NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 2013; **00**:1–18
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nla

A Hybrid Geometric+Algebraic Multigrid Method with Semi-Iterative Smoothers

Cao Lu[1], Xiangmin Jiao[1]* and Nikolaos Missirlis[2]

[1]*Department of Applied Mathematics and Statistics, Stony Brook University, Stony Brook, NY 11794, USA*
[2] *Department of Informatics and Telecommunications, Section of Theoretical Informatics, University of Athens, Panepistimiopolis, 157 84, Athens, Greece*

SUMMARY

We propose a multigrid method for solving large-scale sparse linear systems arising from discretizations of partial differential equations, such as those from finite element and generalized finite difference (GFD) methods. Our proposed method has the following two characteristics. First, we introduce a hybrid geometric+algebraic multigrid method, or *HyGA*, to leverage the rigor, accuracy and efficiency of geometric multigrid (GMG) for hierarchical unstructured meshes, with the flexibility of algebraic multigrid (AMG). Second, we introduce efficient smoothers based on the Chebyshev-Jacobi method for both symmetric and asymmetric matrices. We also introduce a unified derivation of restriction and prolongation operators for weighted residual formulations over unstructured hierarchical meshes, and apply it to both finite element and generalized finite difference methods. We present numerical results of our method for Poisson equations in both 2-D and 3-D, and compare our method against the classical GMG and AMG as well as Krylov-subspace methods. Copyright © 2013 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: Multigrid methods; geometric+algebraic multigrid; Chebyshev-Jacobi method; Krylov-subspace methods; finite element methods; generalized finite differences

1. INTRODUCTION

We consider the problem of efficient solution of large-scale sparse linear systems that arise from the discretizations of partial differential equations (PDEs). The multigrid methods, including *geometric multigrid* (*GMG*) and *algebraic multigrid* (*AMG*), are often the most efficient and effective methods for solving such linear systems [1, 2, 3]. Traditionally, these methods utilize stationary iterative methods (such as Jacobi, Gauss-Seidel, or successive over/under relaxation) to *smooth* out high-frequency errors, and accelerate the convergence of solution by transferring the residual and correction vectors across different resolutions (or levels) via the so-called *prolongation* and *restriction* operators. However, the GMG and AMG have their respective advantages and disadvantages, and different stationary iterative methods also have trade-offs in terms of effectiveness and ease of implementation and parallelization. Therefore, programers and users of multigrid methods often must face difficult choices among these variants. To make the matter worse, the interaction between the prolongation and restriction operators and the PDE discretizations introduces significant challenges and complexities to multigrid methods, especially for unstructured meshes.

*Correspondence to: Department of Applied Mathematics and Statistics, Stony Brook University, Stony Brook, NY 11794, USA. E-mail: jiao@ams.sunysb.edu.

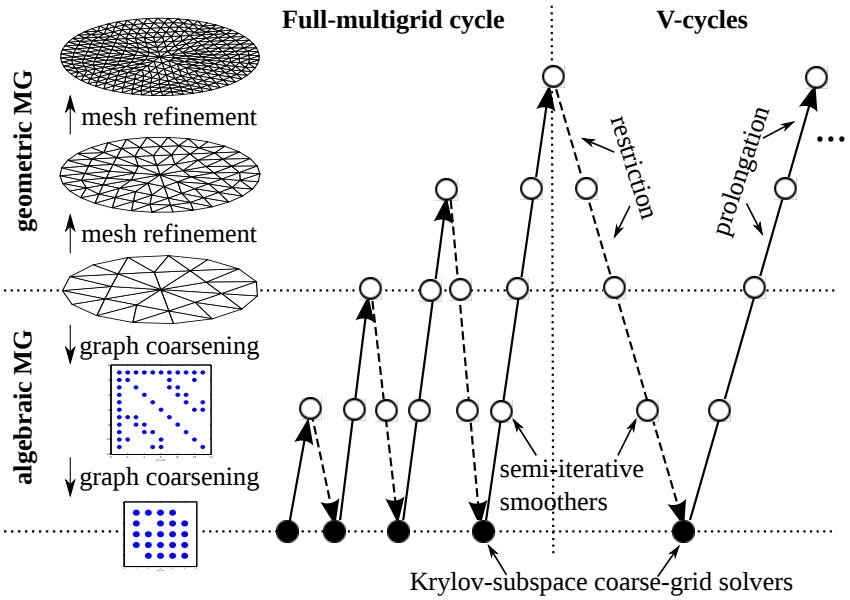


Figure 1. Schematic of hybrid geometric+algebraic multigrid method with semi-iterative smoothers.

In this paper, we propose a new framework of multigrid method for solving elliptic PDEs over hierarchical unstructured meshes. Figure 1 shows the schematic of our proposed method, which integrates various components in a systematic fashion. At a high-level, our overall approach may be summarized as follows. Our hierarchical mesh generator starts from a good-quality coarse unstructured mesh that is sufficiently accurate representation and allows accurate high-order reconstruction of the geometry [4]. It iteratively refines the mesh with guaranteed mesh quality (by uniform refinements) and geometric accuracy (by high-order boundary reconstruction). We apply GMG with a multilevel weighted-residual formulation on these hierarchical meshes, and employ the AMG at the coarsest level. We use a semi-iterative method, namely the Chebyshev-Jacobi method, as the smoother at both the GMG and AMG levels, and utilize Krylov-subspace methods as coarse-grid solvers.

Algorithmically, our proposed approach has two key characteristics. First, we introduce a hybrid geometric+algebraic approach, called *HyGA*, for better efficiency and robustness. From a practical point of view, this hybrid strategy is motivated by our observations that very large-scale meshes often have a few levels of mesh hierarchies. This is because it is increasingly common for high-resolution meshes to be generated by iteratively refining a moderate-resolution mesh on parallel computers [4]. Leveraging this inherent mesh hierarchy, HyGA can effectively couple the rigor, accuracy and runtime-and-memory efficiency of GMG at finer resolutions, with the flexibility and simplicity of AMG at coarser resolutions. Second, we introduce simple and effective semi-iterative smoothers based on the Chebyshev-Jacobi method, for both symmetric and asymmetric matrices. We demonstrate that these semi-iterative methods are comparable to Gauss-Seidel iterations as smoothers, while being as easy to implement and as scalable as Jacobi iterations.

Besides the above features, we also strive to establish a more systematic theoretical foundation of multigrid method for unstructured meshes. In particular, we introduce a unified derivation of

DOI: 10.1002/nla

restriction and prolongation operators for multilevel weighted-residual methods for linear PDEs with hierarchical basis functions. 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 element, finite difference, and finite volume methods by defining different weighting functions (see section 3 for details). In addition, unstructured meshes are very general in resolving complex geometries. Our formulation allows a more systematic derivation of geometric multigrid methods for finite element methods (FEMs) as well as generalized finite differences (GFDs) over unstructured meshes.

The remainder of the paper is organized as follows. Section 2 reviews some background and related work on geometric and algebraic multigrid methods, as well as Krylov-subspace methods. Section 3 presents a unified derivation of restriction and prolongation operators for multilevel weighted-residual formulation with hierarchical basis functions. Section 4 introduces our hybrid multigrid method for solving elliptic PDEs with hierarchical unstructured meshes. Section 5 introduces the Chebyshev-Jacobi method for symmetric and asymmetric matrices based on the theory of semi-iterative methods. Section 6 presents some numerical results with our method, as well as comparisons with AMG, GMG, and Krylov-subspace methods. Section 7 concludes the paper with a discussion on future research directions.

2. BACKGROUND AND RELATED WORK

2.1. Multigrid Methods

We briefly review the multigrid method and introduce some notation, similar to those in textbooks such as [1] and [2]. 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. A two-level method can be outlined as follows:

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

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

Coarse-solve: construct the *coarse-grid operator* $\boldsymbol{A}^{(2)}$ and solve $\boldsymbol{A}^{(2)}\boldsymbol{s}^{(2)} = \boldsymbol{r}^{(2)}$;

Prolongation: compute the correction vector $\boldsymbol{s}^{(1)} = \boldsymbol{P}\boldsymbol{s}^{(2)}$, where $\boldsymbol{P}$ is the *prolongation matrix,* and update solution $\boldsymbol{v}^{(1)} \leftarrow \boldsymbol{v}^{(1)} + \boldsymbol{s}^{(1)}$;

Post-smoothing: run the smoother on $\boldsymbol{A}^{(1)}\boldsymbol{u}^{(1)} = \boldsymbol{b}^{(1)}$ for a few iterations with initial guess $\boldsymbol{v}^{(1)}$.

Ideally, $\boldsymbol{A}^{(2)} \approx \boldsymbol{R}\,\boldsymbol{A}^{(1)}\boldsymbol{P}$, which defines an important connection among these different operators. The smoothers are typically based on stationary iterative methods, such as some variants of Jacobi or Gauss-Seidel iterations [5]. The above procedure may repeat for many iterations. A coarse-grid solver is employed at the coarsest level, and its choices include stationary iterative methods, Krylov-subspace methods, and direct methods. 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 main differences between different methods mostly lie in the choices of the coarse-grid, restriction, and prolongation operators. Based on the construction of coarse-grid operators, multigrid methods are typically categorized into *geometric* (*GMG*) or *algebraic* (*AMG*). Geometric multigrid was first developed for structured meshes over regular domains; see e.g. [6], and it also generalizes to semi-structured meshes over regular domains (such as octree [7]). For applications with complex geometries, it is more desirable to use unstructured meshes [8, 9, 10, 11]. A typical process of these methods is to start from a fine

DOI: 10.1002/nla

Table I. Advantages and disadvantages of GMG and AMG, along with the design goals for HyGA.

	geometric MG	algebraic MG	HyGA
memory requirement	**low**	high	**low**
operator complexity	**less costly**	more costly	**less costly**
matrix quality	**high**	less control	**high at finer levels**
user friendliness	less friendly	**more friendly**	**intermediate**
irregular domains	not robust	**robust**	**robust**

mesh and coarsen it repeatedly, which is a difficult process. A dual approach is to refine a coarse mesh repeatedly, such as in [12, 13]. This approach is substantially simpler and more effective, but it is less general and requires tighter integration with mesh generation. 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 [14], smoothed aggregation [15], element interpolation [16], etc. The core idea of the classical AMG 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 Prometheus [17] for GMG and Hypre [18] and Trilinos [19] for AMG.

The GMG and AMG have their own advantages and disadvantages, which we summarize in Table I. The main 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. It is also difficult to implement and to scale with unstructured meshes, especially for domains with complex topologies. The advantages of AMG include its generality and flexibility. In addition, they are more robust and effective for solving problems involving anisotropy or irregular domains. However, AMG tends to be more expensive. 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.

One of the goals of this paper is to develop a general *hybrid geometric+algebraic multigrid method*, or *HyGA*, to combine the advantages and overcome the disadvantages of the GMG and AMG. Our hybrid method utilizes the GMG over hierarchical unstructured meshes at the finer levels along with the classical AMG at the coarser levels. The GMG strategy utilizes mesh refinement and allows more effective control of the sparsity and the condition numbers of the resulting matrices. The classical AMG at the coarse level enables us to resolve complex topologies easily and handle non-smooth solutions more robustly. In addition, the relatively high computational cost and memory requirement of AMG become negligible due to the significantly reduced problem sizes at the coarse levels. Therefore, our approach can effectively combine the high accuracy and runtime-and-memory efficiency of GMG with the robustness and flexibility of AMG.

A similar idea was recently proposed in [20]. However, unlike their approach, our method uses unstructured triangle or tetrahedral meshes for geometric flexibility, and it uses Chebyshev-Jacobi iterations as smoothers. In addition, we present a detailed comparative study on different combinations of geometric and algebraic multigrid solvers.

2.2. *Krylov-subspace Methods*

The Krylov-subspace methods (KSMs) constitute another class of powerful and popular iterative techniques for solving large-scale linear systems [21] and for computing eigenvalues [22]. Some of the representative methods include conjugate gradient methods for symmetric and positive definite matrices, as well as Generalized Minimum RESidual (GMRES) for asymmetric matrices. The KSMs often require good preconditioners to be effective. Multigrid methods, especially AMG, are often used as the preconditioners, along with other methods such as stationary iterative methods,

DOI: 10.1002/nla

and incomplete LU factorization [23]. The focus of this paper is to develop effective standalone multigrid solvers. However, the KSMs are related to our approach in two aspects. First, although unlike stationary iterative methods, KSMs in general are not effective smoothers for multigrid methods, they are very effective and robust coarse-grid solvers. This is because KSMs tend to damp out the higher magnitude components in the errors, instead of damping the higher-frequency components. Second, the KSMs for estimating eigenvalues, such as Lanczos iterations, can be effective means for estimating parameters for semi-iterative methods, which we will use as the smoothers for the multigrid methods.

3. MULTILEVEL WEIGHTED RESIDUAL METHODS

The weighted residual formulation is a general numerical technique for solving PDEs, of which both the Galerkin finite element, finite volume, and 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

$$\mathcal{P}\, u(\boldsymbol{x}) = f(\boldsymbol{x}), \tag{1}$$

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

$$\int_\Omega \mathcal{P}\, u(\boldsymbol{x})\, \psi_j d\boldsymbol{x} = \int_\Omega f(\boldsymbol{x})\, \psi_j d\boldsymbol{x}, \tag{2}$$

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 (1) and (2) is general, and it unifies large classes of PDEs and of numerical methods. Examples of (1) include the Poisson equation $-\Delta u(\boldsymbol{x}) = f(\boldsymbol{x})$ and other linear elliptic problems. An example of (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 [24], 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). The finite volume methods are also examples, where the test functions are step functions, which have value one over the control volume. 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 $\mathcal{P}\, u = \mathcal{P}\left(\boldsymbol{u}^T \boldsymbol{\Phi}\right) = \boldsymbol{u}^T \mathcal{P}\, \boldsymbol{\Phi}$. From (2), we obtain a linear system

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{b}, \tag{3}$$

[†]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.

Prepared using nlaauth.cls
DOI: 10.1002/nla

where

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

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) needs to be updated to incorporate the boundary conditions. The solution of (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 (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, $\int_\Omega -\Delta u(\boldsymbol{x})\,\psi_j d\boldsymbol{x} = \int_\Omega \boldsymbol{\nabla}u(\boldsymbol{x}) \cdot \boldsymbol{\nabla}\psi_j d\boldsymbol{x}$. In the finite volume methods, the integral in the left-hand side of (2) is transformed into a boundary integral by utilizing Stokes's or Green's theorem. We omit these details as they do 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 function space spanned by $\Phi^{(2)}$ is a subspace of $\Phi^{(1)}$. Let $\boldsymbol{R}_\Phi^{(1,2)}$ denote a *restriction matrix* of the function 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). 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

$$\begin{aligned} \boldsymbol{A}^{(2)} &= \int_\Omega \left(\boldsymbol{R}_\Psi^{(1,2)}\Psi^{(1)}\right)\left(\mathcal{P}\,\boldsymbol{R}_\Phi^{(1,2)}\Phi^{(1)}\right)^T d\boldsymbol{x} = \boldsymbol{R}_\Psi^{(1,2)}\left(\int_\Omega \Psi^{(1)}\left(\mathcal{P}\,\Phi^{(1)}\right)^T d\boldsymbol{x}\right)\left(\boldsymbol{R}_\Phi^{(1,2)}\right)^T \\ &= \boldsymbol{R}_\Psi^{(1,2)}\boldsymbol{A}^{(1)}\left(\boldsymbol{R}_\Phi^{(1,2)}\right)^T. \end{aligned} \tag{4}$$

From (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

$$\boldsymbol{R} = \boldsymbol{R}_\Psi^{(1,2)} \text{ and } \boldsymbol{P} = \left(\boldsymbol{R}_\Phi^{(1,2)}\right)^T. \tag{5}$$

In particular with nested meshes, 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 [25] and [1, Chapter 10]. In addition, it is desirable for these interpolations to have the same order of accuracy as the numerical discretizations. For the classical and generalized finite difference methods, $\boldsymbol{P}$ is also an interpolation matrix, but $\boldsymbol{R}$ is not equal to $\boldsymbol{P}^T$ and instead is an $n_2 \times n_1$ permutation matrix (an injection operator) or scaled

DOI: 10.1002/nla

$\boldsymbol{P}^T$, because the ψ_j are Dirac delta functions. For finite volume methods, $\boldsymbol{R}$ is not equal to $\boldsymbol{P}^T$ either. Instead, it corresponds to a constant interpolation from a cell to its subcells.

We prove our result in (5) 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). The residual of linear system (3) with basis $\Phi^{(1)}$ is $\boldsymbol{r}^{(1)} = \boldsymbol{b}^{(1)} - \boldsymbol{A}^{(1)}\boldsymbol{u}^{(1)}$. Let $\boldsymbol{r}^{(2)} = \boldsymbol{R}_{\Psi}^{(1,2)}\boldsymbol{r}^{(1)}$. The residual equation with $\Phi^{(2)}$ is then

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

where $\boldsymbol{s}^{(2)}$ is the correction vector with $\Phi^{(2)}$. Substituting (4) into it, we then obtain

$$\boldsymbol{R}_{\Psi}^{(1,2)}\boldsymbol{A}^{(1)}\left(\boldsymbol{R}_{\Phi}^{(1,2)}\right)^T \boldsymbol{s}^{(2)} = \boldsymbol{R}_{\Psi}^{(1,2)}\boldsymbol{A}^{(1)}\boldsymbol{s}^{(1)} = \boldsymbol{R}_{\Psi}^{(1,2)}\boldsymbol{r}^{(1)}, \tag{6}$$

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, (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 (4) and (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 (1). This allows us to discretize the PDE directly to obtain $\boldsymbol{A}^{(2)}$.

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 particular, triangles and tetrahedra in 2-D and 3-D) as well as generalized finite differences.

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.

4.1.1. 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 2(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 2(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 [26].

DOI: 10.1002/nla

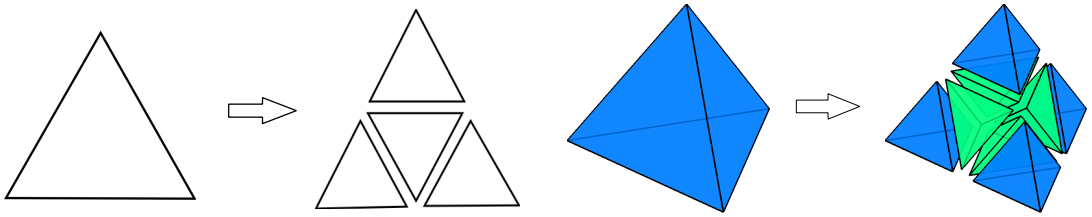


Figure 2. Illustration of refinement of triangle (left) and tetrahedron (right).

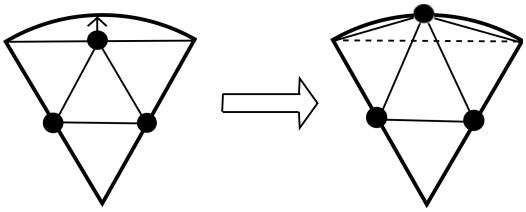


Figure 3. Illustration of the treatment of curved boundary by projecting inserted mid-edge points.

4.1.2. 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 3. The projection can be done either analytically if the geometry is known, or through a high-order reconstruction of the geometry [27, 4]. 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 function space 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 result may not be obvious, because as noted in [2], 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 $\Phi^{(2)} = \boldsymbol{R}_{\Phi}^{(1,2)}\Phi^{(1)}$, we have

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

DOI: 10.1002/nla

where $\tilde{\boldsymbol{u}}^{(1)} = \left(\boldsymbol{R}_{\Phi}^{(1,2)}\right)^T \tilde{\boldsymbol{u}}^{(2)}$. At the same time, because $\Phi^{(k)}$ are Lagrange basis functions, we have

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

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 element 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 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 for the finer levels with the hierarchical mesh, and use an algebraic multigrid (AMG) method, in particular the classical AMG, for the coarser levels. 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 $0 < \theta \leqslant 1$, known as the *strength parameter*, 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 $\boldsymbol{RAP}$.

4.4. Multilevel Generalized Finite Differences

Finite differences are well-established numerical methods for solving differential equations, but they were limited to structured meshes. The *generalized finite differences* (*GFDs*) extend the classical

DOI: 10.1002/nla

finite differences to unstructured meshes with high-order accuracy [24]. The GFD methods can be derived based on a *weighted least squares* (*WLS*) formulation locally over a weighted stencil at each point, which generalizes the classical interpolation-based finite differences. We have shown recently that the WLS can deliver the same order of accuracy as the interpolation-based approximations, while delivering superior stability, flexibility, and robustness for multivariate approximations over irregular unstructured meshes.

Our formulation of multi-level weighted-residual methods is applicable to GFDs over unstructured meshes. As described in section 3.2, the prolongation operator $\boldsymbol{P}$ for GFDs is also an interpolation matrix, but the restriction operator $\boldsymbol{R}$ is its scaled transpose with unit row sums, because the test functions ψ_j are Dirac delta functions. Applying these operators to adjacent grids in a multilevel method, we then obtain a geometric multigrid for GFD, where the coefficient matrices at all levels are obtained from re-discretizations with GFDs. Coupling with algebraic multigrid at the coarser levels, we then obtain HyGA for GFD.

Although the multigrid method for GFDs shares similar derivations as that for FEMs, the matrices in GFDs are asymmetric. Therefore, we must use a coarse-grid solver for asymmetric matrices, such as GMRES. In addition, the asymmetry also introduces additional challenges to the smoothers, which we consider in the next section.

5. CHEBYSHEV-JACOBI METHOD AND SMOOTHER

Our preceding section focused on the prolongation and restriction operators. In multigrid methods, another critical component is the smoother. The standard practice is to use stationary iterative methods, such as Gauss-Seidel iterations, as smoothers, because of their good smoothing properties. However, these stationary iterative methods tend to converge relatively slowly. The semi-iterative methods utilize polynomial acceleration to improve the convergence of the stationary iterative methods [28, 29], and thus from theoretical point of view they are appealing alternatives for smoothers. In addition, the semi-iterative methods, especially those based on polynomial accelerations of Jacobi iterations, are appealing from a practical point of view, because they are inherently parallel. In contrast, its Gauss-Seidel counterpart poses various difficulties especially for unstructured meshes [30].

In some previous studies such as [30], some variances of semi-iterative methods or polynomial accelerations have been proposed as smoothers. In this work, we propose the use of the *Chebyshev-Jacobi method*, or *CJ* for short, as a smoother. This method possesses a number of interesting properties as a standalone solver and as a smoother, for linear systems with both symmetric and asymmetric matrices under some reasonable assumptions.

5.1. Standalone Chebyshev-Jacobi Method

Our discussion on semi-iterative methods mostly follow [31]. For simplicity, let us first consider the case of symmetric matrices, for which all the eigenvalues are real. The semi-iterative method introduces a polynomial acceleration for a stationary iterative scheme

$$\boldsymbol{x}^{(n+1)} = \boldsymbol{G}\boldsymbol{x}^{(n)} + \boldsymbol{k}. \tag{7}$$

In particular, we are interested in accelerating the Jacobi iteration, where $\boldsymbol{G} = \boldsymbol{I} - \boldsymbol{D}^{-1}\boldsymbol{A}$ and $\boldsymbol{D} = \operatorname{diag}(\boldsymbol{A})$. To improve convergence of $\boldsymbol{x}^{(n)}$, we consider a new sequence $\left\{\boldsymbol{u}^{(n)}\right\}$ defined by the following linear combination

$$\boldsymbol{u}^{(n)} = \sum_{i=0}^{n} \alpha_{n,i}\boldsymbol{x}^{(n)}, \tag{8}$$

where the $\alpha_{n,i}$ are some constant coefficients. Under the assumption that $\boldsymbol{A}$ is symmetric, all the eigenvalues of $\boldsymbol{G}$ are real. Let $\{\lambda_i\}$ denote these eigenvalues, where λ_1 and λ_n are the largest

DOI: 10.1002/nla

and smallest eigenvalues of $\boldsymbol{G}$, respectively. In the CJ, the acceleration scheme has the following recurrence relation

$$\boldsymbol{u}^{(n+1)} = \rho_{n+1}\left(\gamma(\boldsymbol{G}\boldsymbol{u}^{(n)} + \boldsymbol{k}) + (1-\gamma)\,\boldsymbol{u}^{(n)}\right) + (1-\rho_{n+1})\,\boldsymbol{u}^{(n-1)}, \tag{9}$$

where the parameters are given by

$$\gamma = \frac{2}{2-\lambda_1-\lambda_n},\ \sigma = \frac{\gamma}{2}(\lambda_1-\lambda_n),\ \text{and}\ \rho_{n+1} = \begin{cases} 1 & n=0 \\ \left(1-\frac{1}{2}\sigma^2\right)^{-1} & n=1 \\ \left(1-\frac{1}{4}\sigma^2\rho_n\right)^{-1} & n\geqslant 2 \end{cases}. \tag{10}$$

The above formulas follow from the three-term recurrence of Chebyshev polynomials. For completeness, we summarize the derivation as follows. Let $\boldsymbol{u}^*$ denote the true solution of the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, and $\boldsymbol{e}^{(n)} = \boldsymbol{u}^{(n)} - \boldsymbol{u}^*$ be the error at the nth iteration. It is easy to show that

$$\boldsymbol{e}^{(n)} = \left(\sum_{i=0}^{n}\alpha_{n,i}\boldsymbol{G}^i\right)\boldsymbol{e}^{(0)} = P_n\left(\boldsymbol{G}\right)\boldsymbol{e}^{(0)},$$

where

$$P_n\left(\boldsymbol{G}\right) = \alpha_{n,0}\boldsymbol{I} + \alpha_{n,1}\boldsymbol{G} + \cdots + \alpha_{n,n}\boldsymbol{G}^n$$

is an nth order matrix polynomial, and $P_n\left(1\right) = 1$. The convergence of the CJ depends on the spectral radius $\rho\left(P_n\left(\boldsymbol{G}\right)\right)$. Note that

$$\rho\left(P_n\left(\boldsymbol{G}\right)\right) \leqslant \max_{\lambda_n\leqslant x\leqslant\lambda_1}\left|P_n\left(x\right)\right|.$$

Thus we want to find a polynomial that its maximum absolute value over the spectrum of $\boldsymbol{G}$ is minimum. It can be shown that the unique optimal polynomial is given by [31]

$$P_n\left(x\right) = T_n\left(\frac{2x-\lambda_1-\lambda_n}{\lambda_1-\lambda_n}\right)\Big/\,T_n\left(\frac{2-\lambda_1-\lambda_n}{\lambda_1-\lambda_n}\right),$$

where $T_n\left(x\right)$ is the nth order Chebyshev polynomial. Using the three-term recurrence of Chebyshev polynomials T_n, we then obtain (9).

5.1.1. Convergence of Chebyshev-Jacobi Method for Symmetric Matrices. An important property of the CJ is that it is guaranteed to converge if all the eigenvalues of $\boldsymbol{G}$ are real and less than 1 [31], which is satisfied if the matrix $\boldsymbol{A}$ is symmetric positive definite. Hence, the CJ significantly robustifies the Jacobi iterations, as it is guaranteed to converge when $\boldsymbol{A}$ is symmetric and positive definite, whereas the Jacobi iterations may diverge in this setting.

In addition, the CJ also significantly improves the convergence rate of Jacobi iterations. More precisely, it can be shown [31] that

$$\rho\left(P_n\left(\boldsymbol{G}\right)\right) = \frac{2r^{n/2}}{\left(1+r^n\right)},\quad \text{where } r = \frac{1-\sqrt{1-\sigma^2}}{1+\sqrt{1-\sigma^2}}, \tag{11}$$

and the asymptotic rate of convergence for CJ is given by

$$R_\infty\left(P_n\left(\boldsymbol{G}\right)\right) = -\frac{1}{2}\log r. \tag{12}$$

In comparison, the asymptotic rate of convergence for Jacobi iterations is given by

$$R_\infty\left(\boldsymbol{G}_\gamma\right) = -\log\sigma, \tag{13}$$

DOI: 10.1002/nla

where $\boldsymbol{G}_\gamma = (1-\gamma)\,\boldsymbol{I} + \gamma \boldsymbol{G}$ is the iteration matrix of weighted Jacobi iterations with a weight γ defined in (10). Combining (12) and (13), it can be shown that

$$R_\infty\left(P_n\left(\boldsymbol{G}\right)\right) \sim \sqrt{2\,R_\infty\left(\boldsymbol{G}_\gamma\right)} \text{ as } \sigma \to 1^-.$$

In words, the CJ in general can result in an order of magnitude improvement in the rate of convergence compared to Jacobi iterations.

The robustness and efficiency of the CJ make it an appealing method for solving linear systems, and they also motivate us to consider it as a smoother. However, a subtle issue of the CJ is that it requires good estimations of extreme eigenvalues for optimal convergence. Let λ_n be the smallest eigenvalue and $\lambda_n^{'}$ be an estimate of λ_n. Fortunately, as shown in [31], the estimation of the smallest eigenvalue λ_n does not affect the convergence much, as long as $\lambda_n^{'} \leq \lambda_n$. More specifically, if the following condition is satisfied

$$0 \leqslant \lambda_n - \lambda_n^{'} \leqslant 0.1 \max\{|\lambda_n|, 1\},$$

then a non-optimal $\lambda_n^{'}$ will cause only up to 4% increase in the number of iterations. Therefore, we do not need a very accurate estimate for λ_n. However, if $\lambda_n^{'} > \lambda_n$, divergence may occur.

On the other hand, when used as a standalone solver, the convergence of CJ is sensitive to the estimation of the largest eigenvalue λ_1. In addition, if we underestimate λ_1, the expected number of iterations would increase much more than if we overestimate λ_1 by an equal amount, so one needs to estimate an upper bound of λ_1. Fortunately, this situation will be much less an issue when CJ is used as a smoother, as we discuss in section 5.2.

5.1.2. Generalization to Asymmetric Matrices. In PDE discretizations, the matrix $\boldsymbol{A}$ is often asymmetric, due to either complex boundary conditions or the asymmetry in the stencils of the numerical discretizations (such as high-order finite differences or generalized finite differences). In this case, the matrix $\boldsymbol{A}$ and in turn the iteration matrix $\boldsymbol{G}$ may have complex eigenvalues, and the situation is somewhat more complicated. However, the matrix $\boldsymbol{A}$ is typically "nearly symmetric," because the symmetry is only due to the boundary effect or numerical discretization errors.

Assume the eigenvalues of $\boldsymbol{G}$ are bounded by the following ellipse

$$\frac{(x-d)^2}{a^2} + \frac{y^2}{b^2} = 1,$$

where the constants a and b are real, and

$$a + d < 1.$$

If $a \geq b$, $c = \sqrt{a^2 - b^2}$ is also real, then the unique polynomial that minimizes the maximum spectrum over the ellipse is given by [32]

$$P_n\left(z\right) = T_n\left(\frac{z-d}{c}\right) \Big/ T_n\left(\frac{1-d}{c}\right),$$

where $T_n\left(z\right)$ is the Chebyshev polynomial. Using the three-term recurrence of T_n, we obtain the acceleration scheme same as (9), except that the parameters γ and σ are now given by

$$\gamma = \frac{1}{1-d} = \frac{2}{2-(d+a)-(d-a)}, \text{ and } \sigma = c\gamma = \gamma\sqrt{a^2-b^2}. \tag{14}$$

These parameters are consistent with (10) when the eigenvalues are all real, where $\lambda_1 = d + a$, $\lambda_n = d - a$, and $c = a = (\lambda_1 - \lambda_n)/2$. In addition, if the imaginary parts of all the eigenvalues of $\boldsymbol{G}$ are small (i.e., $b \ll a$), then $c \approx a$, and thus

$$\gamma \approx \frac{2}{2 - \mathrm{Re}(\lambda_1) - \mathrm{Re}(\lambda_n)}, \text{ and } \sigma \approx \frac{\gamma}{2}(\mathrm{Re}(\lambda_1) - \mathrm{Re}(\lambda_n)).$$

Therefore, if the matrix is nearly symmetric (more precisely if the imaginary parts of the eigenvalues are small), we can approximate the parameters γ and σ as in the symmetric case from the extreme eigenvalues of $\boldsymbol{D}^{-1}\left(\boldsymbol{A} + \boldsymbol{A}^T\right)/2$.

DOI: 10.1002/nla

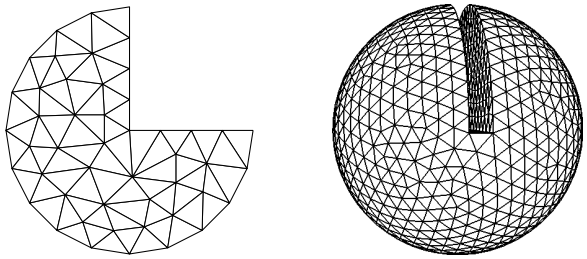


Figure 4. Coarsest initial meshes for 2-D (left) and 3-D tests (right) .

5.2. Chebyshev-Jacobi Method as Smoother

We now analyze the applicability of the CJ as a smoother. The key issue is the estimation of the eigenvalues λ_1 and λ_n and in turn γ and σ. As in the case for standalone CJ, it is not necessary to obtain an accurate estimation of the smallest eigenvalue λ_n of the iteration matrix $\boldsymbol{G}$. In practice, we can simply apply a few Lanczos iterations to get an estimation of the largest eigenvalue of $\boldsymbol{D}^{-1}\boldsymbol{A}$, say t, and then obtain $\lambda_n \approx 1 - t$. If $\boldsymbol{A}$ is asymmetric, instead of $\boldsymbol{D}^{-1}\boldsymbol{A}$, it suffices to estimate the largest eigenvalue of $\boldsymbol{D}^{-1}\left(\boldsymbol{A} + \boldsymbol{A}^T\right)/2$, whose eigenvalues are all real.

For the estimation of λ_1, while it is important for the standalone CJ, it does not need to be very accurate when CJ is used as a smoother. This is because for solving elliptic PDEs, the smoothers only need to damp high-frequency modes. Assuming $\boldsymbol{A}$ is symmetric, these modes typically correspond to the larger eigenvalues of $\boldsymbol{D}^{-1}\boldsymbol{A}$ and the smaller eigenvalues of $\boldsymbol{I} - \boldsymbol{D}^{-1}\boldsymbol{A}$. Therefore, an inaccurate estimation of λ_1 would not undermine the smoothing effect of CJ, as long as estimations satisfy $\lambda_n^{'} < \lambda_1^{'} < 1$, where $\lambda_1^{'}$ denotes the estimation of λ_1. In fact, we observed that an accurate λ_1 does not produce the best smoothing effect, although it gives optimal convergence of the standalone CJ. In our experiments, we have found that it works well to use $\lambda_1^{'} \approx 2/3$ for 2-D problems and $\lambda_1^{'} \approx 0.9$ for 3-D problems.

6. NUMERICAL EXPERIMENTATIONS

In this section, we present some numerical experimentations using HyGA with CJ, and also assess the effectiveness of our method against some other alternatives. For our benchmark problem, we use some sparse linear systems from finite elements or generalized finite differences for Poisson equations with Dirichlet boundary conditions. Figure 4 shows the test geometries for our problem, both of which have irregular and curved boundaries and hence require unstructured meshes.

We solve the linear system using the following four strategies and compare their performances:

AMG: classical AMG.

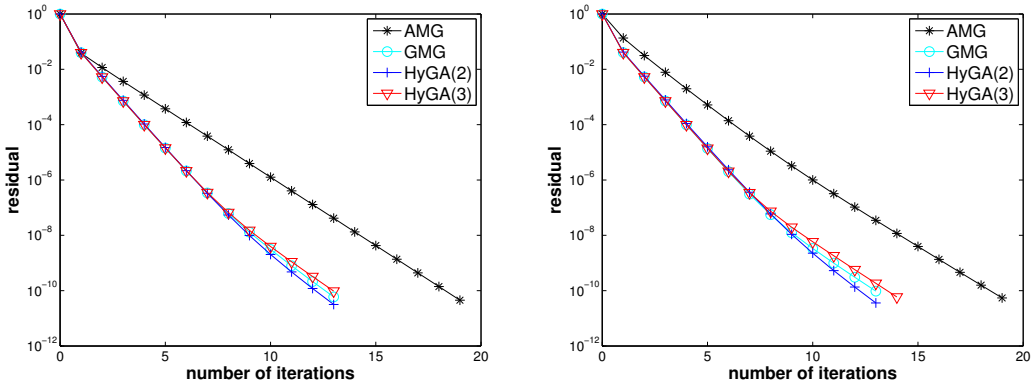
GMG: GMG with multilevel weighted-residual formulation.

HyGA(2)**:** GMG on first two levels and classical AMG on coarser levels.

HyGA(3)**:** GMG on first three levels and classical AMG on coarser levels.

For all the tests, we use one cycle of full multigrid, followed by V-cycles. On the coarsest level, conjugate gradient or GMRES is used to obtain sufficient accuracy. In terms of smoothers, we use 2 Chebyshev-Jacobi iterations for 2D problems and 4 for 3D problems in GMG and HyGA. In AMG, we use Gauss-Seidel method with the same number of iterations as it is more effective. The tolerance is 10^{-10} measured by the relative 2-norm of residuals.

DOI: 10.1002/nla



(a) 35,986 unknowns with 5 levels. (b) 144,674 unknowns with 6 levels.

Figure 5. Relative residual versus numbers of iterations for 2-D test cases.

6.1. *Convergence Results for 2-D Finite Elements*

Our first test case is the 2-D finite element discretization for 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 depicted in Figure 4(left). Starting from a 2-D initial mesh generated using Triangle [33], we generated three meshes with different resolutions, by refining the initial mesh in Figure 4(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 shows the convergence for the five- and six-level meshes with different strategies. For AMG we choose strength of connection θ (c.f. section 4.3) to be 0.25 as it seems to give the best performance. From these results, it can be seen that GMG converges a bit faster than AMG. For HyGA with only two or three levels of GMG, its convergence rate was comparable to GMG. These results indicate that the finer levels are critical to the overall convergence of multigrid methods. Therefore, it is advantageous to use GMG to achieve the best accuracy at finer levels. In addition, as we observe in section 6.3, GMG at the finer levels tend to produce coarser operators and hence better efficiency. On coarser levels, the more slowly converging but more flexible AMG suffices to ensure good overall performance of HyGA. This alleviates the complications of further mesh coarsening of a pure GMG and reduces the computational cost of a pure AMG while maintaining good convergence.

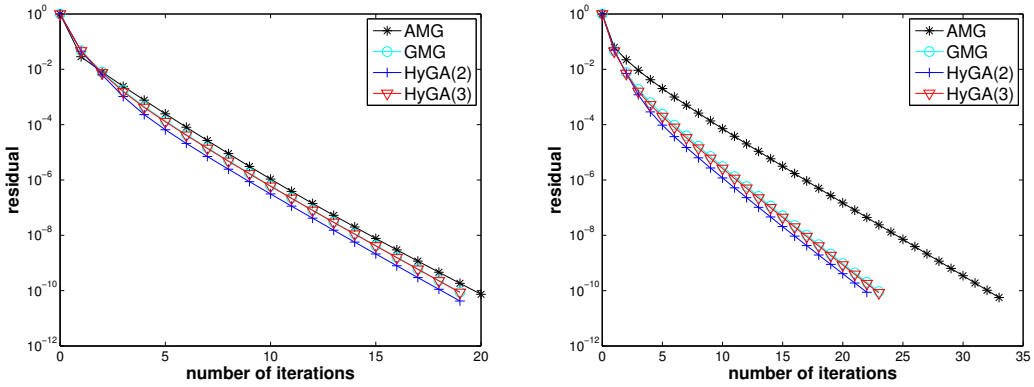
6.2. *Convergence Results for 3-D Finite Elements*

Our second test case is the 3-D finite-element discretization for 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 depicted in Figure 4(right). We generated a tetrahedral mesh from the surface mesh using TetGen [34], and then build two meshes with different resolutions by refining the initial mesh in Figure 4(right) for three and four times, to obtain hierarchical meshes with four and five levels, respectively. We use similar strategies and settings as for the 2-D tests. However for AMG, the parameter θ is chosen to be 0.65 and the value decreases in lower levels to achieve good performance. Figure 6 shows the convergence of different strategies, which are qualitatively similar to the 2-D results. It can be seen that GMG converged faster than AMG. At the same time, HyGA performed nearly identically as GMG, while being much more flexible. Moreover, the performance of HyGA was about the same on the 2D and 3D mesh, while the performance of AMG was more sensitive to the choices of parameters.

DOI: 10.1002/nla



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

Figure 6. Relative residual versus numbers of iterations for 3-D test cases.

6.3. Comparison of Computational Costs

In terms of computational times, Table II compares the runtimes of different strategies for FEMs 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 all strategies, we show only the solve time since the setup times were less significant. 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. Similar to the convergence result, the performance of hybrid schemes are comparable to GMG with only two or three geometric levels. In addition, GMG and HyGA are faster than AMG even when their convergence rates are close. The reason is the higher complexity of the coarse-grid operators of the AMG, which is evident from Figure 7, where we compare the operator complexities of strategies on several levels for the 3D 5-level mesh. From the charts we observe that GMG produces much smaller and more sparse matrices than AMG does. The operator complexity of HyGA is also moderate, as GMG is used on the finest levels. Since the computational time is dominated by smoothing, which is affected by matrix sparsity, the performances of GMG and HyGA overall are much better.

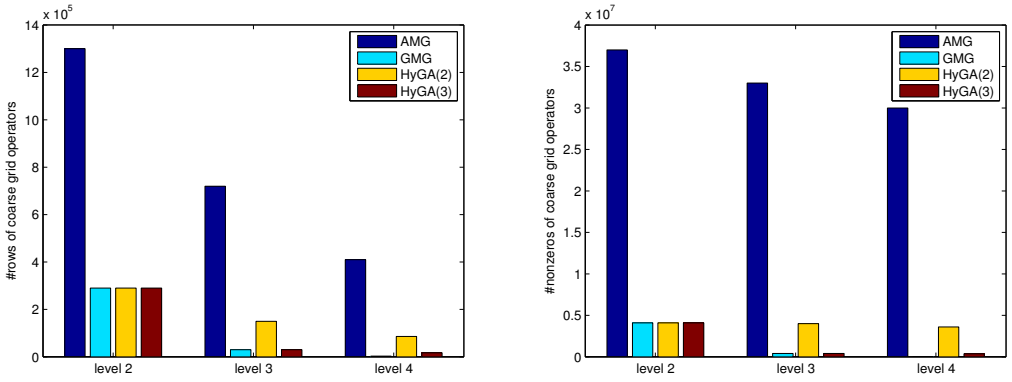
Table II. Timing results (in seconds) for AMG, GMG and HyGA. For references, times for PCG (with incomplete Cholesky preconditioner) are shown.

dim	#rows in the system	AMG	GMG	HyGA(2)	HyGA(3)	IC-PCG
2-D	8,906	0.12	0.04	0.08	0.04	0.21
	35,986	0.37	0.14	0.25	0.17	1.27
	144,674	1.43	0.61	1.14	0.79	11.1
3-D	291,684	30	5.26	6.67	5.22	9.32
	2,484,807	397	58.3	85.8	61.2	186

6.4. Application to Asymmetric Matrices

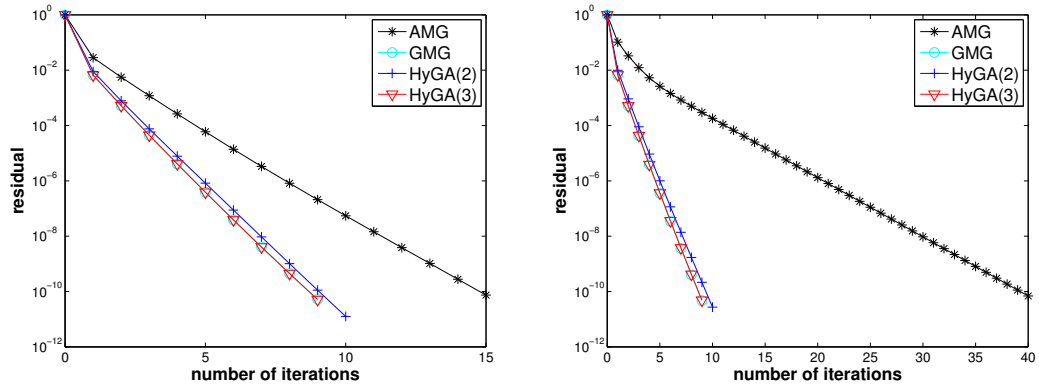
To test our method for asymmetric matrices, we applied HyGA to GFDs using the same FEM meshes for the 2-D geometry in Figure 4(left). As in the 2D test case, we use 2 iterations of presmoothing and post smoothing iterations. For the parameters in CJ, we chose $\lambda_1^{'}$ to be $2/3$ as in the FEM case. We obtained $\lambda_n^{'}$ by estimating the largest eigenvalue of $\boldsymbol{D}^{-1}\left(\boldsymbol{A}+\boldsymbol{A}^T\right)/2$. The resulting estimation turned out to be very accurate. For example, for the 2-D 5-level mesh, the smallest real part of λ_n is ≈ -0.746, while the estimated $\lambda_n^{'}$ is $1-1.747=-0.747$, whose accuracy is more than sufficient.

DOI: 10.1002/nla



(a) number of rows in the coarse grid operators. (b) number of nonzeros in the coarse grid operators.

Figure 7. Number of rows and nonzeros in the coarse grid operators of different strategies on levels 2, 3, and 4 for the 3-D 5 level mesh.



(a) 35,986 unknowns with 5 levels. (b) 144,674 unknowns with 6 levels.

Figure 8. Relative residual versus numbers of iterations for 2-D GFD tests.

Figure 8 shows the convergence results for GFDs. These results are similar to the FEM case: HyGA is nearly identical to GMG, and they are both faster than AMG.

6.5. *Assessment of Chebyshev-Jacobi Smoother*

We now assess the effectiveness of the CJ as a smoother in HyGA. We first investigate the influence of parameters $\lambda_n^{'}$ and $\lambda_1^{'}$, and then compare the performances of the CJ versus lexicographic Gauss-Seidel iterations.

6.5.1. Influences of Parameters. To evaluate the influence of parameters, we consider the 2-D Poisson equation with a 5-level mesh and 3-D Poisson equation with a 4-level mesh as described in the previous subsection. We computed a reference value for λ_n using MATLAB's eigs function with a relative tolerance of 10^{-5}. In these test cases, $\lambda_n < 0$. We compare the numbers of iterations required by the multigrid method for the cases of $\lambda_n^{'}$ equal to the optimal value λ_n, a loose bound $1.2\lambda_n$, and the very loose bound -2. In addition, we also test the case where $\lambda_n^{'} = 0.8\lambda_n$, which violates the requirement $\lambda_n^{'} \leq \lambda_n$ and hence the CJ may diverge. In all these cases, we set $\lambda_1^{'}$ to $2/3$ for 2-D and 0.9 for 3-D.

Table III shows the numbers of multigrid iterations of HyGA with these parameters. It can be seen that the convergence of CJ is insensitive to λ_n as long as $\lambda_n^{'} \leq \lambda_n$. Even a naive loose estimate

DOI: 10.1002/nla

Table III. Sensitivity of $\lambda_n^{'}$ for 2-D and 3-D test cases.

$\lambda_n^{'}$	number of multigrid iterations	
	2-D	3-D
λ_n	14	19
$1.2\lambda_n$	14	21
-2	16	23
$0.8\lambda_n$	47	diverged

Table IV. Comparison of Chebyshev-Jacobi and Gauss-Seidel as smoothers. All times are in seconds.

Tests	Chebyshev-Jacobi		Gauss-Seidel	
	iterations	time	iterations	time
2-D 5-level	14	0.17	9	0.11
2-D 6-level	14	0.79	10	0.50
3-D 4-level	19	5.24	21	6.08
3-D 5-level	23	61.3	24	61.6

of $\lambda_n \approx -2$ did not affect the convergence very much. However, when $\lambda_n^{'} > \lambda_n$, the convergence deteriorated significantly, and it even diverged in our 3-D test.

6.5.2. Comparison with Gauss-Seidel Iterations. To assess the effectiveness of the CJ method as smoother, we compare the convergence of HyGA(3,$\ell - 3$) with CJ versus Gauss-Seidel iterations as the smoother. In both cases, we use the conjugate gradient method as the coarse grid solver. Table IV shows the numbers of iterations as well as the run times of the methods. In both 2-D and 3-D, Gauss-Seidel works rather well as a smoother. The convergence of CJ is a bit worse than Gauss-Seidel in the 2-D tests, but it is better than Gauss-Seidel in 3-D when degree-4 polynomials are used. This result is remarkable, because the CJ method is much easier to parallelize and is more scalable than Gauss-Seidel, so it can be expected to out-perform Gauss-Seidel iterations on parallel computers. Similar behaviors of other semi-iterative methods have been reported in [30].

7. CONCLUSIONS

In this paper, we introduced a hybrid geometric+algebraic multigrid method, *HyGA*, with semi-iterative smoothers. Our approach is motivated by a new trend of large-scale parallel mesh generation through mesh refinements. Our HyGA multigrid solver utilizes a few levels of geometric multigrid on these hierarchical meshes for accuracy and efficiency, along with algebraic multigrid on the coarse-levels for simplicity and robustness. We presented a unified derivation of the prolongation and restriction operators on the GMG levels for weighted-residual methods with hierarchical basis functions, including finite elements and generalized finite differences with hierarchical unstructured meshes. Our numerical experiments demonstrated the advantages of our proposed hybrid technique compared to the classical GMG and AMG methods.

Another main contribution of this paper was the introduction of the Chebyshev-Jacobi method, or CJ, as a smoother for multigrid methods for both symmetric and asymmetric linear systems. As a standalone solver, the CJ drastically accelerates Jacobi iterations by an order of magnitude, and also significantly robustifies the Jacobi iteration. At the same time, it is as easy to implement as the Jacobi iterations, especially on parallel computers. As a standalone solver, the CJ requires relatively accurate estimates of the largest eigenvalue of the iteration matrix. As a smoother, we showed that CJ is significantly less sensitive to the estimations of eigenvalues, and it drastically simplifies, and reduces the cost of, the estimation procedure. This property also simplifies the application of CJ to asymmetric matrices, especially those nearly symmetric matrices arising from PDE discretizations.

DOI: 10.1002/nla

Our proposed approach has the potential to deliver an efficient, robust, and easy-to-implement multigrid solver for PDEs. We are currently developing parallel implementations of HyGA and will report the results in future publications.

References

1. Briggs WL, Henson VE, McCormick SF. *A Multigrid Tutorial*. Second edition edn., SIAM, 2000.
2. Trottenberg U, Oosterlee CW, Schuller A. *Multigrid*. Academic Press, 2000.
3. Brannick J, Brezina M, Falgout R, Manteuffel T, McCormick S, Ruge J, Sheehan B, Xu J, Zikatanov L. Extending the applicability of multigrid methods. *J. Phys.: Conf. Ser.* 2006; **46**:443–452.
4. Jiao X, Wang D. Reconstructing high-order surfaces for meshing. *Engineering with Computers* 2012; **28**:361–373.
5. Bramble J, Pasciak J. The analysis of smoothers for multigrid algorithms. *Math. Comput.* 1992; **58**:467–488.
6. Hackbusch W. *Multi-Grid Methods and Applications*. Springer, 1985.
7. Sampath RS, Biros G. A parallel geometric multigrid method for finite elements on octree meshes. *SIAM J. Sci. Comput.* 2010; **32**:1361–1392.
8. Adams MF, Demmel J. Parallel multigrid solver algorithms and implementations for 3D unstructured finite element problem. *ACM/IEEE Proceedings of SC99: High Performance Networking and Computing*, Portland, Oregon, 1999.
9. Mavriplis DJ. *Multigrid Techniques for Unstructured Meshes*. VKI Lecture Series VKI-LS 1995-02, Rhode-Saint-Genese, Belgium, 1995.
10. Mavriplis DJ. An assessment of linear versus nonlinear multigrid methods for unstructured mesh solvers. *Journal of Computational Physics* 2002; **175**(1):302 – 325, doi:10.1006/jcph.2001.6948.
11. Xu J. The auxiliary space method and optimal multigrid preconditioning techniques for unstructured grids. *Computing* 1996; **56**(3):215–235, doi:10.1007/BF02238513.
12. Mavriplis D, Jameson A. Multigrid solution of the euler equations on unstructured and adaptive meshes. *3rd Copper Mountain Conference on Multigrid Methods*, Copper Mountain, CO, 1987. Paper 23.
13. Hulsemann F, Kowarschik M, Mohr M, Rude U. Parallel geometric multigrid. *Numerical Solution of Partial Differential Equations on Parallel Computers, volume 51 of LNCSE, chapter 5*, 2005; 165–208.
14. Brandt A, McCormick S, Ruge J. Algebraic multigrid (amg) for sparse matrix equations. Cambridge University Press, 1984.
15. Vanek P. Acceleration of convergence of a two-level algorithm by smoothing transfer operators. 1992.
16. Brezina M, Cleary AJ, Falgout RD, Henson VE, Jones JE, Manteuffel TA, McCormick SF, Ruge JW. Algebraic multigrid based on element interpolation (amge). *SIAM J. Sci. Comput.* May 2000; **22**(5):1570–1592, doi: 10.1137/S1064827598344303.
17. Adams M. Prometheus- scalable unstructured finite element solver.
18. Falgout R, Cleary A, Jones J, Chow E, Henson V, Baldwin C, Brown P, Vassilevski P, Yang UM. Hypre home page 2001.
19. Gee M, Siefert C, Hu J, Tuminaro R, Sala M. Ml 5.0 smoothed aggregation user's guide. *Technical Report SAND2006-2649*, Sandia National Laboratories, Albuquerque, NM 2006.
20. Sundar H, Stadler G, Ghattas O, Biros G. Parallel geometric-algebraic multigrid on unstructured forests of octrees. *Proceedings of IEEE/ACM SC12*, 2012.
21. Saad Y. *Iterative methods for sparse linear systems*. 2nd edn., SIAM, 2003.
22. Saad Y. *Numerical Methods for Large Eigenvalue Problems*. SIAM, 2011.
23. Balay S, Buschelman K, Eijkhout V, Gropp WD, Kaushik D, Knepley MG, McInnes LC, Smith BF, Zhang H. PETSc users manual. *Technical Report ANL-95/11 - Revision 3.0.0*, Argonne National Laboratory 2008.
24. Benito JJ, Ureña F, Gavete L. *Leading-Edge Applied Mathematical Modeling Research*, chap. Chapter 7: The Generalized Finite Difference Method. Nova Science Publishers, Inc., 2008.
25. Shaidurov V. *Multigrid methods for finite elements*. Springer, 1995.
26. Jiao X, Wang D, Zha H. Simple and effective variational optimization of surface and volume triangulations. *Engineering with Computers* 2011; **27**:81–94.
27. Jiao X, Wang D. Reconstructing High-Order Surfaces for Meshing. *Proceedings of the 19th International Meshing Roundtable*, Shontz S (ed.), Springer Berlin Heidelberg, 2010; 143–160.
28. Golub GH, Varga RS. Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second-order Richardson iterative methods. *Numer. Math. Parts I and II* 1961; **3**:147–168.
29. Varga R. *Matrix iterative analysis*. Prentice-Hall: Englewood Cliffs, NJ, 1962.
30. Adams M, Brezina M, Hu J, Tuminaro R. Parallel multigrid smoothing: polynomial versus Gauss-Seidel. *J. Comput. Phys.* 2003; **188**(2):593–610, doi:10.1016/S0021-9991(03)00194-3.
31. Hageman LA, Young DM. *Applied Iterative Methods*. Dover Books on Mathematics, Dover Publications, 2004.
32. Clayton AJ. Further results on polynomials having least maximum modulus pver an ellipse in the complex plane. *Technical Report AEEW-M348*, United Kingdom Atomic Energy Authority, Winfrith, Dorchester 1963.
33. Shewchuk JR. Triangle: Engineering a 2d quality mesh generator and Delaunay triangulator. *Lecture Notes in Computer Science*, vol. 1148, 1996; 203–222.
34. Si H. TetGen, a quality tetrahedral mesh generator and three-dimensional Delaunay triangulator v1.4 2006.