grids on the same level of refinement, and interlevel communication between hierarchy levels. For this small problem size, scalability was limited to about sixteen processors, mostly due to the increasing costs of intralevel communication for the fixed problem size.

The relative cost of intralevel communication is shown graphically in Figure 4.3. This image is a snapshot of the Hartree computation on eight processors of the T3D. Time increases from left to right, and portions of the timeline without a filled box represent numerical computation. The leftmost portion of the timeline is the computation of the composite grid residual in FAC; note the interlevel communication to compute values at coarse-fine grid interfaces. The rightmost portion shows one multigrid V-cycle. Here communication is obviously dominating computation at the coarser levels of the multigrid hierarchy.

**5. Computational Results for Diatomic Molecules.** To validate the adaptive mesh refinement approach, we have applied our adaptive techniques to some simple diatomic problems whose LDA solutions are known. Figure 5.1 illustrates LDA results and Morse energy curve fits for  $\text{Be}_2$ ,  $\text{Li}_2$ ,  $\text{BeF}$ , and  $\text{F}_2$ .

All computations were performed using unfiltered Hamann pseudopotentials [15] with between  $200 \times 10^3$  (for  $\text{Be}_2$ ) and  $370 \times 10^3$  (for  $\text{F}_2$ ) grid points per wavefunction. Molecules were embedded into a computational domain measuring approximately 22 a.u. by 22 a.u. by 44 a.u. with a coarse grid spacing of 0.75 a.u. The mesh spacing of the finest grid used to resolve Li was about 0.1 a.u.; the mesh spacing for F was four times smaller.

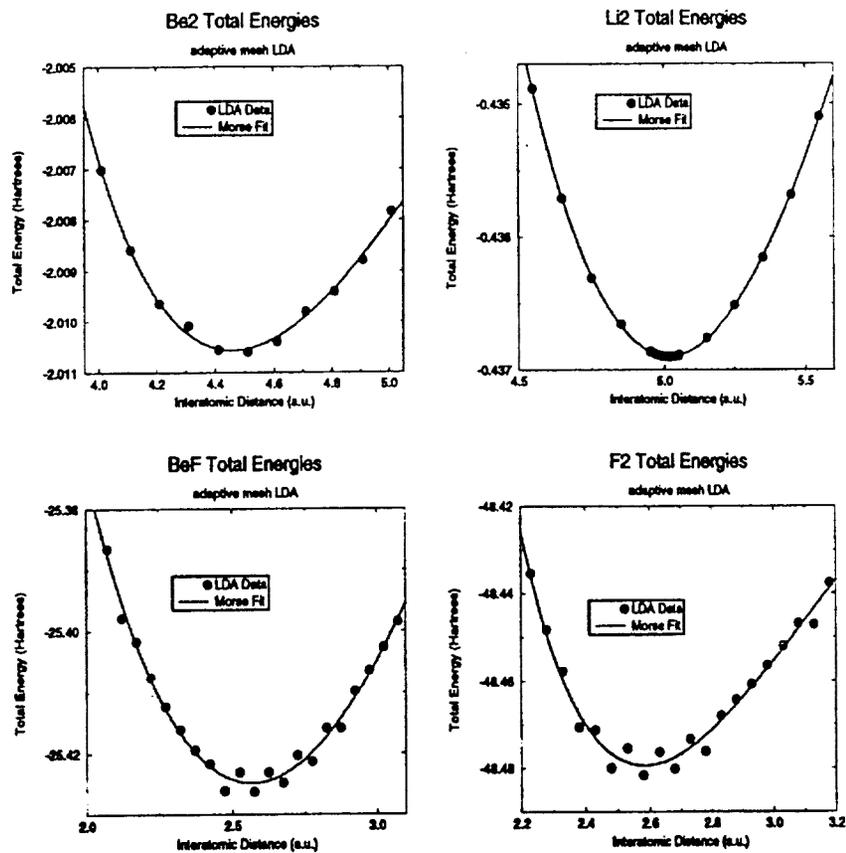


FIG. 5.1. Sample adaptive LDA calculations for various diatomic molecules:  $Be_2$ ,  $Li_2$ ,  $BeF$ , and  $F_2$ . The Morse curve values were calculated by taking a least squares fit of the LDA data to the standard diatomic Morse energy profile.

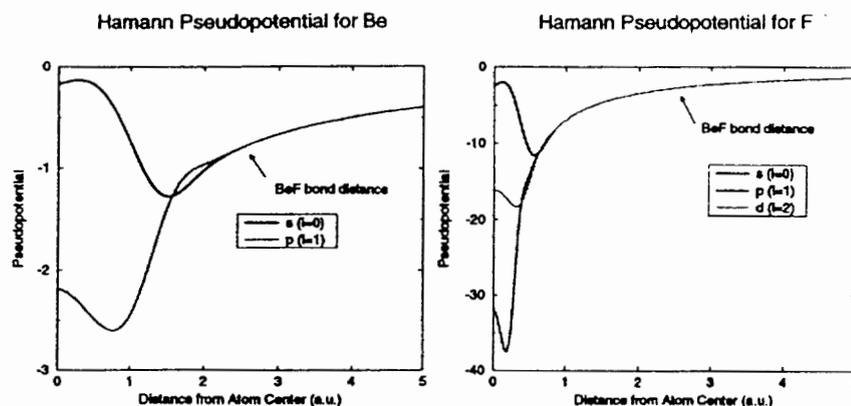


FIG. 5.2. Hamann pseudopotential descriptions for Be ( $V_l^{Be}$  at left) and F ( $V_l^F$  at right). Note the difference in vertical scales.

The refinement structures followed the atoms as they moved; otherwise, the atomic centers—where the most refinement is needed—would not have been resolved on the finest grid levels.

The number of refinement levels and mesh spacing were determined ad hoc by looking at the solution of the corresponding single-atom problems and comparing those solutions to the known LDA atomic energies. Clearly, this approach has limitations; for example, it cannot detect the formation of charge localization due to bonding between atoms. The design of a good automatic error estimator for the LDA equations is still an open research problem.

The  $\text{Be}_2$  and  $\text{Li}_2$  systems are easily calculated using the planewave approach, and our results match the planewave solutions.  $\text{BeF}$  is an example of a material with two very disparate length scales: the Be pseudopotential is very soft and delocalized whereas the F pseudopotential is very stiff and localized about the nucleus (see Figure 5.2). Computations with an unfiltered Hamann fluorine pseudopotential would require grids of size  $128^3$  or larger for the planewave method as compared to an equivalent of about  $70^3$  for the adaptive method.

The oscillations in the solution about the Morse fit for  $\text{BeF}$  and  $\text{F}_2$  are due to accuracy limitations in our current implementation of the adaptive method. We are currently using only second order finite elements, and our mid-point integration scheme does not preserve the variational nature of the finite element formalism. Obviously, these oscillations must be eliminated before it is possible to accurately calculate forces, which require derivatives of the energy profile with respect to position.

The  $O(h^2)$  convergence of our current finite element discretization means that we must use numerous mesh points to obtain the millihartree or better accuracy desired for materials calculations. Figure 5.3 illustrates

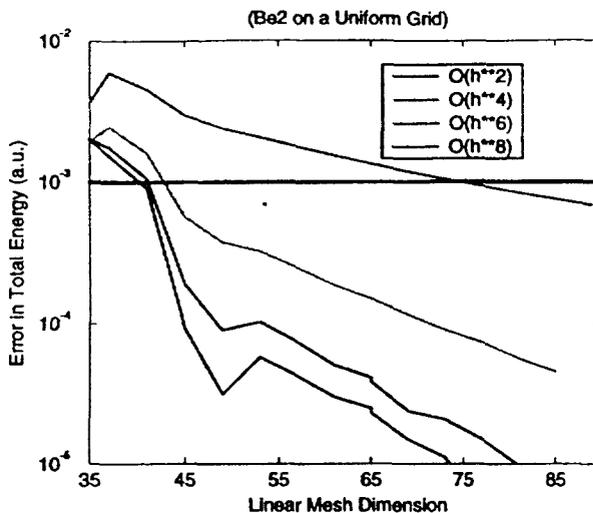


FIG. 5.3. Convergence in total LDA energy as a function of discretization order and mesh spacing for a Be<sub>2</sub> molecule. These results suggest that a sixth order  $O(h^6)$  stencil requires approximately half as many points (40 vs. 75 in each dimension) as a second order  $O(h^2)$  stencil for the same millihartree accuracy. In three dimensions, this represents an eight-fold reduction in mesh size.

the slow convergence in energy for the second order method as compared to the higher order methods. These results were calculated for Be<sub>2</sub> on a uniform computational grid with various orders of finite difference discretizations. (We used finite differences here since the grid is uniform and high order finite difference stencils are straightforward to derive and implement.) For millihartree accuracy, the second order method requires eight times more points (in 3d) than a sixth order method. Equivalently, for the same number of grid points, a sixth order method can provide 0.01 millihartree accuracy as compared to only millihartree accuracy for the second order method.

We are planning to develop an adaptive method that employs higher order elements to improve the accuracy of the method and reduce memory requirements. We plan to use either fourth or sixth order orthogonal elements [27]. Higher order methods should have the additional benefit of reducing the number of levels of refinement and thus improving the condition number of our discretized problems.

**6. Summary and Future Research Directions.** We have implemented an adaptive mesh refinement real-space code that solves the LDA equations for materials design. Our approach is unique in that it supports local spatial refinement of the computational domain. Unfortunately, our

computational results for simple diatomics indicate that our current code is not yet sufficiently accurate to address the classes of problems we wish to study, such as clusters containing transition metals.

The primary limitation of our current code is the accuracy of the second order finite element discretization. We plan to use high order orthogonal elements [27]; however, such elements introduce a number of interesting research questions. For example, it is unclear how to define the smoother, interpolation, and restriction operations within a geometric multigrid solver for the Hartree equations.

Another open research area is the design of an automatic error detector for guiding adaptive refinement. In this work, we used an ad hoc method based on the single-atom LDA solutions. However, this approach is clearly limited since it cannot detect the charge localization that results from molecular bonds. Given a variational formulation, we know that adding more basis functions will always decrease the energy of the solution, but it is not obvious that we can determine *a priori* where to place the refinement patches and how much to refine given an overall energy error tolerance.

Although we consider the results presented here promising, it is clear that further work remains before adaptive real space solvers can be considered competitive with planewave methods.

**Acknowledgements.** We would like to thank Eric Bylaska and Steven Fink for numerous valuable discussions on numerical methods in materials science and parallel implementations and performance.

## REFERENCES

- [1] C. R. ANDERSON, *An implementation of the fast multipole method without multipoles*, SIAM Journal on Scientific and Statistical Computing, 13 (1992), pp. 923–947.
- [2] J. ANDZELM AND E. WIMMER, *DGauss – a density functional method for molecular and electronic structure calculations in the 1990s*, Physica B, 172 (1991), pp. 307–317.
- [3] M. J. BERGER AND P. COLELLA, *Local adaptive mesh refinement for shock hydrodynamics*, Journal of Computational Physics, 82 (1989), pp. 64–84.
- [4] M. J. BERGER AND J. OLIGER, *Adaptive mesh refinement for hyperbolic partial differential equations*, Journal of Computational Physics, 53 (1984), pp. 484–512.
- [5] J. H. BRAMBLE, J. E. PASCIAK, AND J. XU, *Parallel multilevel preconditioners*, Mathematics of Computation, 55 (1990), pp. 1–22.
- [6] E. L. BRIGGS, D. J. SULLIVAN, AND J. BERNHOLC, *Large-scale electronic-structure calculations with multigrid acceleration*, Physical Review B, 52 (1995), pp. R5471–R5474.
- [7] W. L. BRIGGS, *A Multigrid Tutorial*, SIAM, 1987.
- [8] E. J. BYLASKA, S. R. KOHN, S. B. BADEN, A. EDELMAN, R. KAWAI, M. E. G. ONG, AND J. H. WEARE, *Scalable parallel numerical methods and software tools for material design*, in Proceedings of the Seventh SIAM Conference on Parallel Processing for Scientific Computing, San Francisco, CA, February 1995, SIAM, pp. 219–224.

- [9] J. R. CHELIKOWSKY, N. TROULLIER, K. WU, AND Y. SAAD, *Higher-order finite-difference pseudopotential method: An application to diatomic molecules*, Physical Review B, 50 (1994), pp. 11355–11364.
- [10] K. CHO, T. A. ARIAS, J. D. JOANNOPOULOS, AND P. K. LAM, *Wavelets in electronic structure calculations*, Physical Review Letters, 71 (1993), pp. 1808–1811.
- [11] S. J. FINK, S. B. BADEN, AND S. R. KOHN, *Flexible communication schedules for block structured applications*, in Third International Workshop on Parallel Algorithms for Irregularly Structured Problems, Santa Barbara, California, August 1996.
- [12] G. H. GOLUB AND C. F. V. LOAN, eds., *Matrix Computations (Second Edition)*, The Johns Hopkins University Press, Baltimore, 1989.
- [13] F. GYGI, *Electronic-structure calculations in adaptive coordinates*, Physical Review B, 48 (1993), pp. 11692–11700.
- [14] F. GYGI AND G. GALLI, *Real-space adaptive-coordinate electronic-structure calculations*, Physical Review B, 52 (1995), pp. R2229–R2232.
- [15] D. R. HAMANN, *Generalized norm-conserving pseudopotentials*, Physical Review B, 40 (1989), pp. 2980–2987.
- [16] L. KLEINMAN AND D. M. BYLANDER, *Efficacious form for model pseudopotentials*, Physical Review Letters, 48 (1982), pp. 1425–1428.
- [17] S. KOHN AND S. BADEN, *A parallel software infrastructure for structured adaptive mesh methods*, in Proceedings of Supercomputing '95, San Diego, California, December 1995.
- [18] S. R. KOHN, *A Parallel Software Infrastructure for Dynamic Block-Irregular Scientific Calculations*, Ph.D. thesis, University of California at San Diego, June 1995.
- [19] W. KOHN AND L. SHAM, *Self-consistent equations including exchange and correlation effects*, Physical Review, 140 (1965), pp. A1133–A1138.
- [20] D. E. LONGSINE AND S. F. MCCORMICK, *Simultaneous Rayleigh quotient minimization methods for  $Ax = \lambda Bx$* , Linear Algebra and Its Applications, 34 (1980), pp. 195–234.
- [21] S. F. MCCORMICK, ed., *Multilevel Adaptive Methods for Partial Differential Equations*, SIAM, Philadelphia, 1989.
- [22] MESSAGE PASSING INTERFACE FORUM, *MPI: A Message-Passing Interface Standard (v1.0)*, May 1994.
- [23] D. REMLER AND P. MADDEN, *Molecular dynamics without effective potentials via the Car-Parrinello approach*, Molecular Physics, 70 (1990), pp. 921–966.
- [24] M.-W. SUNG, M. E. G. ONG, AND J. H. WEARE, *Direct minimization of the Kohn-Sham equations using preconditioned conjugate gradient methods*, March 1995. American Physical Society March Meeting.
- [25] C. J. TYMCZAK AND X.-Q. WANG, *Orthonormal wavelet bases for quantum molecular dynamics*, Physical Review Letters, 78 (1997), pp. 3654–3657.
- [26] S. J. VOSKO, L. WILK, AND M. NUSAIR, *Accurate spin-dependent electron liquid correlation energies for local spin density calculations: A critical analysis*, Canadian Journal of Physics, 58 (1980), pp. 1200–1211.
- [27] S. R. WHITE, J. W. WILKINS, AND M. P. TETER, *Finite-element method for electronic structure*, Physical Review B, 39 (1989), pp. 5819–5833.

232076

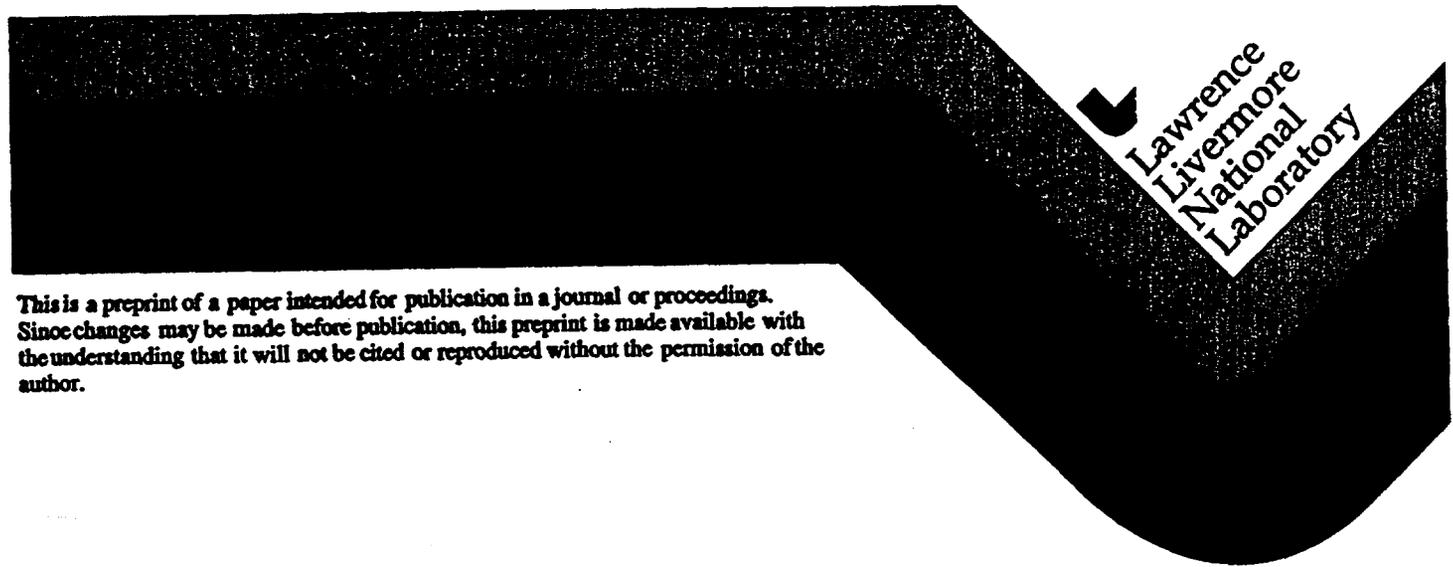
UCRL-JC-126043  
PREPRINT

# Parallel Adaptive Mesh Refinement for Electronic Structure Calculations

S. Kohn  
J. Weare  
E. Ong  
S. Baden

This paper was prepared for submittal to the  
*Eighth SLAM Conference on Parallel Processing for Scientific Computing*  
*Minneapolis, MN*  
*March 14-17, 1997*

December 1996



This is a preprint of a paper intended for publication in a journal or proceedings.  
Since changes may be made before publication, this preprint is made available with  
the understanding that it will not be cited or reproduced without the permission of the  
author.

#### DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

•

# Parallel Adaptive Mesh Refinement for Electronic Structure Calculations\*

Scott Kohn<sup>†</sup>    John Weare<sup>†</sup>    Elizabeth Ong<sup>‡</sup>    Scott Baden<sup>§</sup>

## Abstract

We have applied structured adaptive mesh refinement techniques to the solution of the LDA equations for electronic structure calculations. Local spatial refinement concentrates memory resources and numerical effort where it is most needed, near the atomic centers and in regions of rapidly varying charge density. The structured grid representation enables us to employ efficient iterative solver techniques such as conjugate gradients with multigrid preconditioning. We have parallelized our solver using an object-oriented adaptive mesh refinement framework.

## 1 Introduction

Computational materials design seeks to accurately model the chemical properties of important materials through computer simulation. Such simulations help scientists to understand the chemistry of complex compounds and also guide the design of new materials. One common first-principles approach used for electronic structure calculations is the Local Density Approximation (LDA) of Kohn and Sham [14].

Over the past thirty years, computational scientists have developed various approaches to solving the LDA equations. The most common and successful techniques in use today include Fast Fourier Transform (FFT) methods that expand the LDA equations using a planewave basis set [17] and LCAO (Local Combination of Atomic Orbitals) methods that use a Gaussian basis set [1]. Other computational techniques include finite difference methods on uniform grids [6, 4], wavelets [7], finite elements with  $p$ -refinement (but not spatial refinement) [19], and adaptive coordinate methods [9, 10].

We are primarily interested in studying systems that exhibit multiple length scales and therefore require local spatial refinement [5]. Ideally, our basis set should adapt to local changes in the electronic charge density, such as near atomic centers. Although LCAO methods support a form of local refinement, they do not scale well with increasing system size. Planewave methods do not readily support local adaptivity (although we are currently investigating techniques to implement Fast Fourier Transforms over structured adaptive mesh hierarchies). The adaptive coordinate method has been somewhat successful in supporting spatial adaptivity; however, it is limited in the amount of local refinement since large mesh deformations can result in numerical instabilities.

---

\*This work has been supported by the NSF (ASC-9503997, ASC-9520372, and CCR-9403864), AFOSR (F49620-94-1-0286), ONR (N00014-93-1-0152 and N00014-91-J-1835), the UCSD School of Engineering, the San Diego Supercomputer Center, and by LLNL under DOE contract W-7405-Eng-48.

<sup>†</sup>Center for Applied Scientific Computing, Lawrence Livermore National Laboratories, Livermore, CA

<sup>‡</sup>Department of Chemistry and Biochemistry, University of California at San Diego, La Jolla, CA

<sup>§</sup>Department of Mathematics, University of California at San Diego, La Jolla, CA

<sup>§</sup>Department of Computer Science, University of California at San Diego, La Jolla, CA

We have developed a prototype LDA code based on structured adaptive mesh refinement techniques using a finite element basis set. Adaptive methods nonuniformly place computational effort and memory in those portions of the problem domain with the highest error; thus, adaptive codes can target systems that are difficult or infeasible with other approaches.

In this paper, we describe some of the computational and numerical issues surrounding structured adaptive mesh refinement methods for electronic structure calculations. We also present computational results for some simple diatomic systems. Although our adaptive implementation is not yet competitive with the more mature planewave methods, we have identified changes that will improve the accuracy and competitiveness of the adaptive approach.

## 2 The LDA Equations

In the Local Density Approximation, the electronic wavefunctions are given by the solution of the nonlinear Kohn-Sham eigenvalue equations

$$(1) \quad \mathcal{H}\psi_i = \epsilon_i\psi_i,$$

where the symmetric, indefinite LDA Hamiltonian  $\mathcal{H}$  is

$$(2) \quad \mathcal{H} = \left( \frac{-\nabla^2}{2} + V_{\text{ext}} + V_H(\rho) + V_{\text{xc}}(\rho) \right).$$

The electronic charge density is  $\rho(x) = \sum_{i=1}^N \|\psi_i(x)\|^2$ , and  $N$  is the number of occupied electron orbitals describing the system. The eigenvectors (or wavefunctions)  $\psi_i$  are orthonormal and  $\epsilon_i$  is an eigenvalue. In general, we require the lowest  $N$  eigenvalues and associated eigenvectors. The Hartree potential  $V_H$  models electron-electron repulsion and is the solution to the free-space or infinite domain Poisson problem

$$(3) \quad \nabla^2 V_H(x) = -4\pi\rho(x), V_H(x) = 0 \text{ as } r = \|x\| \rightarrow \infty.$$

$V_{\text{ext}}$  describes electron-ion interaction, and  $V_{\text{xc}}$  is the electron exchange-correlation functional. Both  $V_H$  and  $V_{\text{xc}}$  are functionals of the electron density and thus the Kohn-Sham eigenvalue problem must be solved self-consistently.

The atomic core  $\frac{1}{r}$  singularities in  $V_{\text{ext}}$  create stiffness and conditioning problems for the eigenvalue equations. These singularities can be removed without much loss of accuracy by replacing the Coulomb attraction of the atomic centers using pseudopotentials [11, 12]. However, depending on the types of atoms in the system,  $V_{\text{ext}}$  may still be too stiff for uniform grid methods. Pseudopotentials may be softened, but softening can introduce artificial physics. Our adaptive approach has been motivated by the need to accurately describe atoms such as oxygen or transition metals with stiff pseudopotentials.

## 3 Structured Adaptive Mesh Refinement for LDA

Structured adaptive mesh refinement methods represent partial differential equations using a hierarchy of nested, locally structured grids. All grids at the same level of the hierarchy have the same mesh spacing, but successive levels have finer spacing than the ones preceding it, providing a more accurate representation of the solution (see Figure 1). Structured adaptive mesh techniques were originally developed for computational fluid dynamics [2]. Although the data representations are similar, our eigenvalue problem has very different mathematical properties and requires different discretization techniques and solver algorithms.

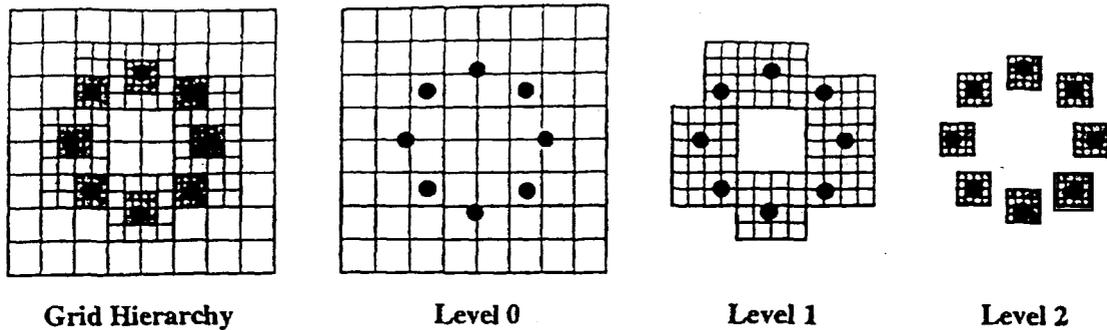


FIG. 1. Three levels of a structured adaptive mesh hierarchy. The eight dark circles represent regions of high error, such as atomic centers in a materials design application. The mesh spacing of each level is half of the previous coarser level.

### 3.1 Finite Element Discretization

We discretize the Kohn-Sham equations (1) using the finite element method, which has certain advantages over competing discretization techniques. Finite elements readily admit local adaptivity. Finite element basis functions are very localized in space, interacting only with their immediate neighbors, and therefore do not suffer from the scaling problems of LCAO methods that use Gaussian basis sets. Finally, the finite element approach provides a consistent framework for defining operators across coarse-fine grid interfaces in adaptive grid hierarchies, as opposed to finite difference discretizations that can result in nonsymmetric operators with complex Kohn-Sham eigenvalues.

The finite element approach expands the wavefunctions  $\psi$  in a basis of  $M$  functions  $\{\phi_i\}$

$$\psi(x) = \sum_{i=1}^M \psi_i \phi_i(x)$$

and the Kohn-Sham equations are discretized using a Ritz formulation, resulting in the discrete nonlinear eigenvalue problem

$$\frac{1}{2} \int \nabla \phi_i \nabla \psi + \int \phi_i (V_{ext} + V_H + V_{xc}) \psi = \epsilon \int \phi_i \psi, \quad i = 1, \dots, M.$$

Note that we have shown only one wavefunction  $\psi$  to simplify the notation; the full Kohn-Sham equations involve a set of  $N$   $\psi$  coupled through the charge density and  $V$ . Our current code uses a 3d trilinear basis element  $\phi_i$  and approximates the rightmost two integrals in the above equation using the mid-point integration rule.

Numerical computations on structured adaptive meshes consist of local array-based calculations on refinement patches and “fix-up” computations on the boundaries of the patches. Since computations are over structured domains, there is no need to explicitly create and store a sparse matrix. Instead, all operations are performed matrix-free. For example, the code to compute the Laplacian over a grid hierarchy consists of a standard 27-point stencil kernel along with code to correct values at grid interfaces. In our particular application, the time spent on boundary computations is less than about 10% of the time spent on patch interiors.

### 3.2 Code Infrastructure and Parallelization

Structured adaptive mesh applications are difficult to implement on parallel architectures because they rely on dynamic, complicated data structures with irregular communication

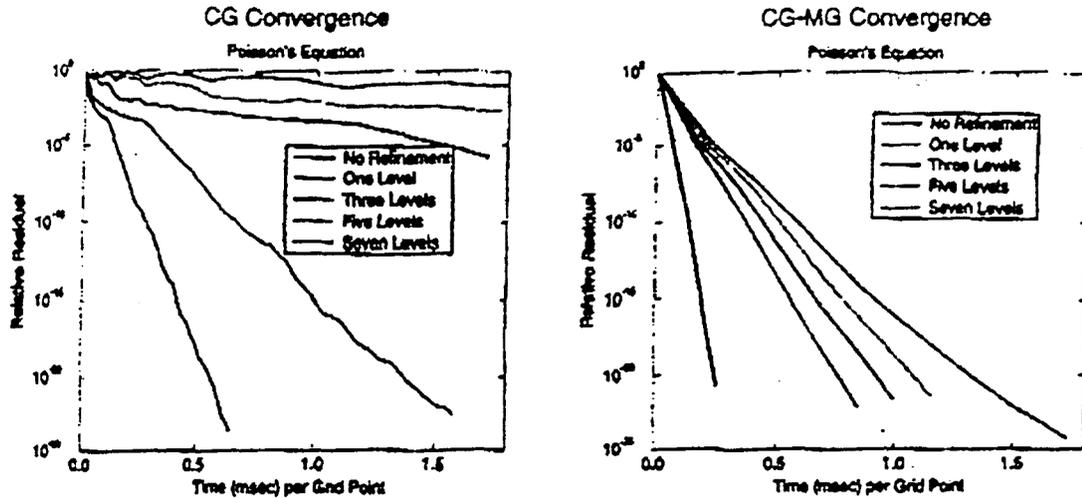


FIG. 2. Preconditioning becomes essential with increasing local refinement. These graphs compare the convergence of (a) an unpreconditioned conjugate gradient method with (b) a multigrid-preconditioned conjugate gradient solver as the number of levels of adaptive refinement is varied. The number of iterations is approximately proportional to the time per grid point.

patterns. To simplify the development of our application, we have developed an object-oriented adaptive mesh software infrastructure in C++ that provides high-level support for structured adaptive mesh applications [13].

Typically, parallelism in structured adaptive mesh applications lies across patches at a particular level of the grid hierarchy. For example, the FAC multigrid method (see Section 3.3.1) cycles through levels sequentially but computes in parallel across the refinement patches at each level. We have used the KeLP framework [8] to parallelize work at one level of the mesh hierarchy. KeLP provides high-level mechanisms that manage data decomposition and interprocessor communication for irregular block-structured applications running on parallel architectures.

### 3.3 Numerical Solvers

One of the difficulties of the adaptive approach is that the condition number of the discrete Kohn-Sham equations is dependent on the number of levels of refinement in the adaptive mesh hierarchy. As shown in Figure 2a, iterative methods such as unpreconditioned conjugate gradients require twice as many iterations to converge with each new level of adaptive refinement (assuming a mesh refinement factor of two). Typical adaptive mesh computations such as the ones presented in Section 4 need between two and four levels of adaptive refinement, resulting in between two and sixteen times more iterations for a naive solver. Thus, practical and efficient implementations of the adaptive method require more sophisticated numerical algorithms and preconditioners.

**3.3.1 FAC Multigrid** The multigrid method is a highly efficient and practical solver for many elliptic partial differential equations. Multigrid is optimal in the sense that it converges in a constant number of iterations independent of the size and conditioning of the linear system of equations.

We have implemented a multigrid preconditioner to accelerate the computation of the Hartree potential (3). We use a variant of multigrid for structured adaptive mesh hierarchies

called FAC (Fast Adaptive Composite) [16]. The advantage of FAC over competing adaptive multigrid methods is that it provides a consistent framework for defining the composite grid operator at interfaces between fine and coarse grids.

Figure 2b illustrates the performance of our Hartree solver with the FAC preconditioner. (Although we could use FAC by itself without CG, the conjugate gradient wrapper provides some extra stability to the iterative solver.) Preconditioning significantly reduces the time to solution, especially for adaptive mesh hierarchies with many levels of refinement. For example, for an adaptive mesh with six levels, the FAC solver reduces the Hartree residual by more than twenty orders of magnitude in the same time that the standard conjugate gradient method reduces it by only two orders of magnitude.

**3.3.2 Rayleigh Quotient Minimization** The same types of condition number scaling described in the previous Section for the Hartree equation (3) also apply to the Kohn-Sham eigenvalue problem (1). A naive iterative method such as steepest descent would require too many iterations to converge for the adaptive approach. Therefore, we use an eigenvalue solver technique developed by Longsine and McCormick called Simultaneous Rayleigh Quotient Minimization with Subspace Diagonalization [15].

The basic idea behind this approach is to take iterative steps that minimize the Rayleigh Quotient:

$$RQ(\psi) = \frac{\int \psi \mathcal{H} \psi}{\int \psi \psi}$$

where  $\mathcal{H}$  is the Hamiltonian (2) of the Kohn-Sham equations. The algorithm begins by freezing the nonlinear terms in the Hamiltonian. It then cycles through the wavefunctions in turn. For each wavefunction  $\psi_i$ , it takes a small number of steps of the form  $\psi_i^{new} \leftarrow \psi_i + \alpha d$ , where  $\alpha$  minimizes the Rayleigh Quotient for that wavefunction,

$$\min_{\alpha} RQ(\psi_i + \alpha d),$$

assuming all other wavefunctions are fixed. Under the assumption that the Hamiltonian operator is approximately linear about the location  $\psi_i$ , the method can compute the step size  $\alpha$  efficiently without a nonlinear search. The step directions  $d$  are generated via a CG-like process. After working through all wavefunctions, the solver performs a subspace diagonalization, which accelerates the convergence for the wavefunctions with outlying eigenvalues.

## 4 Computational Results for Diatomic Molecules

To validate the adaptive mesh refinement approach, we have applied our adaptive techniques to some simple diatomic problems whose LDA solutions are known. Figure 3 illustrates LDA results and Morse fits for  $\text{Be}_2$ ,  $\text{Li}_2$ ,  $\text{BeF}$ , and  $\text{F}_2$ . All computations were performed using unfiltered Hamann pseudopotentials [11] with between  $200 \times 10^3$  (for  $\text{Be}_2$ ) and  $370 \times 10^3$  (for  $\text{F}_2$ ) grid points per wavefunction.

The  $\text{Be}_2$  and  $\text{Li}_2$  systems are easily calculated using the planewave approach, and our results match the planewave solutions.  $\text{BeF}$  is an example of a material with two very disparate length scales: the Be pseudopotential is very soft and delocalized whereas the F pseudopotential is very stiff and localized about the nucleus. Computations with an unfiltered Hamann fluorine pseudopotential would require grids of size  $128^3$  or larger for the planewave method as compared to an equivalent of about  $70^3$  for the adaptive method.

The oscillations in the solution about the Morse fit for  $\text{BeF}$  and  $\text{F}_2$  are due to accuracy limitations in our current implementation of the adaptive method. We are currently using

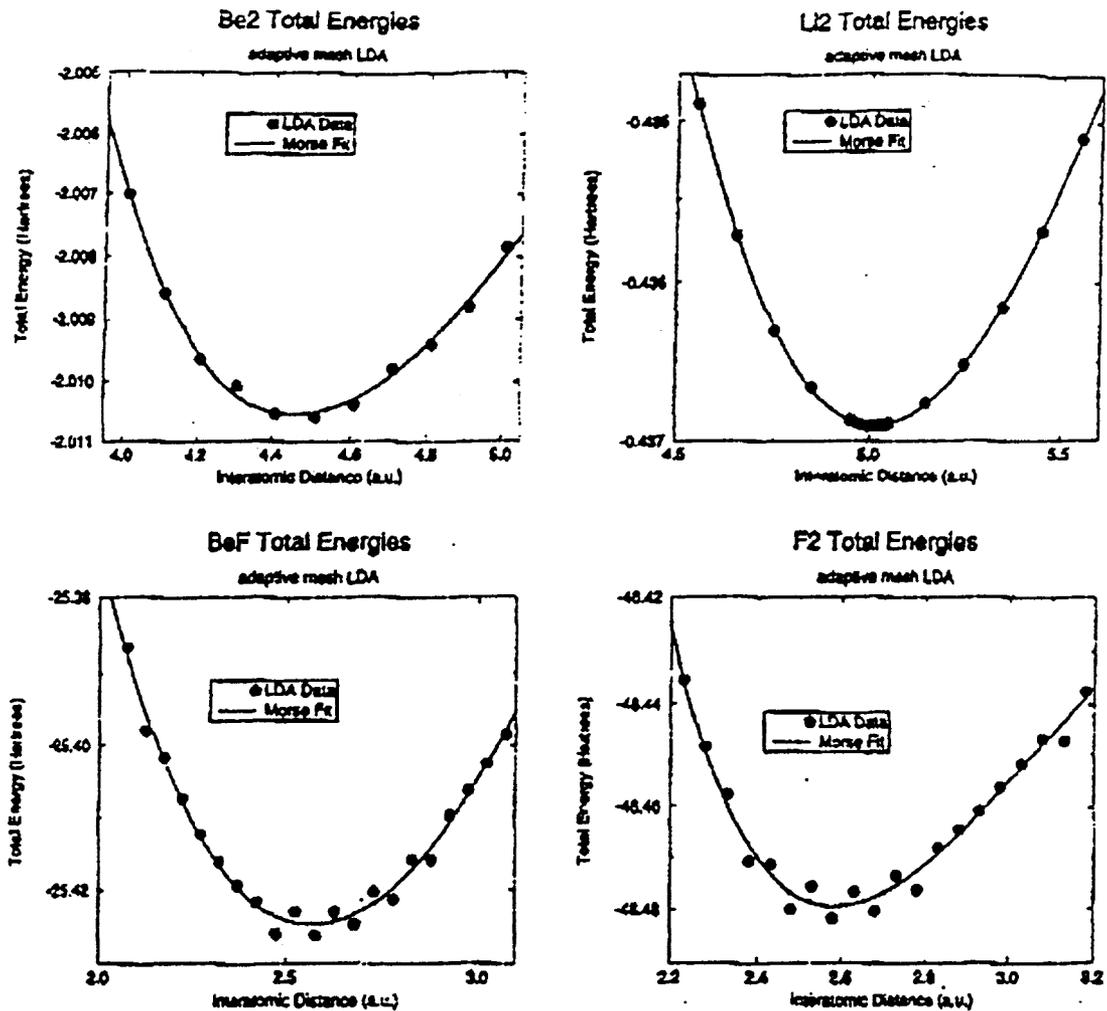


FIG. 3. Sample adaptive LDA calculations for various diatomic molecules:  $\text{Be}_2$ ,  $\text{Li}_2$ ,  $\text{BeF}$ , and  $\text{F}_2$ . The Morse curve values were calculated by taking a least squares fit of the LDA data to the standard diatomic Morse energy profile.

only second order finite elements, and our mid-point integration scheme does not preserve the variational nature of the finite element formalism. Obviously, these oscillations must be eliminated before it is possible to accurately calculate forces, which require derivatives of the energy profile with respect to position. In the following Section, we discuss future development efforts that will improve the accuracy of our adaptive code.

## 5 Analysis and Future Research Directions

We have implemented an adaptive mesh refinement real-space code that solves the LDA equations for materials design. Our approach is unique in that it supports local spatial refinement of the computational domain. Unfortunately, our computational results for simple diatomics indicate that our current code is not yet sufficiently accurate to address the classes of problems we wish to study, such as clusters containing transition metals. However, below we identify changes that will improve the accuracy and computational speed of the adaptive approach.

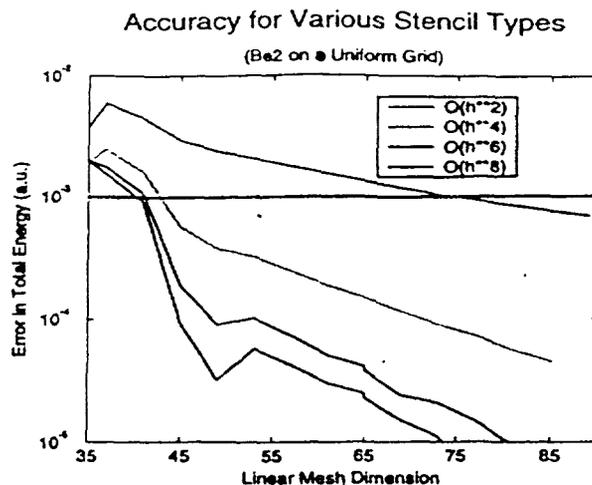


FIG. 4. Convergence in total LDA energy as a function of stencil order and mesh spacing for a  $\text{Be}_2$  molecule. These results suggest that a sixth order  $O(h^6)$  stencil requires approximately half as many points (40 vs. 75 in each dimension) as a second order  $O(h^2)$  stencil for the same millihartree accuracy. In three dimensions, this represents an eight-fold reduction in mesh size.

### 5.1 Higher Accuracy

Our current adaptive solver uses 3d trilinear elements that are  $O(h^2)$  accurate; these types of elements are commonly used in the finite elements applications community. Unfortunately, this low order means that we must use numerous mesh points to obtain the millihartree or better accuracy desired for materials calculations. Figure 4 illustrates the slow convergence in energy for the second order method as compared to the higher order methods. These results were calculated for  $\text{Be}_2$  on a uniform computational grid using various orders of finite difference stencils. For millihartree accuracy, the second order method requires eight times more points (in 3d) than a sixth order method. Equivalently, for the same number of grid points, a sixth order method can provide 0.01 millihartree accuracy as compared to only millihartree accuracy for the second order method.

We are currently developing an adaptive method that employs higher order elements to improve the accuracy of the method and reduce memory requirements. We plan to use either fourth or sixth order orthogonal elements [19]. Higher order methods should have the additional benefit of reducing the number of levels of refinement and thus improving the condition number of our numerical problems.

Another potential approach employs nonuniform Fast Fourier Transforms over the adaptive grid hierarchy. By exploiting the locally uniform grid spacing of a structured hierarchy, we have been able to implement a fast  $O(N \log N)$  transform for 1d hierarchies. We are currently developing a full 3d transform. Although this work is very preliminary, a nonuniform FFT may offer the best of both worlds: spectral accuracy for reciprocal space computations and local spatial refinement for real space computations.

### 5.2 Eigenvalue Solver Preconditioning

The computational results in Section 3.3.1 illustrate how a good preconditioning method can significantly reduce the iteration count and thus time to solution. Although we have been successful in developing good preconditioning techniques for the Hartree calculation, we have not as yet developed an efficient preconditioner for our eigenvalue solver.

Our Rayleigh Quotient solver is more efficient than a steepest descent approach, but it still suffers from scaling problems with additional refinement levels. We are actively pursuing a multilevel preconditioning technique to reduce the number of iterations needed for convergence. We are considering either a multigrid preconditioner or a multilevel nodal basis preconditioner [3]. Experiments by Sung, Ong, and Weare [18] for plane-wave methods show the effectiveness of multilevel preconditioners for the eigenvalue equations.

## References

- [1] J. Andzelm and E. Wimmer. DGauss - a density functional method for molecular and electronic structure calculations in the 1990s. *Physica B*, 172:307-317, 1991.
- [2] Marsha J. Berger and Phillip Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82(1):64-84, May 1989.
- [3] James H. Bramble, Joseph E. Pasciak, and Jinchao Xu. Parallel multilevel preconditioners. *Mathematics of Computation*, 55:1-22, 1990.
- [4] E. L. Briggs, D. J. Sullivan, and J. Bernholc. Large-scale electronic-structure calculations with multigrid acceleration. *Physical Review B*, 52(8):R5471-R5474, 1995.
- [5] Eric J. Bylaska, Scott R. Kohn, Scott B. Baden, Alan Edelman, Ryoichi Kawai, M. Elizabeth G. Ong, and John H. Weare. Scalable parallel numerical methods and software tools for material design. In *Proceedings of the Seventh SIAM Conference on Parallel Processing for Scientific Computing*, pages 219-224, San Francisco, CA, February 1995. SIAM.
- [6] James R. Chelikowsky, N. Troullier, K. Wu, and Y. Saad. Higher-order finite-difference pseudopotential method: An application to diatomic molecules. *Physical Review B*, 50(16):11355-11364, 1994.
- [7] K. Cho, T. A. Arias, J. D. Joannopoulos, and Pul K. Lam. Wavelets in electronic structure calculations. *Physical Review Letters*, 71(12):1808-1811, September 1993.
- [8] Stephen J. Fink, Scott B. Baden, and Scott R. Kohn. Flexible communication schedules for block structured applications. In *Third International Workshop on Parallel Algorithms for Irregularly Structured Problems*, Santa Barbara, California, August 1996.
- [9] Francois Gygi. Electronic-structure calculations in adaptive coordinates. *Physical Review B*, 48(16):11692-11700, 1993.
- [10] Francois Gygi and Giulia Galli. Real-space adaptive-coordinate electronic-structure calculations. *Physical Review B*, 52(4):R2229-R2232, 1995.
- [11] D. R. Hamann. Generalized norm-conserving pseudopotentials. *Physical Review B*, 40(5):2980-2987, 1989.
- [12] Leonard Kleinman and D. M. Bylander. Efficacious form for model pseudopotentials. *Physical Review Letters*, 48(20):1425-1428, 1982.
- [13] Scott Kohn and Scott Baden. A parallel software infrastructure for structured adaptive mesh methods. In *Proceedings of Supercomputing '95*, San Diego, California, December 1995.
- [14] W. Kohn and L.J. Sham. Self-consistent equations including exchange and correlation effects. *Physical Review*, 140(4A):A1133-A1138, 1965.
- [15] D. E. Longsine and S. F. McCormick. Simultaneous Rayleigh quotient minimization methods for  $Ax = \lambda Bx$ . *Linear Algebra and Its Applications*, 34:195-234, 1980.
- [16] Steve F. McCormick, editor. *Multilevel Adaptive Methods for Partial Differential Equations*. SIAM, Philadelphia, 1989.
- [17] Dahlia Remler and Paul Madden. Molecular dynamics without effective potentials via the Car-Parrinello approach. *Molecular Physics*, 70(6):921-966, 1990.
- [18] Ming-Wen Sung, Maria Elizabeth G. Ong, and John H. Weare. Direct minimization of the Kohn-Sham equations using preconditioned conjugate gradient methods, March 1995. American Physical Society March Meeting.
- [19] Steven R. White, John W. Wilkins, and Michael P. Teter. Finite-element method for electronic structure. *Physical Review B*, 39(9):5819-5833, March 1989.