HYGA: A HYBRID GEOMETRIC+ALGEBRAIC MULTIGRID SOLVER FOR WEIGHTED-RESIDUAL METHODS WITH HIERARCHICAL MESHES

CAO LU*†, XIANGMIN JIAO*‡, AND XIONGFEI WEI*

Abstract. We propose a *hybrid geometric+algebraic multigrid method*, or *HyGA*, for weighted residual methods with hierarchical basis functions. We present a unified derivation of restriction and prolongation operators for these methods. Based on this derivation, we propose a hybrid multigrid method *HyGA*, which combines a high-quality hierarchical mesh generator, a geometric multigrid solver with a multilevel weighted residual formulation, and an algebraic multigrid solver at the coarsest levels. Our method combines the rigor, high accuracy and runtime-and-memory efficiency of geometric multigrid with the robustness and flexibility of algebraic multigrid, and at the same time it is relatively easy to implement. We apply HyGA to weighted-residual finite element methods in both 2-D and 3-D, and present numerical experiments to demonstrate the effectiveness of HyGA compared with both geometric and algebraic multigrid methods.

Key words. multigrid method; weighted residual formulation; hierarchical meshes; restriction and prolongation operators

1. Introduction.

Multigrid methods are efficient and effective for solving large sparse linear systems, especially those from numerical discretizations of partial differential equations (PDEs). The multigrid methods are based on stationary iterative methods (such as Jacobi, Gauss-Seidel, and damped Gauss-Seidel), which serve as smoothers at different resolutions to smooth out errors at different frequencies. The residual and correction vectors are transferred back and forth between different resolutions to effectively accelerate the convergence of the stationary iterative methods.

Generally speaking, multigrid methods may be classified into *geometric multigrid* (*GMG*) and *algebraic multigrid* (*AMG*), each of which has advantages and disadvantages. The advantages of GMG include its better convergence with smooth solutions, better efficiency in terms of computational time, and also better efficiency in storage, especially with a matrix-free implementation. However, GMG tends to have difficulties for non-smooth problems or very coarse resolutions, and it is difficult to implement and to scale with unstructured meshes, especially for domains with complex topologies. The advantages of AMG include its simplicity and flexibility, and its robustness for non-smooth solutions and irregular domains. However, AMG tends to be more expensive, especially in terms of its setup stage. It also tends to generate denser matrices than GMG and cannot be easily implemented in a matrix-free fashion, so it has higher memory and computational costs. A more serious problem is that AMG has no guarantee on the condition numbers of the matrices at the coarse level, which may lead to non-convergence. Table 1.1 summarizes a comparison of GMG and AMG.

The goal of this paper is to develop a novel and general *hybrid geometric+algebraic multigrid method*, or *HyGA*, which will combine the advantages and overcome the disadvantages of the GMG and AMG, as shown in Table 1.1. We focus on sparse linear systems arising from a weighted residual method for linear PDEs over unstructured meshes. Linear PDEs cover a wide range of mathematical models, such as Poisson equations and heat equations. The weighted residuals are general discretization techniques, which can unify many commonly used finite elements and finite differences by defining different weighting functions. In addition, unstructured meshes are very general in resolving complex geometries. An important contribution of this paper is a new, unified derivation of restriction and prolongation operators for weighted residuals with hierarchical basis functions. This rigorous analysis coupled with the generality of our formulation will allow us to address a large class of linear systems arising from scientific and engineering applications.

At the algorithmic level, the main contribution of this paper is the hybrid multigrid solver *HyGA*, which combines a high-quality hierarchical mesh generator, a geometric multigrid solver with a multilevel weighted residual formulation, and an algebraic multigrid solver at the coarsest levels. Our proposed algorithm may be summarized as follows. Our hierarchical mesh generator starts from a good-quality coarse unstructured mesh that is sufficiently accurate for representing the topology and for reconstructing the geometry of the domain. It iteratively refines the mesh with guaranteed mesh quality (by uniform refinements) and geometric accuracy (by high-order boundary reconstruction). Because this refinement involves only local operations, it is highly efficient and scalable. We apply GMG with a multilevel weighted-residual formulation on these

[1]Department of Applied Mathematics & Statistics, Stony Brook University, Stony Brook, NY 11794, USA.
[2]Primary student author. Email: clu@ams.sunysb.edu.
[3]Corresponding author. Email: jiao@ams.sunysb.edu.

TABLE 1.1
Advantages and disadvantages of GMG and AMG. The goal of HyGA is to combine their advantages.

	geometric MG	algebraic MG	HyGA
set up stage	**inexpensive**	expensive	**inexpensive**
memory requirement	**low**	high	**low**
matrix free	**possible**	impossible	**possible at finer levels**
computational cost	**low**	high	**low**
matrix quality	**high**	no control	**high at finer levels**
convergence	**faster**	slower	**faster**
implementation	difficult	**simple**	**intermediate**
irregular domains	not robust	**robust**	**robust**

hierarchical meshes using our unified prolongation and restriction operators, which are coupled with PDE rediscretizations over coarse meshes. This combination enables efficient matrix-free implementations on the finer levels, and also effective control of the sparsity and the condition numbers of the resulting matrices. At the coarsest level, we employ the classical AMG, which allow us to resolve complex topologies easily and handle non-smooth solutions robustly. In addition, the relatively high setup cost and memory requirement of AMG become negligible due to the significantly reduced problem size. Therefore, our approach effectively combines the rigor, high accuracy and runtime-and-memory efficiency of GMG with the simplicity, robustness and flexibility of AMG, and at the same time it is relatively easy to implement. We will present numerical experiments to demonstrate the effectiveness of HyGA compared with both geometric and algebraic multigrid methods for 2-D and 3-D problems.

The remainder of the paper is organized as follows. Section 2 reviews some background and related work on geometric and algebraic multigrid solvers. Section 3 describes a general, multilevel weighted-residual formulation with hierarchical basis functions, and derive the restriction and prolongation operators for it. Section 4 describes our hybrid multigrid algorithm, with a focus on the generation of hierarchical meshes for complex geometries with curved boundaries, and the application of the multilevel weighted-residual formulation to hierarchical meshes. Section 5 presents some preliminary results with our method, and some comparisons with both geometric and algebraic multigrid methods. Section 6 concludes the paper with a discussion on future research directions.

2. Background and Related Work. We briefly review the multigrid method and introduce some notation, similar to those in textbooks such as [7] and [22]. Let us first consider a two-level approach for solving a linear system $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{b}$. Let $\boldsymbol{A}^{(1)} = \boldsymbol{A}$ be the coefficient matrix on the finer level, and $\boldsymbol{A}^{(2)}$ be that on the coarser level. We outline a two-level method as follows:

Pre-smoothing: starting from initial solution $\boldsymbol{u}_0$, run smoother on $\boldsymbol{A}^{(1)}\boldsymbol{u}^{(1)} = \boldsymbol{b}$ for a few iterations to obtain $\boldsymbol{u}^{(1)}$;

Restriction: compute residual vector $\boldsymbol{r}^{(1)} = \boldsymbol{b} - \boldsymbol{A}^{(1)}\boldsymbol{u}^{(1)}$ and $\boldsymbol{r}^{(2)} = \boldsymbol{R}\boldsymbol{r}^{(1)}$, where $\boldsymbol{R}$ is a *restriction matrix*;

Smoothing: construct *coarse-grid operator* $\boldsymbol{A}^{(2)}$ and run smoother on $\boldsymbol{A}^{(2)}\boldsymbol{s}^{(2)} = \boldsymbol{r}^{(2)}$ for a few iterations to obtain $\boldsymbol{s}^{(2)}$, starting from initial guess $\boldsymbol{s}^{(2)} = \boldsymbol{0}$;

Prolongation: compute correction vector $\boldsymbol{s}^{(1)} = \boldsymbol{P}\boldsymbol{s}^{(2)}$, where $\boldsymbol{P}$ is *prolongation matrix*;

Post-smoothing: run smoother on $\boldsymbol{A}^{(1)}\boldsymbol{s}^{(1)} = \boldsymbol{r}^{(1)}$ for a few iterations from prolongated $\boldsymbol{s}^{(1)}$, and update solution $\boldsymbol{u}^{(1)} \leftarrow \boldsymbol{u}^{(1)} + \boldsymbol{s}^{(1)}$.

Ideally, $\boldsymbol{A}^{(2)} \approx \boldsymbol{R}\,\boldsymbol{A}^{(1)}\boldsymbol{P}$, although this may be implemented in a matrix-free fashion without using matrix-matrix multiplications. The above procedure may repeat for many iterations. When more than two levels are used, the smoothing step is performed on the coarsest level, whereas the restriction, prolongation, pre- and post-smoothing operations are applied between adjacent levels in a global loop for all the levels, in a so-called V-cycle, W-cycle, or full-multigrid cycle.

There are many variants of multigrid/multilevel methods. The smoothers in different methods are similar, typically based on a stationary iterative method (such as some variant of Jacobi or Gauss-Seidel iterations) [5]. The main differences between different methods lie in the choices of the coarse-grid, restriction, and prolongation operators. Based on coarse-grid operators, multigrid methods are typically categorized into *geometric* (*GMG*) or *algebraic* (*AMG*). In our proposed approach, we use different coarse-grid operators (and

also different restriction and prolongation operators) for the upper and lower levels.

Geometric multigrid was first developed for structured meshes over regular domains; see e.g. [10]. They can be generalized to semi-structured meshes over regular domains (such as octree [17]). For applications with complex geometries, it is more desirable to use unstructured meshes [2, 15, 16]. A typical process of these methods is to start from a fine mesh and coarsen it repeatedly, which is a difficult process. A dual approach is to refine a coarse mesh repeatedly, such as in [14]. This approach is substantially simpler and more effective, but it is less general and requires tighter integration with mesh generation. Our hybrid method combines the latter approach with AMG.

Algebraic multigrid methods (AMGs) construct the coarse-grid operator directly from the matrix without explicit knowledge of the geometry. There are several variants of AMGs: classical algebraic multigrid [7], smoothed aggregation [21], element interpolation [6]. etc.. These methods are effective for solving problems involving anisotropy, discontinuity, or irregular domains, where GMG may be difficult to apply. Our work leverages the classical AMG, whose core idea is the complementarity of smoothing and coarse grid correction. In particular, it utilizes this principle to define smooth errors as the near null space of coefficient matrix. The coarsening and interpolations are chosen to ensure good approximation of fine grid smooth errors.

The multigrid methods have been implemented in some software packages, such as Hypre [8] and Trilinos [9] for AMG, and Prometheus [1] for GMG. These packages are effective when they are applicable, but they are often not robust enough as standalone solvers. For this reason, multigrid methods are often used as preconditioners of Krylov-subspace iterative methods (such as in PETSc [3]). One objective of our proposed approach is to combine GMG and AMG to deliver a more robust multigrid solver.

3. Multilevel Weighted Residual Methods. The weighted residual formulation is a general numerical technique for solving PDEs, of which both the Galerkin finite element methods and the finite difference methods can be viewed as special cases. We describe a general form of multilevel weighted residuals for linear partial differential equations over a hierarchy of basis functions. We will use this form to derive a geometric multigrid over hierarchical meshes in the next section.

3.1. A General Weighted Residual Formulation of Linear PDEs. For generality, let us consider an abstract but general form of linear, time-independent partial differential equations

$$\mathcal{P}\, u(\boldsymbol{x}) = f(\boldsymbol{x}), \tag{3.1}$$

with Dirichlet or Neumman boundary conditions, where $\mathcal{P}$ is a linear differential operator. In a weighted residual method,[1] given a set of test functions $\Psi(\boldsymbol{x}) = \{\psi_j(\boldsymbol{x})\}$, we obtain a linear equation for each ψ_j as

$$\int_\Omega \mathcal{P}\, u(\boldsymbol{x})\, \psi_j d\boldsymbol{x} = \int_\Omega f(\boldsymbol{x})\, \psi_j d\boldsymbol{x}, \tag{3.2}$$

and then the boundary conditions may be applied by modifying the linear system. In general, a set of basis functions $\Phi(\boldsymbol{x}) = \{\phi_i(\boldsymbol{x})\}$ is used to approximate u and f, where Φ and Ψ do not need to be equal.

The above formulation with (3.1) and (3.2) is general, and it unifies large classes of PDEs and of numerical methods. Examples of (3.1) include the Poisson equation $-\Delta u(\boldsymbol{x}) = f(\boldsymbol{x})$ and other linear elliptic problems. An example of (3.2) is the finite element methods, where the basis and test functions are piecewise Lagrange polynomials. When $\Phi = \Psi$, this reduces to the Galerkin method. Another example is the classical and the generalized finite difference methods [4], where the test functions are the Dirac delta functions at each node of a mesh, and the basis functions are piecewise polynomials (i.e., the basis functions of the polynomial interpolations/approximations in computing the finite-difference formulae). This unification will allow us to derive a general formulation of the restriction and prolongation operators for geometric multigrid methods.

For the convenience of notation, suppose Φ and Ψ are both column vectors composed of the basis functions, i.e. $\Phi = [\phi_1, \phi_2, \ldots, \phi_n]^T$ and $\Psi = [\psi_1, \psi_2, \ldots, \psi_n]^T$. Let $\boldsymbol{u}$ denote the vector of coefficients u_i associated with ϕ_i in the approximation of u, i.e., $u \approx \boldsymbol{u}^T\boldsymbol{\Phi} = \sum_i u_i\phi_i$, and similarly $f(\boldsymbol{x}) \approx \boldsymbol{f}^T\boldsymbol{\Phi} = \sum_i f_i\phi_i$. Then $\mathcal{P}\, u = \mathcal{P}\left(\boldsymbol{u}^T\boldsymbol{\Phi}\right) = \boldsymbol{u}^T\mathcal{P}\,\boldsymbol{\Phi}$, and from (3.2) we obtain a linear system

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{b}, \tag{3.3}$$

[1] The term "residual" appears in two different contexts in this paper: the residual of a linear system ($\boldsymbol{b} - \boldsymbol{A}\boldsymbol{u}$) and the residual of a PDE ($f(\boldsymbol{x}) - \mathcal{P}\, u(\boldsymbol{x})$). A "residual equation" is based on the former, and "weighted residual" is based on the latter.

where

$$A_{ij} = \int_\Omega \left(\mathcal{P}\,\phi_j(\boldsymbol{x})\right)\,\psi_i(\boldsymbol{x})d\boldsymbol{x} \quad \text{and} \quad b_i = \int_\Omega f(\boldsymbol{x})\psi_i(\boldsymbol{x})d\boldsymbol{x}.$$

For example, for the Poisson equation, we have $A_{ij} = \int_\Omega -\Delta\phi_j(\boldsymbol{x})\,\psi_i(\boldsymbol{x})d\boldsymbol{x} = \int_\Omega \boldsymbol{\nabla}\phi_j(\boldsymbol{x})\cdot\boldsymbol{\nabla}\psi_i(\boldsymbol{x})d\boldsymbol{x}$. The matrix $\boldsymbol{A}$ is the *stiffness matrix*, and $\boldsymbol{b}$ is the *force vector*. The equation (3.3) needs to be updated to incorporate the boundary conditions. The solution of (3.3) gives a vector $\boldsymbol{u}$ such that the residual $\boldsymbol{f}^T\boldsymbol{\Phi} - \mathcal{P}(\boldsymbol{u}^T\boldsymbol{\Phi})$ is orthogonal to the test functions ψ_j. If Φ is composed of Lagrange basis functions, such as in the finite element methods, $\boldsymbol{u}$ and $\boldsymbol{f}$ are composed of the nodal values of u and f on a mesh, respectively.

Remark: In finite-element methods, the integral in the left-hand side of (3.2) is often transformed into integrals of a lower-order differential operator multiplied (in an inner-product sense) with $\boldsymbol{\nabla}\psi_j(\boldsymbol{x})$. For example, in a finite element method where ψ_j vanishes along the boundary, $\int_\Omega -\Delta u(\boldsymbol{x})\,\psi_j d\boldsymbol{x} = \int_\Omega \boldsymbol{\nabla}u(\boldsymbol{x})\cdot\boldsymbol{\nabla}\psi_j d\boldsymbol{x}$. We omit this detail as it does not affect the derivations.

3.2. Weighted Residuals with Hierarchical Basis. In a multilevel context, we assume a hierarchy of basis functions $\Phi^{(k)}(\boldsymbol{x}) = \left[\phi_1^{(k)}(\boldsymbol{x}), \phi_2^{(k)}(\boldsymbol{x}), \ldots, \phi_{n_k}^{(k)}(\boldsymbol{x})\right]^T$, and similarly for the test functions $\Psi^{(k)}$. Without loss of generality, let us consider two levels first, and the construction will apply to adjacent levels in a multilevel setting. Suppose $\Phi^{(1)}$ and $\Phi^{(2)}$ correspond to the basis functions on the fine and coarse levels, respectively, and the functional space spanned by $\Phi^{(2)}$ is a subspace of $\Phi^{(1)}$. Let $\boldsymbol{R}_\Phi^{(1,2)}$ denote a *restriction matrix* of the functional space such that

$$\Phi^{(2)} = \boldsymbol{R}_\Phi^{(1,2)}\Phi^{(1)},$$

where $\boldsymbol{R}_\Phi^{(1,2)} \in \mathbb{R}^{n_2\times n_1}$. Similarly, let $\Psi^{(2)} = \boldsymbol{R}_\Psi^{(1,2)}\Psi^{(1)}$ with another restriction matrix $\boldsymbol{R}_\Psi^{(1,2)}$.

At the kth level, let $\boldsymbol{A}^{(k)}$ denote the matrix $\boldsymbol{A}$ in (3.3). A key question is the relationships between $\boldsymbol{A}^{(1)}$ and $\boldsymbol{A}^{(2)}$. To derive this, let us re-write $\boldsymbol{A}^{(k)}$ in the form of an integral of an outer product of $\Psi^{(k)}$ and $\mathcal{P}\,\Phi^{(k)}$, i.e.,

$$\boldsymbol{A}^{(k)} = \int_\Omega \Psi^{(k)}\left(\mathcal{P}\,\Phi^{(k)}\right)^T d\boldsymbol{x}.$$

Substituting $\Phi^{(2)} = \boldsymbol{R}_\Phi^{(1,2)}\Phi^{(1)}$ and $\Psi^{(2)} = \boldsymbol{R}_\Psi^{(1,2)}\Psi^{(1)}$ into it, we then obtain

$$\begin{aligned}\boldsymbol{A}^{(2)} &= \int_\Omega \left(\boldsymbol{R}_\Psi^{(1,2)}\Psi^{(1)}\right)\left(\mathcal{P}\,\boldsymbol{R}_\Phi^{(1,2)}\Phi^{(1)}\right)^T d\boldsymbol{x} = \boldsymbol{R}_\Psi^{(1,2)}\left(\int_\Omega \Psi^{(1)}\left(\mathcal{P}\,\Phi^{(1)}\right)^T d\boldsymbol{x}\right)\left(\boldsymbol{R}_\Phi^{(1,2)}\right)^T \\ &= \boldsymbol{R}_\Psi^{(1,2)}\boldsymbol{A}^{(1)}\left(\boldsymbol{R}_\Phi^{(1,2)}\right)^T. \end{aligned} \tag{3.4}$$

From Eq. (3.4), we conclude that the restriction matrix $\boldsymbol{R}$ and the prolongation matrix $\boldsymbol{P}$ in a two-level multigrid method for weighted residual methods should be

$$\boldsymbol{R} = \boldsymbol{R}_\Psi^{(1,2)} \text{ and } \boldsymbol{P} = \left(\boldsymbol{R}_\Phi^{(1,2)}\right)^T. \tag{3.5}$$

In particular, for Galerkin methods, $\boldsymbol{P} = \boldsymbol{R}^T = \left(\boldsymbol{R}_\Phi^{(1,2)}\right)^T$ is an interpolation matrix, which is a well-known result for Poisson equations [18] and [7, Chapter 10]. For finite difference methods with nested meshes, $\boldsymbol{P}$ is also an interpolation matrix, but $\boldsymbol{R} \neq \boldsymbol{P}^T$. However, $\boldsymbol{R}$ should be an $n_2 \times n_1$ permutation matrix (an injection operator), because the ψ_j are Dirac delta functions.

Our general result in (3.5) seems to be new, and we prove it as follows. Let $u^{(1)} = \left(\boldsymbol{u}^{(1)}\right)^T\boldsymbol{\Phi}^{(1)}$ denote the approximation of u with basis $\Phi^{(1)}$, and let $\boldsymbol{b}^{(1)}$ denote the right-hand vector in (3.3). The residual of linear system (3.3) with basis $\Phi^{(1)}$ is $\boldsymbol{r}^{(1)} = \boldsymbol{b}^{(1)} - \boldsymbol{A}^{(1)}\boldsymbol{u}^{(1)}$. Let $\boldsymbol{r}^{(2)} = \boldsymbol{R}_\Psi^{(1,2)}\boldsymbol{r}^{(1)}$, and then the residual equation with $\Phi^{(2)}$ is

$$\boldsymbol{A}^{(2)}\boldsymbol{s}^{(2)} = \boldsymbol{R}_\Psi^{(1,2)}\boldsymbol{r}^{(1)},$$

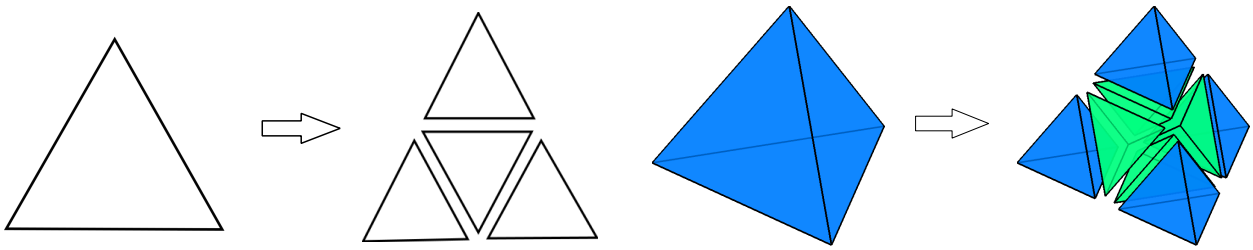


FIG. 4.1. *Illustration of refinement of triangle (left) and tetrahedron (right).*

where $\boldsymbol{s}^{(2)}$ is the correction vector with $\Phi^{(2)}$. Substituting (3.4) into it, we then obtain

$$\boldsymbol{R}_{\Psi}^{(1,2)}\boldsymbol{A}^{(1)}\left(\boldsymbol{R}_{\Phi}^{(1,2)}\right)^{T}\boldsymbol{s}^{(2)} = \boldsymbol{R}_{\Psi}^{(1,2)}\boldsymbol{A}^{(1)}\boldsymbol{s}^{(1)} = \boldsymbol{R}_{\Psi}^{(1,2)}\boldsymbol{r}^{(1)}, \tag{3.6}$$

where $\boldsymbol{s}^{(1)} = \boldsymbol{P}\boldsymbol{s}^{(2)} = \left(\boldsymbol{R}_{\Phi}^{(1,2)}\right)^{T}\boldsymbol{s}^{(2)}$ is the prolongated correction vector with $\Phi^{(1)}$. In the functional form, (3.6) can be rewritten as

$$\int_{\Omega}\Psi^{(2)}\left(\Phi^{(1)}\right)^{T}\left(\boldsymbol{u}^{(1)}+\boldsymbol{s}^{(1)}\right)\,d\boldsymbol{x} = \int_{\Omega}\Psi^{(2)}f(\boldsymbol{x})\,d\boldsymbol{x},$$

i.e., $\boldsymbol{s}^{(1)}$ gives a correction to $\boldsymbol{u}^{(1)}$ so that the updated residual of the PDE is orthogonal to all the test functions in $\Psi^{(2)}$.

From (3.4) and (3.5), we see that the matrix $\boldsymbol{A}^{(2)}$ is identical to the matrix $\boldsymbol{R}\boldsymbol{A}^{(1)}\boldsymbol{P}$ in a multilevel weighted-residual formulation with hierarchical basis functions for any linear partial differential equation in the form of (3.1). This allows us to discretize the PDE directly to obtain $\boldsymbol{A}^{(2)}$. This is advantageous as it avoids sparse matrix-matrix multiplications and allows a matrix-free implementation.

4. Hybrid Multigrid Methods. We now present our hybrid multigrid method utilizing our results on multilevel weighted residuals. Our hybrid multigrid method, called *HyGA*, combines a geometric multigrid solver with a few levels of hierarchical meshes, and an algebraic multigrid on the coarsest level. We will focus on finite element methods with linear simplicial elements in 2-D and 3-D (in particular, triangles and tetrahedra) in this paper, although the approach can be generalized to higher-order elements and to generalized finite difference methods, which we will explore our future research.

4.1. Generation of Hierarchical Meshes. We first describe the construction of hierarchical meshes. which are needed for our geometric multigrid based on multilevel weighted residuals.

Guaranteed-Quality Mesh Refinement. We construct hierarchical meshes through iterative mesh refinement, instead of mesh coarsening. In two dimensions, we start from a good-quality coarse triangular mesh. To generate a finer mesh, we subdivide each triangle into four equal sub-triangles that are similar to the original triangle, as illustrated in FIgure 4.1(left). The element quality (in terms of angles) is preserved under mesh refinement. To generate an ℓ-level hierarchical mesh, we repeat the refinement $(\ell-1)$ times. In three dimensions, we start from a good-quality coarse tetrahedral mesh, and subdivide each tetrahedron into eight sub-tetrahedra, as illustrated in FIgure 4.1(right). There are three different choices in the subdivisions. Any of these subdivisions will produce eight sub-tetrahedra of the same volume, so it does not introduce very poor-quality elements. Among these sub-tetrahedra, the four sub-tetrahedra incident on the original corner vertices are similar to the original tetrahedra. The four interior sub-tetrahedra may vary in shapes. We choose the subdivision that minimizes the edge lengths, as it tends to minimize the aspect ratio, defined as the ratio of the sum of squared edge lengths and the two-thirds root of the volume [13].

Treatment of Curved Boundaries. One of the main reasons why unstructured meshes are useful in practice is its flexibility to deal with complex geometries, especially those with curved boundaries. When generating hierarchical meshes, we need to respect the curved boundaries. We achieve this by projecting the newly inserted mid-edge points onto the curved geometry, as illustrated in Figure 4.2. The projection can be done either analytically if the geometry is known, or through a high-order reconstruction of the geometry

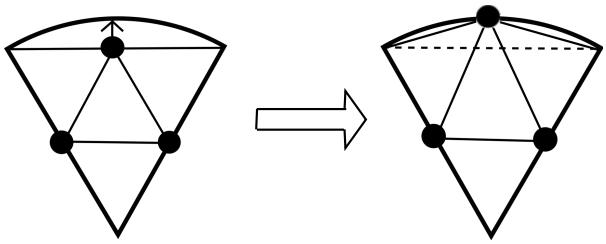


FIG. 4.2. *Illustration of the treatment of curved boundary by projecting inserted mid-edge points.*

[11, 12]. Note that if the mesh is too coarse at a concave region, some mesh smoothing may be needed to avoid inverted elements (i.e., elements with negative Jacobian) We omit this issue in this paper.

4.2. Prolongation and Restriction Operators. After we obtain the hierarchical meshes, the basis and test functions are determined on each level. In Galerkin finite elements, the basis functions $\Phi^{(k)}$ and test functions $\Psi^{(k)}$ are the same, and they are piecewise Lagrange polynomials. Assume the meshes are exactly nested, then the functional spaces on a coarser mesh is strictly a subspace of that on a finer mesh. Therefore, there is an exact restriction matrix $\boldsymbol{R}_{\Phi}^{(k,k+1)}$ of the functional space between levels k and $k+1$. As we have shown in section 3.2, for Galerkin finite elements the optimal prolongation operator is $\boldsymbol{P}^{(k)} = \left(\boldsymbol{R}_{\Phi}^{(k,k+1)}\right)^T$, and the optimal restriction operator is $\boldsymbol{R}^{(k)} = \boldsymbol{R}_{\Phi}^{(k,k+1)}$. With these definitions, the matrix $\boldsymbol{A}^{(k+1)}$ from discretizing the PDE over the $(k+1)$st grid is equivalent to $\boldsymbol{R}^{(k)}\boldsymbol{A}^{(k)}\boldsymbol{P}^{(k)}$.

In general, computing $\boldsymbol{R}_{\Phi}^{(k,k+1)}$ may be a daunting task. However for Lagrange basis functions over hierarchical meshes, $\boldsymbol{R}_{\Phi}^{(k,k+1)}$ is precisely the transpose of the interpolation matrix $\boldsymbol{I}^{(k+1,k)}$ of the nodal values from the $(k+1)$st level mesh to the kth level mesh. This may not be straightforward, because as noted in [22], such nodal interpolations may require proper scaling to produce the correct prolongation and restriction operators. In the following, we show that $\boldsymbol{R}_{\Phi}^{(k,k+1)} = \left(\boldsymbol{I}^{(k+1,k)}\right)^T$.

Without loss of generality, let $k=1$, and let $\tilde{u}^{(2)} = \left(\tilde{\boldsymbol{u}}^{(2)}\right)^T \boldsymbol{\Phi}^{(2)}$ denote the approximation of u with basis $\Phi^{(2)}$. Since $\Phi^{(2)} = \boldsymbol{R}_{\Phi}^{(1,2)}\Phi^{(1)}$, then

$$\tilde{u}^{(2)} = \left(\tilde{\boldsymbol{u}}^{(2)}\right)^T \boldsymbol{R}_{\Phi}^{(1,2)}\Phi^{(1)} = \left(\tilde{\boldsymbol{u}}^{(1)}\right)^T \Phi^{(1)},$$

where $\tilde{\boldsymbol{u}}^{(1)} = \left(\boldsymbol{R}_{\Phi}^{(1,2)}\right)^T \tilde{\boldsymbol{u}}^{(2)}$. At the same, in terms of nodal interpolation $\boldsymbol{I}^{(2,1)}$, because $\Phi^{(k)}$ are Lagrange basis functions,

$$\tilde{u}^{(2)} = \left(\boldsymbol{I}^{(2,1)}\tilde{\boldsymbol{u}}^{(2)}\right)^T \Phi^{(1)}.$$

Thus, $\boldsymbol{I}^{(2,1)}\tilde{\boldsymbol{u}}^{(2)} = \left(\boldsymbol{R}_{\Phi}^{(1,2)}\right)^T \tilde{\boldsymbol{u}}^{(2)}$ for any $\tilde{\boldsymbol{u}}$, so $\boldsymbol{R}_{\Phi}^{(1,2)} - \left(\boldsymbol{I}^{(2,1)}\right)^T = \boldsymbol{0}$.

In summary, for hierarchical Lagrange basis functions, $\boldsymbol{P}^{(k)} = \boldsymbol{I}^{(k+1,k)}$, i.e., interpolation matrix of nodal values from $(k+1)$st level to the kth level. In addition for the Galerkin finite element methods, $\boldsymbol{R}^{(k)} = \left(\boldsymbol{I}^{(k+1,k)}\right)^T$. One subtle point is that after we project the points onto curved boundaries, the elements is no longer nested, so the Lagrange basis functions are no longer strictly hierarchical. When constructing the prolongation and restriction matrices, we treat the elements as nested (in other words, creating the prolongation and restriction operators as if boundary projection did not occur). This is because that for linear elements, whose geometric errors are second order when approximating curved boundaries, this projection introduces second-order corrections to the positions between each pair of meshes. Therefore, the additional errors introduced by omitting the curved boundaries in the prolongation and restriction operators are in the same order as the truncation errors, so it would not affect the convergence rate.

4.3. Hybrid Geometric+Algebraic Multigrid. Our preceding formulation based on hierarchical basis functions is rigorous, efficient, and relatively easy to implement. However, it has one limitation: it requires a hierarchical mesh. In practice, it is reasonable to assume a small number of levels (such as two or three levels) in a hierarchical mesh, but we cannot expect to have too many levels. This can limit the scalability of GMG with hierarchical meshes alone.

To overcome this issue, we propose to use a hybrid geometric+algebraic multgird method, or *HyGA*. In this approach, we use geometric multigrid based on multilevel weighted residuals on the upper levels with the hierarchical mesh. On the coarsest level, we switch to an algebraic multigrid (AMG) method, and in particular the classical AMG. For completeness, we give a brief review of the classical AMG, Its coarsening process is performed by the following steps:
1) Choose a threshold value $0 < \theta \leqslant 1$ to define strong dependence.
2) Choose an independent set containing points, which strongly influences as many other points as possible.
3) Convert some of the fine grid points into coarse ones to satisfy interpolation requirement.

After coarse grid points and fine grid points are chosen, the prolongation operator is built by:

$$P\left(e_i\right) = \begin{cases} e_i & \text{if } i \text{ is in coarse mesh} \\ \sum w_{ij} e_j & \text{otherwise.} \end{cases}$$

The weights w_{ij} are given by

$$w_{ij} = -\frac{a_{ij} + \sum_{m \in D_i^s} \left(\frac{a_{im} a_{mj}}{\sum_{k \in C_i} a_{mk}} \right)}{a_{ii} + \sum_{n \in D_i^w} a_{in}},$$

where D_i^s, C_i and D_i^w are three sets partitioned by strong dependence. The prolongation operator is expressed in a matrix $\boldsymbol{P}$, and the restriction matrix $\boldsymbol{R}$ is defined as $\boldsymbol{P}^T$. The matrix on the coarse grid is then $\boldsymbol{RAP}$.

Note that in both GMG and AMG, the coefficient matrix on the coarse grid is equal to $\boldsymbol{RAP}$. The difference is that in GMG, we obtain the hierarchy through iterative mesh refinement and we compute the coefficient matrix by discretizing the PDE, which is more accurate and more efficient, whereas in AMG, the hierarchy is obtained through coarsening, which is more flexible. Compared to a pure GMG method, the hybrid approach provides us more flexibility in resolving irregular geometries. Compared to a pure AMG method, the hybrid approach can potentially deliver better convergence and higher efficiency, because at the finer levels, GMG is more accurate and less expensive. In the next section, we will present comparisons of GMG, AMG and HyGA to demonstrate the advantages of HyGA.

5. Numerical Experimentations. We present some numerical experimentations using HyGA. In particular, we solve some large-scale linear systems from finite-element methods for Poisson equations with Dirichlet boundary conditions in both 2-D and 3-D. Figure 5.1 shows the test geometries, both of which have curved boundaries and hence require unstructured meshes.

Let ℓ denote the number of levels of a hierarchical mesh. We solve the linear system using the following five strategies and compare their performances:
AMG($\ell+1$): classical AMG with $\ell+1$ levels, since AMG with ℓ levels performed poorly.
GMG(ℓ): GMG with multilevel weighted-residual formulation with ℓ levels.
HyGA(2,$\ell-2$): GMG on first two levels and classical AMG on $\ell-2$ coarse levels.
HyGA(3,$\ell-3$): GMG on first three levels and classical AMG on $\ell-3$ coarse levels.
HyGA(3,$\ell-2$): GMG on first three levels and classical AMG on $\ell-2$ coarse levels.
For all the tests, we use one cycle of full multigrid, followed by V-cycles. We use the Gauss-Seidel method as the smoothers, with ten iterations at the coarsest level. For the pre- and post-smoothing, we use two the Gauss-Seidel iterations for 2-D problems and four iterations for 3D problems.

5.1. Convergence Results in 2-D. For our 2-D test, we solve the Poisson equation

$$\Delta u(\boldsymbol{x}) = f(\boldsymbol{x}), \text{ where } f(\boldsymbol{x}) = -2\pi^2 \left(\sin(\pi x) + \sin(\pi y)\right)$$

with homogeneous boundary conditions on three quarters of a unit disk. Starting from a 2-D initial mesh generated using Triangle [19], we generated three meshes with different resolutions, by refining the initial

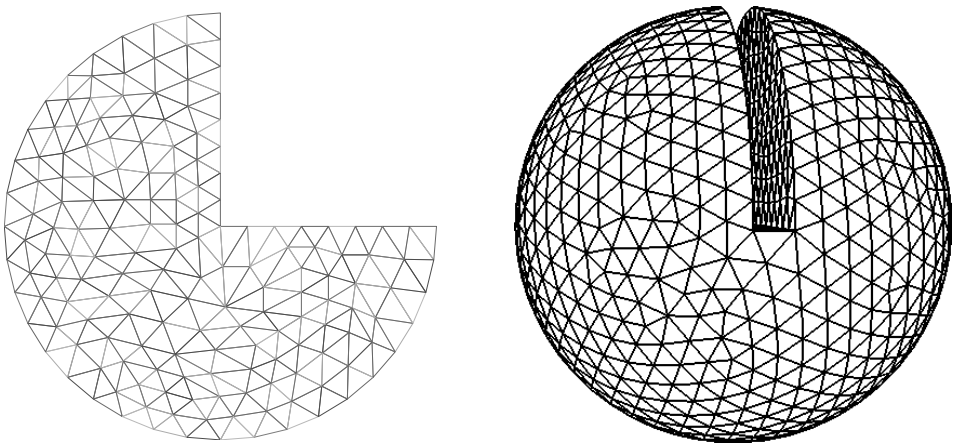
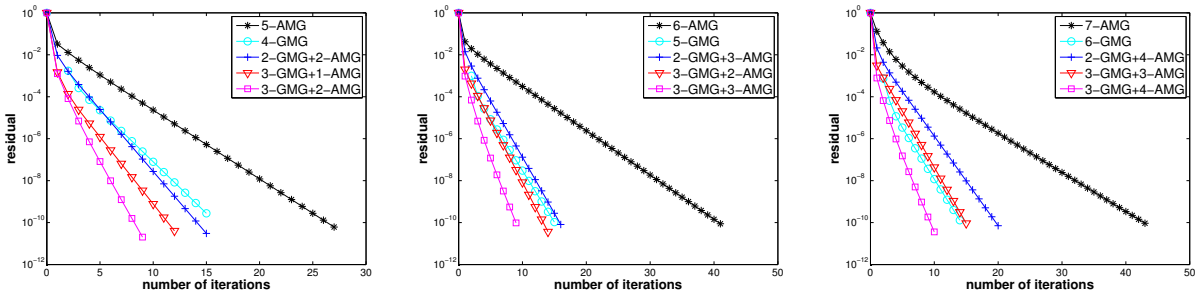


FIG. 5.1. *Coarsest initial meshes for 2-D (left) and 3-D tests (right) .*



(a) 8,906 unknowns with 4 levels. (b) 35,986 unknowns with 5 levels. (c) 144,674 unknowns with 6 levels.

FIG. 5.2. *Relative residual versus numbers of iterations for 2-D test cases.*

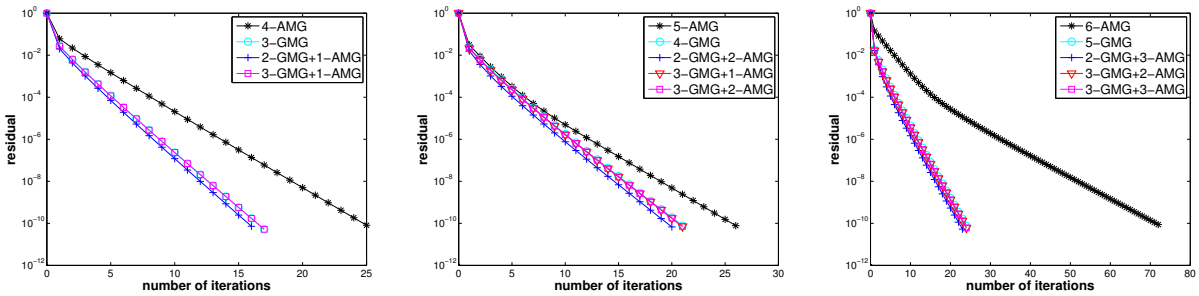
mesh in Figure 5.1(left) for three, four, and five times, to obtain hierarchical meshes with four, five, and six levels, respectively. After each refinement, we projected the newly inserted boundary points onto the curved boundary.

Figure 5.2 shows the convergence of these different strategies. From the plots, it can be seen that GMG(ℓ) converges a few times faster than AMG($\ell+1$). We do not show the results for AMG(ℓ), as it took about twice more iterations than AMG($\ell+1$). For AMG, we set its parameter $\theta = 0.25$ (c.f. section 4.3), as it seemed to deliver the best performance. Most importantly, for HyGA with only two or three levels of GMG, its convergence rate was comparable to GMG with the same total number of levels, and it was consistently faster when we added one additional AMG level at the coarse level. These results indicate that the finer levels are more critical to the overall convergence of multigrid methods, so it is advantageous to use GMG to achieve the best accuracy at finer levels. For the coarse levels, the slower converging but more flexible AMG suffices to ensure good overall convergence of HyGA. This alleviates the compilations of further mesh coarsening of a pure GMG, and also allows us to adapt the total number of levels easily.

5.2. Convergence Results in 3-D. For our 3-D test, we solve the Poisson equation

$$\Delta u(\boldsymbol{x}) = f(\boldsymbol{x}), \text{ where } f(\boldsymbol{x}) = -3\pi^2\left(\sin(\pi x) + \sin(\pi y) + \sin(\pi z)\right)$$

with homogeneous boundary conditions on a slotted sphere. We generate a tetrahedral mesh from the surface mesh shown in Figure 5.1(right) using TetGen [20], and then build three meshes with different resolutions by refining the initial mesh in Figure 5.1(right) for two, three and four times, to obtain hierarchical meshes with three, four and five levels, respectively. Let ℓ denote the number of levels for each case. We use similar strategies and settings as for the 2-D tests. However for AMG, it failed to converge with $\theta = 0.25$ for some of our test meshes. To ensure convergence, we set $\theta = 0.25$, 0.5, and 0.7 for the three cases, respectively. Figure 5.3 shows the convergence of the different strategies, which are qualitatively similar to the 2-D results.



(a) 31,841 unknowns with 3 levels. (b) 291,684 unknowns with 4 levels. (c) 2,484,807 unknowns with 5 levels.

FIG. 5.3. *Relative residual versus numbers of iterations for 3-D test cases.*

It can be seen that GMG(ℓ) converged faster than AMG($\ell+1$). At the same time, HyGA performed very similarly to GMG(ℓ), while being much more flexible and requiring fewer mesh levels.

5.3. Comparison of Runtimes. In terms of computational times, Table 5.1 compares the runtimes of these different strategies on a Mac Pro with two 2.4 GHz quad-core Intel Xeon processor and 24 GB of memory, running Mac OS X 10.7.5 with gcc 4.2. For AMG(ℓ), we show the runtimes of setup and solve steps separately. For the others, we show only the solve time since the setup time is negligible. As a reference, we also show the running times of MATLAB's built-in preconditioned conjugate gradient (PCG) with incomplete Cholesky factorization (ichol) as the preconditioner, which is the best in its class. It can be seen that HyGA(3,$\ell-2$) delivered the best overall performance in 2-D. It is about five times faster than the solve-step of AMG($\ell+1$), and more than an order of magnitude faster than PCG. For 3-D problems, HyGA schemes were much faster than AMG and PCG, and performed similarly to GMG.

TABLE 5.1

Timing results (in seconds) for AMG, GMG and HyGA. For references, times for PCG (with incomplete Cholesky preconditioner) are shown. Numbers in parentheses denote numbers of levels, which depend on parameter ℓ.

dim	test case		AMG($\ell+1$)		GMG(ℓ)	HyGA			PCG
	vertices	levels (ℓ)	setup	solve	solve	(2,ℓ-2)	(3,ℓ-3)	(3,ℓ-2)	(ichol)
2D	8,906	4	0.03	0.13	0.04	0.08	0.04	0.03	0.21
	35,986	5	0.15	0.74	0.16	0.36	0.20	0.14	1.27
	144,674	6	0.63	3.68	0.65	1.52	0.90	0.58	11.1
3D	31,841	3	0.37	1.79	0.38	0.42	0.38	0.39	0.44
	291,684	4	3.98	24.5	5.66	7.32	5.71	5.77	9.32
	2,484,807	5	28.5	509	58.3	89.5	59.2	59.8	186

6. Conclusions and Discussions. In this paper, we introduced a novel hybrid geometric+algebraic multigrid solver, HyGA, for weighted-residual methods with hierarchical basis functions, such as finite elements with hierarchical unstructured meshes. Our method combines a guaranteed-quality hierarchical mesh generator, a geometric multigrid solver at the finer levels, enabled by our unified derivation of prolongation and restriction operators for multilevel weighted residuals, and an algebraic multigrid solver at the coarse levels. We showed that this approach combines the high accuracy and efficiency of geometric multigrid with the robustness and flexibility of algebraic multigrid, to enable better convergence, while being relatively easy to implement. Our numerical experiments demonstrated the advantages of our proposed hybrid technique compared to the classical GMG and AMG separately.

HyGA has the potential to deliver an efficient, robust, and easy-to-implement multigrid solver for PDEs. A few issues have yet to be addressed to achieve full efficiency and robustness. In terms of efficiency, the implementation presented in this paper was serial. We are currently developing parallel implementations of HyGA. In terms of robustness, one potential issue is that AMG does not guarantee the conditioning of the matrices at the coarse levels. We have observed non-convergence of AMG occasionally, which may also affect the convergence of HyGA. We will address this issue in our future research.

REFERENCES

[1] M. Adams. Prometheus- scalable unstructured finite element solver. http://www.columbia.edu/ ma2325/prometheus/.

[2] M. F. Adams and J. Demmel. Parallel multigrid solver algorithms and implementations for 3D unstructured finite element problem. In *ACM/IEEE Proceedings of SC99: High Performance Networking and Computing*, Portland, Oregon, November 1999.

[3] S. Balay, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.0.0, Argonne National Laboratory, 2008.

[4] J. J. Benito, F. Ureña, and L. Gavete. *Leading-Edge Applied Mathematical Modeling Research*, chapter Chapter 7: The Generalized Finite Difference Method. Nova Science Publishers, Inc., 2008.

[5] J. Bramble and J. Pasciak. The analysis of smoothers for multigrid algorithms. *Math. Comput.*, 58:467–488, 1992.

[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). *SIAM J. Sci. Comput.*, 22(5):1570–1592, May 2000.

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

[8] R. Falgout, A. Cleary, J. Jones, E. Chow, V. Henson, C. Baldwin, P. Brown, P. Vassilevski, and U. M. Yang. Hypre home page. http://acts.nersc.gov/hypre, 2001.

[9] M. Gee, C. Siefert, J. Hu, R. Tuminaro, and M. Sala. Ml 5.0 smoothed aggregation user's guide. Technical Report SAND2006-2649, Sandia National Laboratories, Albuquerque, NM, 2006.

[10] W. Hackbusch. *Multi-Grid Methods and Applications*. Springer, 1985.

[11] X. Jiao and D. Wang. Reconstructing High-Order Surfaces for Meshing. In S. Shontz, editor, *Proceedings of the 19th International Meshing Roundtable*, pages 143–160. Springer Berlin Heidelberg, 2010.

[12] X. Jiao and D. Wang. Reconstructing high-order surfaces for meshing. *Engineering with Computers*, 28:361–373, 2012.

[13] X. Jiao, D. Wang, and H. Zha. Simple and effective variational optimization of surface and volume triangulations. *Engineering with Computers*, 27:81–94, 2011.

[14] D. Mavriplis and A. Jameson. Multigrid solution of the euler equations on unstructured and adaptive meshes. In *3rd Copper Mountain Conference on Multigrid Methods*, Copper Mountain, CO, 1987. Paper 23.

[15] D. J. Mavriplis. *Multigrid Techniques for Unstructured Meshes*. VKI Lecture Series VKI-LS 1995-02. Rhode-Saint-Genese, Belgium, 1995.

[16] D. J. Mavriplis. An assessment of linear versus nonlinear multigrid methods for unstructured mesh solvers. *Journal of Computational Physics*, 175(1):302 – 325, 2002.

[17] R. S. Sampath and G. Biros. A parallel geometric multigrid mmthod for finite elements on octree meshes. *SIAM J. SCI. COMPUT.*, 32:1361–1392, 2010.

[18] V. Shaidurov. *Multigrid methods for finite elements*. Springer, 1995.

[19] J. R. Shewchuk. Triangle: Engineering a 2d quality mesh generator and Delaunay triangulator. In *Lecture Notes in Computer Science*, volume 1148, pages 203–222, 1996.

[20] H. Si. Tetgen, a quality tetrahedral mesh generator and three-dimensional Delaunay triangulator v1.4, 2006.

[21] K. Stüben. A review of algebraic multigrid. *Journal of Computational and Applied Mathematics*, 128(1-2):281 – 309, 2001. Numerical Analysis 2000. Vol. VII: Partial Differential Equations.

[22] U. Trottenberg, C. W. Oosterlee, and A. Schuller. *Multigrid*. Academic Press, 2000.