

Mesh and Solver Co-adaptation in Finite Element Methods for Anisotropic Problems

Qiang Du,^{1,2} Zhaohui Huang,³ Desheng Wang^{4,5}

¹Department of Mathematics, Penn State University, University Park, PA 16802

²Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100080, China

³Center for Space Science Research, Chinese Academy of Sciences, Beijing 100080, China

⁴Department of Mathematics, Xiangtan University, Xiantan, Hunan 411105, China

⁵Civil and Computational Engineering Centre, School of Engineering, University of Wales Swansea, Singleton Park, Swansea SA2 8PP, UK

Received 16 July 2004; accepted 15 January 2005

Published online 4 March 2005 in Wiley InterScience (www.interscience.wiley.com).

DOI 10.1002/num.20072

Mesh generation and algebraic solver are two important aspects of the finite element methodology. In this article, we are concerned with the joint adaptation of the anisotropic triangular mesh and the iterative algebraic solver. Using generic numerical examples pertaining to the accurate and efficient finite element solution of some anisotropic problems, we hereby demonstrate that the processes of geometric mesh adaptation and the algebraic solver construction should be adapted simultaneously. We also propose some techniques applicable to the co-adaptation of both anisotropic meshes and linear solvers. © 2005 Wiley Periodicals, Inc. *Numer Methods Partial Differential Eq* 21: 859–874, 2005

Keywords: finite element; anisotropic meshing; iterative solver; joint co-adaptation

I. INTRODUCTION

The finite element solution of partial differential equations involves the mesh generation and optimization, the assembly of discrete algebraic systems using the finite element basis, and the solution of these systems by some algebraic solvers. It is well known that the performance of finite element computations depends critically on both the geometric meshes and the algebraic

Correspondence to: Qiang Du, Department of Mathematics, Penn State University, University Park, PA 16802 (e-mail: qdu@math.psu.edu)

Contract grant sponsor: Chinese State Major Basic Research Project; contract grant number: G1999032800

Contract grant sponsor: U.S. National Science Foundation; contract grant number: DMS-0196522.

© 2005 Wiley Periodicals, Inc.

solvers. In modern finite element methodology, in particular, within the adaptive finite element framework, these two aspects are becoming closely related since the performance of method is ultimately judged by the efficiency of the linear solver and the quality of its output.

Traditionally, the generation of finite element meshes is often exclusively focused on providing a good resolution of the solutions (thus, to assure the accuracy of the discretization) [1–8]. Many mesh improvement techniques (in the sense of providing a better control on the element shape and interpolation errors), including local reconnection schemes, node smoothing, and adaptive refinement or coarsening, have been developed [2, 3, 6, 9–11]. They usually are not related to the performance of the linear solvers, but are aimed at matching the shape and the size of the (triangular) mesh with the smoothness of the local solution. Such a strategy has remained as a dominating practice in many adaptive algorithms. On the other hand, finite element discretizations reduce the problem of solving partial differential equations to that of solving large systems of algebraic equations. Most of the algebraic solvers, in particular linear solvers, are not directly concerned with the mesh geometry and mesh quality, except the multigrid methods with semi-coarsening and line-smoothing. Though in practice, the attributes of the underlying mesh, for instance, anisotropic meshes containing triangles with large angles, may have a serious effect on the convergence of linear solvers (thus, on the efficiency of the solution process).

The purpose of this article is to advocate the following simple principle: the mesh generation and optimization should be based on not only the behavior, or the spatial variations of the solution, but also the (linear) solver(s) used for computing the numerical solutions; vice versa, information on the PDEs and the underlying mesh structure should both be taken into account when designing suitable linear solvers. Such a principle is what we refer to as the mesh and solver co-adaptation of the finite element method.

As an illustration of such a principle, we consider instances where the solutions of certain model partial differential equations display anisotropic behavior and thus demand the use of anisotropic grids in order to provide optimal numerical resolution. Our test problems are purposefully chosen so that we may investigate the use of several different types of triangulations, both structured and unstructured, as underlying meshes. Meanwhile, we take both the algebraic multigrid methods and the Conjugate Gradient type methods as representative linear solvers. One of the important observations drawn from our numerical experiments is that eliminating elements having *bad shapes* for the sake of improving the efficiency of the linear solver may, in practice, be an even more pressing issue than generating suitable meshes for the optimal resolution targeting the reduction of the approximation error.

Though the model problems considered here are limited to simple cases, the observations one can make are very generic and they provide insight for more practical problems. Guided by these observations, we discuss how to reconcile the need for a high resolution mesh using anisotropic triangular meshes and the need for efficient linear solvers for the resulting linear systems. The existing algebraic multigrid algorithms are adapted to the given anisotropic mesh with large angles by modifying the selection of coarse grids, and by introducing local relaxation; the generated anisotropic meshes are optimized by adding midpoints to the long edges to delete the large angles. The co-adaptation provides better overall accuracy and efficiency for the finite element solution on anisotropic finite element grids.

The rest of this article is organized as follows. We first present the model equations and their discretizations in section 2. In section 3, anisotropic mesh generation methods are briefly presented, and some element quality measures are discussed. We then discuss the effect of the large angles in the anisotropic mesh on the performance of several basic iterative solvers including an algebraic multigrid method. Then, we propose how to co-adapt the iterative solvers

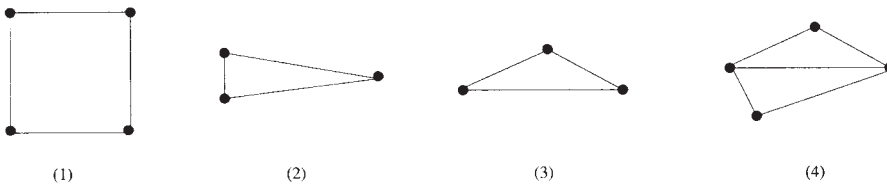


FIG. 1. Different computational domains Ω_i for the model equation.

and the anisotropic meshes together. Numerical results on the improvement of the numerical efficiency of the solver after the joint adaptation are given as demonstrations in section 4 and section 5, respectively. Finally, section 6 contains the conclusions and future directions.

II. A MODEL EQUATION AND ITS FINITE ELEMENT DISCRETIZATION

To analyze how the properties and qualities of the anisotropic mesh affect the overall accuracy and efficiency of the numerical solution of PDEs, including the performance of the linear solvers, we take a simple two-dimensional model problem as an illustration:

$$\begin{cases} -u_{xx} - u_{yy} = 102 - 0.02\pi^2 \cos(\pi(x + y)), & \text{in } \Omega, \\ u = x^2 + 50y^2 + 0.01 \cos(\pi(x + y)), & \text{on } \partial\Omega. \end{cases} \tag{2.1}$$

The above test problem is a linear Poisson type equation with an inhomogeneous Dirichlet boundary condition over a domain Ω to be specified later. The equation is chosen to yield an exact solution $u(x, y) = x^2 + 50y^2 + 0.01 \cos(\pi(x + y))$, which displays a strong anisotropic behavior, with a ratio between the two second-order derivatives u_{xx} and u_{yy} on the order of 50. Thus, it demands an anisotropic finite element grid with strong anisotropy, i.e., in our context, the element sizing in y should be much smaller than that in the x direction (the mesh elements are stretched in the x direction). Such an anisotropic meshing leads to significant savings in total number of grid points and the solution costs. Though more complex and realistic problems can be considered, we limit our attention to the model problem described here for the sake of easy benchmarking.

The model equation defined above is discretized with the standard linear, triangular finite elements [5]. We consider the numerical solution on a number of possible computational domains that include: the unit square (Ω_1) shown in Fig. 1(1), an acute triangle (Ω_2) shown in Fig. 1(2), an obtuse triangle (Ω_3) shown in Fig. 1(3), and a composition of long thin triangles (Ω_4) shown in Fig. 1(4).

For the domain Ω_1 , we consider the following three types of anisotropic meshes: 1) a structured triangulation (ST) [as shown in Fig. 2(1)]; 2) a triangulation with crossed triangles (CT) [as shown in Fig. 2(2)]; and 3) an unstructured Delaunay triangulation (DT) [as shown in Fig. 2(3)].

For the domains Ω_2 and Ω_3 , we consider two meshes: a structured triangulation having triangular elements whose edges all being parallel to the respective boundary edges of the domain [see for example Fig. 3(1, 2)], and an unstructured Delaunay triangulation obtained via standard methods [12–15].

For Ω_4 , we consider either a composition of two compatible meshes for the two individual triangles [see for example Fig. 3(3)], or an unstructured Delaunay triangulation of the union Ω_4 .

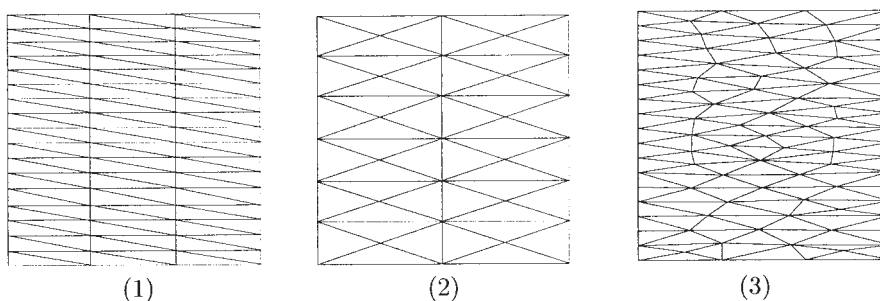


FIG. 2. Different anisotropic meshes for the unit domain.

The motivation for studying the special geometry and the meshes defined above stems mostly from the following considerations: first, with the different meshes defined on the square domain, we are able to examine the effect of the mesh topology on the iterative solvers. Second, in light of the angle conditions imposed on the standard finite element approximations, domains and triangulations with small acute angles and large obtuse angles may give insights to our investigation on the effect of the triangles shapes on the iterative solvers. Third, the unstructured Delaunay triangulation is among the most popular grid generation strategies, and the computation on such a mesh may serve as a bench-marking test [14–17]. Additionally, compositions of different geometry and different grids allow us to simulate slightly more complex situations in which either the grid generation is done with a combination of several different strategies or the solution behavior changes dramatically from part of the domain to another part of domain. In the following section, various methods for the generation of the anisotropic finite element meshes such as the ones described above are briefly presented, along with more extensive discussions on the element quality measures.

For an in-depth discussion on the relation between the mesh and the interpolation and approximation errors as well as the conditioning of the local stiffness matrices, we refer to a recent report [8]. We also refer to Babuska and Aziz [1] for a pioneering work on relating the analysis of the discretization error in the finite element solution with that of the element angles.

III. ANISOTROPIC MESH GENERATION AND ELEMENT QUALITY

The construction of a directionally stretched mesh for a simple domain is straightforward. For very complex geometries, such as those surrounding commercial airplanes [18], the unstructured anisotropic mesh often enjoys the advantage of being better conformed to the boundary, but its generation is nontrivial and much more demanding.

Over the last two decades, various approaches for unstructured anisotropic mesh generation have become available. The two-dimensional problem has been well studied [12, 13, 19–25],

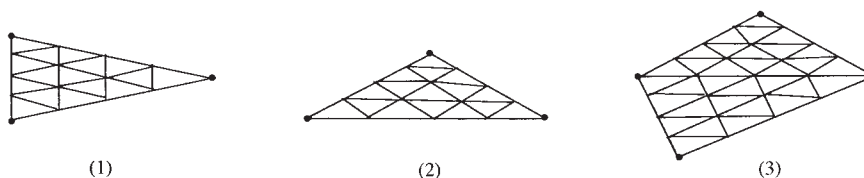


FIG. 3. Structured anisotropic meshes for triangular and quadrilateral domains.

while much preliminary works in the three dimensional space have also become available [26, 27]. One of the most systematic approaches is the anisotropic Delaunay mesh generation proposed by P. L. George by the name of the unit meshing method, which uses a modified Delaunay criterion and the insertion kernel [13, 14]. The classical Advancing-front-method is also extended to include directional stretching, which mainly depends on the directional points generation [13, 15]. Other approaches include the advancing-layer or normal methods, which can be seen as variants of the advancing-front method [22, 23, 25], the local refinement methods based on the longest-edge splitting and the normalized edge-length computation [19, 23, 28, 29], the directional enrichment method [25], and ellipses biting or packing, which is the extension of the sphere biting or packing method [24, 27]. Most of the above methods share a common feature: the meshing process is controlled under a derived metric (a positive definite matrix or tensor which usually comes from the Hessian matrix of the computed solution or some other posterior error estimators) [6, 8, 10, 14, 15, 24, 27]. The anisotropic mesh is generated to provide best agreement with the anisotropy specified by the derived metric tensor. In turn, this means that the mesh sizing and the directional stretching, or say, the shape of elements, is closely related to the spatial variation of the numerical solution.

An element quality measure is used to measure the agreement of the anisotropic mesh with the metric tensor. Most of the existing measures are concerned with either the geometric shapes or the approximation properties of the nodal finite element functions (typically piecewise linear elements). There are many approaches and criteria for developing such quality measures [3, 4, 7, 8, 14, 15]. One of our goals here is to advocate the evaluation of the mesh quality not only based on their approximation properties but also on how they affect the performance of the linear solvers. While the former has been extensively studied and various analytic formulae have been presented, it is still yet difficult to come up with a simple quantity that would characterize the latter. Nevertheless, as we see later, factors such as anisotropy, small and large angles, and local connectivity matrices all contribute to the effectiveness of the linear solver.

Let us first recall the following popular quality measure. For more discussions on the topic, we refer to [8, 14, 15] and the references cited therein. Let τ be an element of a given triangulation with three vertices $\{p_i\}_{i=1}^3$, let $D_i = |Det(\overrightarrow{p_i p_j}, \overrightarrow{p_i p_k})|$ for $\{i, j, k\} = \{1, 2, 3\}$, the quality of τ is defined as: $Q(\tau) = \min_{1 \leq i \leq 3} Q^i(\tau)$, where

$$Q^i(\tau) = c \sqrt{Det M(p_i)} D_i / S \quad \text{with} \quad S = \sum_{\{i,j,k\}=\{1,2,3\}} \overrightarrow{p_i p_j} M(p_i) \overrightarrow{p_i p_k}.$$

Here, $M(p_i)$ is a predefined Riemannian tensor at the point p_i . The value of $Q(\tau)$ ranges from 0 to 1, with the latter corresponding to an “equilateral” triangle (a stretched triangle K whose mapped triangle is an equilateral triangle when transformed into the isotropic space [13–15, 30]). For the isotropic case $M = cI$, a constant multiple of the identity, the above quality measure reduces to the classical measure for the triangle quality in Euclidean space [13–15]. In the adaptive finite element solution, the matrix M is usually derived from the Hessian matrix of the computed solution. Obviously, the above quality measure is a typical one related to approximation accuracy, and there is no obvious relationship to the solution efficiency of the solver.

In Fig. 4, three types of anisotropic triangles are shown: one with small angle and no larger angle [Fig. 4(1)]; one with a large angle and two small angles [Fig. 4(2)]; and the third one [Fig. 4(3)] with either a right angle or generally an angle close to $\pi/2$. Under the anisotropic quality measure, the mesh qualities of these three kinds of triangles are, respectively, 1.0, 1.0, and 0.75. Hence, even with a large angle, the second type of elements still enjoys an “equilateral” quality in the Riemannian metric. The average qualities for the three meshes given on the unit domain

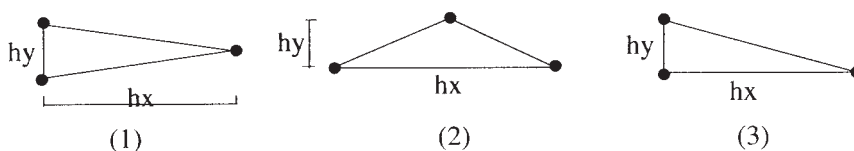


FIG. 4. Three types of anisotropic triangles.

Ω_1 are 0.75, 0.6, and 0.9, respectively. For the other triangular domains and the composite domain (Ω_{2-4}), the structured meshes all have average quality 1.0, and the Delaunay-type meshes 0.9. In the numerical examples presented later, by using these sample grids, it is shown that the classical quality measure may not be adequate when assessing the effect of meshes on the overall quality of the numerical solution.

In [1], Babuska and Aziz showed that the accuracy of the finite element solutions degrades seriously on triangular meshes with larger angles approaching π , while elongated triangles with only small angles do not suffer such a loss of accuracy. In [8], Shewchuk points out that the same claim does not hold for anisotropic problems. The conclusion is that, triangles with either very small or very large angles can both perform well in anisotropic circumstances, if they are oriented properly. This has been further corroborated by other applications [8]. Nevertheless, in much of the existing anisotropic mesh generation approaches, little attention is paid to the issue of preventing large angles. Accordingly, even for simple domains, a portion of triangles with large angles may be generated in the final mesh [8, 12, 13, 15]. In particular, the generalized Delaunay-type anisotropic mesh often has a high quality in general and offers good approximations in applications, even with larger angles [8, 13, 14].

The relation between the mesh quality and the condition numbers of the stiffness matrices generated from the finite element method is more complicated and dependent on the anisotropy of the underlying PDEs [1–4, 6–8]. Thin elements with only small angles may not affect accuracy but it may give rise to poor conditioning even for isotropic problems. Of course, for some iterative solvers, such as multigrid methods, a poorly conditioned stiffness matrix is not necessarily an impediment [8]. We will not go beyond this observation here as much more detailed discussions on the relationship between the element shape and the conditioning of the stiffness matrix have been given in [8].

What concerns us the most are those elements with large angles, or say, the cap-like angles. On the basis of our numerical examples to be presented in the next section, we find the existence of these large-angle triangles hampers the efficiency of Multigrid solvers. We propose to go in two directions. One is to modify the solver (here we take the algebraic Multigrid solver as an example) according to the given anisotropic mesh, despite the existence of some large angles; whereas the other approach is to adapt the anisotropic mesh, in other words, to do *a posteriori* optimization on the mesh, which helps to reduce the number of large-angle elements. This coadaptation enhances the effectiveness of the linear solvers significantly, which will be discussed in details in section 4 and section 5.

IV. ITERATIVE SOLVERS AND THEIR MODIFICATIONS FOR ANISOTROPIC GRIDS

In this section, we hope to exploit the underlying mesh information to get efficient solvers for the anisotropic grids. We first describe our notation convention to be used to present the results

of numerical experiments. In all the tables, concerning mesh abbreviation, the letter “T” represents for triangular meshes, “S” for structured, “C” for crossed, “D” for Delaunay, “A” for anisotropic; “LA” and “SA” refer to the meshes having many large angles and small angles, respectively; and digits “0–9” means the different meshes of the same type but with varying degrees of anisotropy. Regarding the stiffness matrix, N_{eqs} denotes the dimension of the algebraic system; $Cond_2$ the spectral condition number; N_{CG} the number of iterations for the CG convergence, defined by $\|r^k\|/\|r^0\| \leq 10^{-6}$, where r^k is the residual vector at the k th iteration and the same iterative tolerance on the residual is used for AMG; ρ the asymptotic convergence factor, that is, the limit of $\|e^k\|/\|e^{k-1}\|$.

There exist many works on the design of efficient multigrid methods for anisotropic problems using ideas such as semi-coarsening and line-smoothing [25, 31–35]; still, there are very few systematic studies on the effect of the triangle shapes or mesh qualities on these iterative schemes, in particular, algebraic multigrid methods. Here, we mainly discuss the algebraic multigrid (AMG) and the conjugate gradient (CG) solvers. For a description of the CG procedures, we refer to [36]. For the AMG, one may read [37–41] for details. We here briefly recall the outline. For simplicity of illustration, we borrow some of the geometric multigrid terminology such as coarse grids, which need not be explicitly defined in the implementation, but are convenient for the algorithm description. The general AMG procedure has two phases: 1) the set-up phase, in which the five components for the multigrid cycle are defined in terms of the algebraic matrices [42]: the so-called *coarse grid* Ω^{m+1} , the interpolation operator I_{m+1}^m , the restriction operator $I_{m+1}^{m+1} = (I_{m+1}^m)^T$, the *coarse-grid* operator $A^{m+1} = I_{m+1}^{m+1} A^m I_{m+1}^m$, and the smoothing operator G^m ; 2) the solver phase, i.e., the full multigrid cycle. The AMG μ -cycle scheme, denoted by $u^m \leftarrow AMG_{\mu}^m(u^m, f^m)$, is then recursively defined by the following four steps:

$$\text{AMG} \quad u^m \leftarrow AMG_{\mu}^m(u^m, f^m)$$

1. Relax μ_1 times on $A^m u^m = f^m$ starting with an initial guess u^m .
2. If $m = M$, go to Step 4; else, set $r^m = f^m - A^m u^m$, $f^{m+1} = I_{m+1}^{m+1} r^m$, $u^{m+1} \leftarrow 0$, and perform μ times $u^{m+1} \leftarrow AMG_{\mu}^{m+1}(u^{m+1}, f^{m+1})$.
3. Set $u^m \leftarrow u^m + I_{m+1}^m u^{m+1}$.
4. Relax μ_2 times on $A^m u^m = f^m$ starting with u^m .

The AMG V-cycle, defined by AMG_1 , is combined with a nested iteration technique, resulting in the corresponding AMGV algorithm. We emphasize that the terms like *coarse grids* are for convenience of reference only and they pertain only to algebraic information. In the following, some numerical experiments and results will be presented to examine the effect of mesh anisotropy on the basic iterative solvers.

A. Experiment 1: Effect of Anisotropy

We solve the model problem on domains Ω_1 and Ω_3 with different anisotropic grids.

From the results presented in Table I, it can be seen that, for the AT or the ACT type meshes, when the mesh size ratio $h_x : h_y$ increases, i.e., the anisotropy of the mesh aggravates, the condition number of stiffness matrix becomes larger, the convergence rates of AMGV change from 0.961 to 0.990 and from 0.238 to 0.342, respectively, for Ω_1 and Ω_3 , and the iteration count of CG increases correspondingly [43]. At the same time, the presence of elements with many large angles also holds back the fast convergence of the AMG substantially. This presents a

TABLE I. Numerical results for Experiment 1.

Domain	Ω_1	Ω_1	Ω_1	Ω_3	Ω_3	Ω_3
Mesh	ACT1	ACT2	ACT3	AT(LA)1	AT(LA)2	AT(LA)3
Size ratio	1:22	1:50	1:100	10:1	20:1	50:1
N_{eqs}	9846	9491	9094	11026	11026	11026
$Cond_2$	47551	98816	195068	6894	7616	8029
N_{CG}	652	885	1117	456	478	491
ρ_{AMGV}	0.961	0.987	0.990	0.238	0.311	0.342

dilemma: on one hand we should avoid having too many elements with large angles to prevent the slow down of the iterative scheme, but on the other hand, if one hopes to provide a good representation of the solution, anisotropic triangulation may inevitably produce elements with large angles. The latter has been an intrinsic difficulty of anisotropic grid generation as well as the anisotropic Delaunay triangulation, which in part, motivates one aspect of the principles advocated in this article: one should adapt the linear solvers to anisotropic meshes and make appropriate adjustment based on the mesh structure and element shapes. To this end, the appropriate selection of coarse grids with respect to given anisotropic triangular mesh and local relaxation techniques for the *AMG* will be presented next.

B. Selection of Coarse Grids

For the finite element approximation of the second-order elliptic equation introduced in Section 2, the stiffness matrix A can be expressed as the sum of local element-wise stiffness matrices:

$$A = \sum_{\alpha \in \Gamma} A_{\alpha},$$

where Γ is the set of mesh elements used to discretize the problem and each A_{α} is symmetric positive semi-definite.

Let ε_i denote two vectors corresponding to the nodal basis function at the grid point x_i in the finite element space; then we have the nodal set of an element α :

$$P_{\alpha} = \{x_j : \varepsilon_j^T A_{\alpha} \varepsilon_j \neq 0\},$$

the neighborhood element of the grid point x_i :

$$E_i = \{\alpha \in \Gamma : \varepsilon_i^T A_{\alpha} \varepsilon_i \neq 0\}$$

and the nodal neighborhood of i :

$$N_i = \bigcup_{\alpha \in E_i} P_{\alpha}.$$

The node-wide local stiffness matrices are then given by

$$A_i = \sum_{\alpha \in E_i} A_{\alpha}.$$

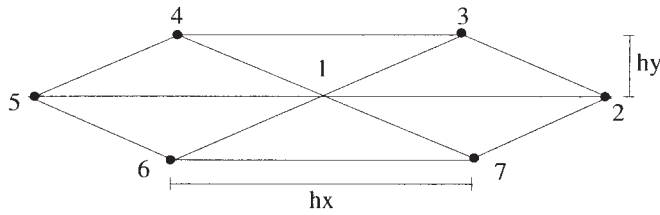


FIG. 5. Finite element node.

In reference to Fig. 5, for piecewise linear element, we obtain the local equation corresponding to the grid point “1” as

$$\sum_{j=1}^7 c_{1j}u_j = f_1,$$

where $c_{11} = \frac{3}{2}(h_x/h_y) + 2(h_y/h_x)$, $c_{12} = c_{15} = \frac{1}{4}(h_x/h_y) - (h_y/h_x)$, and $c_{13} = c_{14} = c_{16} = c_{17} = -\frac{1}{2}(h_x/h_y)$.

We now examine the specific cases where the model problem is discretized on a regular triangle, an obtuse triangle, an acute triangle, and their composite domains.

1. Triangulation made of regular triangles.

This type of structured triangulation has been discussed most frequently in the literature. With $(h_x/h_y) = (2/\sqrt{3})$, the local equation has coefficients satisfying

$$c_1 : c_2 : c_3 : c_4 : c_5 : c_6 : c_7 \approx 6 : -1 : -1 : -1 : -1 : -1 : -1.$$

From [42], we know that AMG enjoys fast convergence when coarser grid lines are selected to be parallel to any one of the three sides of the original triangle.

2. Triangulation made of acute triangles.

As the aspect ratio $(h_x/h_y) \ll 1$, we have

$$c_1 : c_2 : c_3 : c_4 : c_5 : c_6 : c_7 \approx 2 : -1 : 0 : 0 : -1 : 0 : 0.$$

One may draw an analogy between the choice of the above anisotropic grid for our model problem with an isotropic grid for the anisotropic problem $u_{xx} + \epsilon u_{yy} = f$, which has a 5-point finite difference stencil of the form

$$\frac{1}{h^2} \begin{bmatrix} & -\epsilon & \\ -1 & 2(1 + \epsilon) & -1 \\ & -\epsilon & \end{bmatrix}_h,$$

where $\epsilon \ll 1$. A detailed local Fourier analysis has been given in [40, 42, 44] about the anisotropic problem. The geometric multigrid performs a line relaxation in order to ensure

TABLE II. Numerical results for *AMGC*.

Domain	Ω_1	Ω_1	Ω_1	Ω_3	Ω_3	Ω_3
Mesh	ACT1	ACT2	ACT3	AT(LA)1	AT(LA)2	AT(LA)3
Size ratio	1:22	1:50	1:100	10:1	20:1	50:1
N_{eqs}	9846	9491	9094	11026	11026	11026
ρ_{AMGC}	0.0208	0.0191	0.0189	0.128	0.162	0.168

sufficient smoothing when a standard coarsening is used. Moreover, an *AMG* method with a fixed Gauss-Seidel relaxation can also be applied since the semi-coarsening in the x -direction is effective.

In another word, the coarser grid lines can be chosen to be parallel to any hypotenuse. As the computational results below indicate, small angles in the mesh have little effect on the *AMG* convergence.

3. Triangulation made of obtuse triangles, as illustrated in Fig. 3(2).

For the case $(h_x/h_y) \gg 1$, we have

$$c_1 : c_2 : c_3 : c_4 : c_5 : c_6 : c_7 \approx 6 : 1 : -2 : -2 : 1 : -2 : -2.$$

Because of the positive off-diagonal entries in the local stiff matrices, this has always been a challenging problem for *AMG*. Some remedies are given in [40, 42, 44], which work well if the positive entries are relatively small. Otherwise, large positive off-diagonal entries must be taken into account in the coarsening and interpolation processes.

Here we perform the coarsening along grid lines parallel to the base side, which means the elimination from the point “2” and “5” (see Fig. 5) in the interpolation. Despite of this improved strategy, the numerical results given later demonstrate that too many large angles still severely affect the convergence and the accuracy of algebraic iterative solvers.

4. Triangulation composed with two type of triangles.

AMG coarsening in each part of the domain is automatically adapted to this problem. In addition, we apply a geometry-based interpolation formula (and thus the coarsening process), the modified algorithm works effectively in this case. That is to say, by making flexible choices on the coarse grids and the interpolation formula, we get a much improved solver, which we refer to as *AMGC*. For demonstrations, we repeat the experiment 1 with *AMGC*. The following computational results are obtained.

As shown in Table II, for some structured anisotropic triangulation on different domains, significant improvement is made with *AMGC*. For unstructured anisotropic Delaunay triangulation with large angles, our numerical experiments show that *AMGC* is less effective, and alternatively, we propose the local relaxation techniques (LRTs).

C. Local Relaxation Techniques

The above results indicate that the existence of large angles in the underlying anisotropic mesh is a key challenge to the fast convergence of linear solvers. Though with a Riemannian metric, the quality measure of these anisotropic meshes may be very good, it does not reflect their effectiveness in obtaining good quality solutions from the linear solvers.

TABLE III. Numerical results for Experiment 2.

Domain	Ω_1	Ω_3	Ω_4
Mesh	ADT	ADT	ADT
N_{eqs}	11164	15860	9801
$Cond_2$	69530	34103	7212
N_{CG}	978	889	467
ρ_{AMGV}	0.765	0.822	0.215
LRT_3	0.441	0.498	0.0912
LRT_9	0.370	0.343	0.311

Based on our experience, in a given anisotropic Delaunay-type triangulation, the number of large angles usually remain relatively small, and these large angles are often not very close to each other. It is, however, difficult to eliminate these larger angles by local mesh operations as we see from the later section. For the moment, we define a patch around the element with a large angle and assume that these patches are well separated from each other. When some extra smoothing operations are performed inside the local patch, the effect of the large angles on the iterative solver is then reduced. Certainly this requires extra work load and CPU time, but our point of departure is to sacrifice local efficiency to gain global efficiency.

The efficient interplay between the smoothing and the coarse-grid correction is a key to any multilevel methods. Geometric multigrid (*GMG*) approaches set simple requirements on the smoother used, and perform fixed coarsening on pre-defined grid hierarchies; algebraic multigrid methods use very simple point-wise smoothers without sacrificing convergence, and put more effort into the coarse-grid correction process. Generally speaking, those error components which can not be removed by smoothing, need to be efficiently reduced by coarse-grid correction. As a result, *AMG* usually performs one pre- and post-smoothing step, but develops more sophisticated coarsening techniques and operator-dependent interpolation. This is what we have done in the above subsection, from which it can be seen that anisotropic grids make it difficult to select the appropriate coarse grids and produce the accurate interpolation operator. Thus it is not very hopeful to rely on the coarse-grid correction step to remove the effect of large angles. We then switch to the smoothing process. Care must be taken to differ this kind of local smoothing from global smoothing used usually in *GMG*. Furthermore, in practical implementation, we sweep and smooth not only inside the patches, but also those points who are strongly connected to the elements with large angles. In general, the smoothing domains are slightly larger than the patches themselves though they remain localized. The detailed algorithm is as follows:

Algorithm LRT_s :

1. Sweep each element i that has a larger angle, and determine its neighborhood N_i by strong connections [42], $N = \cup_{i=1}^I N_i$.
2. Do s times relaxation of $Au = f$ on N .
3. Perform *AMG* coarse-grid correction process.
4. Do s times relaxation of $Au = f$ on N .

D. Experiment 2: Model Problem with Anisotropic Delaunay Meshes

We now consider the numerical solution of the model problem on Ω_1 , Ω_3 , and Ω_4 .

Clearly, *AMG* with LRT_s enjoys faster convergence as the effects of large angles are greatly reduced through the extra smoothing (Table III).

V. MODIFICATIONS TO ANISOTROPIC MESHES

While the algebraic iterative solvers must adapt to the underlying problems and the mesh structures, we also advocate the coadaptation of the meshes to the iterative solvers, in particular, in reducing the number of large angles. Here we concentrate on the unstructured anisotropic meshes, especially the Delaunay-type. There are two approaches we can adopt: one is to modify the mesh generation method; the other is to perform post-processing to the resulting mesh upon its generation; more details will be given here for the latter approach, while only preliminary exploration will be made for the former.

A. Modified Approach for Mesh Generation

There are some general mesh generation strategies for avoiding the existence of many large angles. For example, the advancing-normal or layer methods which generate points in the normal directions of the front points [15, 20, 22, 23, 25] often lead to resulting elements having one angle closer to $\pi/2$, shown in Fig. 4. This kind of triangles offer good performance both in interpolation accuracy and in solution efficiency, even though they do not have ideal mesh quality values (by the measurement presented in Section 3).

Another possible way to reduce the percentage of large angles is to make some modifications to the criterion of the Delaunay insertion with respect to large angles. However, for complicated geometric domains and metrics, the modifications are likely to be quite complex and their implementations and improvements need to be studied further.

B. Local Operations

For most Delaunay-type anisotropic meshes, the portion of the triangles with large angles remains relatively small. Moreover, most of these large angles are not adjacent to each other, and they appear mostly in regions where there are large metric variations. Hence, in the mesh post-processing step, we may assume that these conditions are met. Then, to eliminate these large angles, we apply some local operations such as the insertion of additional vertices [19, 28, 29]. That is, we obey the principle that the large angles are to be reduced even with the increase of redundancy (at the loss of local efficiency) with respect to the anisotropy. Such a sacrifice does not affect the interpolation or approximation accuracy at the expense of added degree of freedoms. In this sense, the local operations we consider here are different from local edges swapping or smoothing [14, 15].

Based on a detailed analysis, we find that there are three typical kinds of local configurations where large angles exist, see Fig. 6. The first [shown in Fig. 6(1)] contains two elements both with large angles sharing a common edge. In this case, we add one interior point such that by trading the initial two triangles with four triangles, no large angles remain in the new triangles. The second kind shown in Fig. 6(2) is one boundary triangle with a large angle facing the boundary edge. This is often generated in the Delaunay-type methods with the points generated via the advancing-front-methods [14, 15]. For this type of triangle, splitting the boundary edge solves the problem [see Fig. 6(2)]. The third kind, shown in Fig. 6(3), is a configuration where between two elements having large angles, there are several elements with no large angles. If the two elements with large angles are sufficiently apart from each other, we give up the local operations; otherwise, we add points to the elements in between and apply local reconnections so that the final configuration contains no elements with large angles [see Fig. 6(3)]. Naturally, there also exist other kinds of local configurations, we leave their treatment for future study.

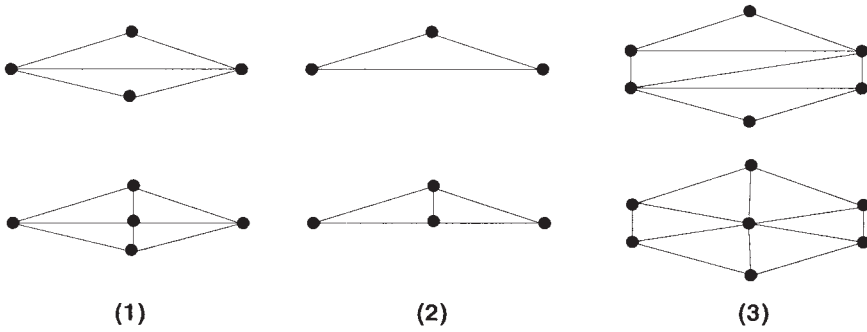


FIG. 6. Different local operations for large angles.

C. Experiment 3: Effect of the Number of Poor-Quality Elements on the Convergence of the Iterative Solvers

To demonstrate the effectiveness of the above local operations, we begin with a triangular domain Ω_2 having structured triangulations. Obviously, there are no large angles, and our numerical results show a good convergence rate of the *AMG*. Here, we artificially swap some short edges consecutively and perform a sequence of computations with different number of large angles. In Table IV, it is shown that the convergence rate gets worse with the increasing number of the large angles. The mesh *ATS1* refers to the original mesh and *ATS2*, *ATS3*, and *ATS4* represent swapping 5, 10, and 20 short edges, respectively. We also use ϵ_{L^2} to denote the L^2 norm of the finite element approximation on the given mesh. It is computed based on the difference between the exact solution and the final converged solution from the iterative solver.

D. Experiment 4: Effect of the Addition of the Middle Points to Long Sides

For the same problem as in the previous subsection, we now perform the local operations discussed as above to reduce the number of large angles, also in sequent manner. In fact, the specific type of local operation is the addition of middle points to the long sides shared by two elements with large angles. We then perform a sequence of *AMG* computations to see the effect on its convergence with a decreasing number of large angles. The computational results are shown in Table V, and it leads to better results with less large angles and there is no loss of solution accuracy in our computation. In Table V, *AT(SA)5*, 6, 7, and 8 mean adding 8, 12, 18, and 20 nodes in *ATS4*. The results of Table V demonstrate that our local operation strategy is effective since ρ_{AMGV} is significantly smaller while ϵ_{L^2} remains to be the same order as before.

Application to practical problems with more complicated geometries and general Delaunay anisotropic triangular meshes will be further studied in the future.

TABLE IV. Numerical results for Experiment 3.

Mesh	ATS1	ATS2	ATS3	ATS4
N_{eqs}	11026	11026	11026	11026
$Cond_2$	6364	9543	9542	9547
N_{CG}	660	695	704	713
ρ_{AMGV}	0.0527	0.183	0.231	0.251
ϵ_{L^2}	0.090469	0.090423	0.090364	0.090271

TABLE V. Numerical results for Experiment 4.

Mesh	AT(SA)5	AT(SA)6	AT(SA)7	AT(SA)8
N_{eqs}	11034	11038	11044	11046
$Cond_2$	10827	10835	10835	10835
N_{CG}	750	753	760	759
ρ_{AMGV}	0.237	0.233	0.123	0.0815
ε_{L_2}	0.0909	0.0908	0.0905	0.0905

VI. CONCLUDING REMARKS

In this work, we have presented some numerical studies on the effect of anisotropic finite element meshes on the convergence of some commonly used iterative solvers such as the *CG* solver and the *AMG* iteration. We focus on the relation between anisotropic mesh generation and the iterative solvers. A central issue throughout our discussion is to answer the question whether a mesh that is perhaps “good” under some measures related to the element shapes or to the discretization of solution is at the same time a “good” mesh that allows an efficient solution of the resulting linear system of equations. Similar issues have been noted also in other works. For instance, in [8], when discussing the demand for anisotropy with that for the matrix conditioning, it was pointed out that while refinement may compensate for the effects of badly shaped elements on the interpolation and discretization errors, such refinement does not compensate the effects on matrix conditioning.

Though new modifications to the *AMG* methods has been presented in this article to improve their efficiency for anisotropic problems, we also have found that improving the performance of the iterative solvers is much harder if there exists bad elements in the mesh as seen in the earlier examples. One remedy we propose here is to sacrifice the efficiency in the resolution to gain efficiency in the linear solver. Such a fact is also seen from the numerical examples presented here. For instance, in a contrived example, it has been shown that a few triangles with large angles slow down the iterative solvers significantly. Based on our experiments, in such a case, by introducing grid points along the side opposed to the large angles, and re-triangulate, fast convergence rate can be recovered for the iterative solver. Though such a placement does not respect the anisotropy of the solution (and thus lack the optimal efficiency for resolving the anisotropic solution), it nevertheless may eliminate all the badly shaped elements in the new triangulation and thus make the iterative solver work much faster. In this way, the overall solution efficiency are greatly enhanced. Consequently, a truly good mesh to be advocated is a mesh that should be both resolution dependent and also solver dependent.

Our ultimate goal is to build up a framework for adaptive computation, which accounts for both the accuracy and the efficiency for solving the resulting equations. Thus more research efforts are expected from two fronts: one on how to improve the grid adaptation to best suit iterative solvers, and the other on how to tune iterative solvers to work for less structured, highly nonuniform and highly anisotropic grids.

The authors thank Prof. Jinchao Xu for interesting discussions and the referees for helpful comments on the article.

References

1. I. Babuaska and A. K. Aziz, On the angle condition in the finite method, *SIAM J Numer Anal* 13 (1976), 214–226.

2. R. Bank, Mesh smoothing using a posteriori estimates, *SIAM J Numer Anal* 34 (1997), 979–997.
3. M. Berzins, A solution-based triangular and tetrahedral mesh quality indicator, *SIAM J Sci Comput* 19 (1998), 2051–2069.
4. M. Berzins, Mesh quality: a function of geometry, error estimates or both? *Proc 7th International Meshing Roundtable 15* (1999), 236–247.
5. P. Ciarlet, *Finite element methods for elliptic equations*, North-Holland, 1975.
6. E. D’Azevedo and B. Simpson, On optimal triangular meshes for minimizing the gradient error, *Numerische Mathematik* 59 (1991), 321–348.
7. S. Rippa, Long and thin triangles can be good for linear interpolation, *SIAM J Numer Anal* 29 (1992), 257–270.
8. J. Shewchuk, What is a good linear finite element? Interpolation, conditioning, anisotropy and quality measures, CS report, UC Berkeley.
9. N. Amenta, M. Bern, and D. Eppstein, Optimal point placement for mesh smoothing, *J Algorithms* 30 (1999), 302–322.
10. G. Buscaglia and E. Dari, Anisotropic mesh optimization and its application in adaptivity, *Int J Num Methods Engrg* 40 (1997), 4119–4136.
11. Q. Du and D. Wang, Tetrahedral mesh generation and optimization based on CVTs, *Int J Numer Methods Engrg* 56 (2003), 1355–1373.
12. H. Borouchaki and P. Frey, Adaptive triangular-quadrilateral mesh generation, *Int J Numer Methods Eng* 41 (1998), 915–934.
13. H. Borouchaki, P. L. George, F. Hecht, P. Laug, and E. Saltel, Delaunay mesh generation governed by metric specifications Part I, *Algorithms* 25 (1997), 61–83.
14. P. L. George and H. Borouchak, *Delaunay triangulation and meshing, Application to finite elements methods*, Hermès, Paris, 1998.
15. J. Thompson, B. K. Soni, and N. P. Weatherill, *Handbook of grid generation*, CRC Press LLC, 1999.
16. A. Bowyer, Computing Dirichlet tessellations, *Computer J* 24(2) (1981), 162–166.
17. D. Watson, Computing the n-dimensional Delaunay tessellation with applications to Voronoi polytopes, *Comput J* 24 (1981), 167–172.
18. Q. Du and D. Wang, Constrained boundary recovery for three dimensional Delaunay triangulation, *Int J Numer Meth Eng* 61 (2004), 1471–1500.
19. F. Bossen and P. Heckbert, A pliant method for anisotropic mesh generation, *Proc 5th International Meshing Roundtable*, 1996, pp 63–74.
20. M. J. Castro-Diaz, F. Hecht, and B. Mohammadi, New programs in anisotropic grid adaptation for inviscid and viscous flows simulations, *Proc 4th International Meshing Roundtable*, 1993, pp 73–85.
21. Q. Du and D. Wang, Anisotropi centroidal Voronoi tessellations and their applications, *SIAM J Sci Comp*, 2004, to appear.
22. R. Garimella and M. S. Shephard, Boundary layer meshing for viscous flows in complex domain, *Proc 7th International Meshing Roundtable*, 1998, pp 107–118.
23. J. Krause, N. Strecker, and W. Fichtner, Boundary-sensitive mesh generation using an offsetting technique, *Int J Num Methods Engrg* 49 (2000), 51–59.
24. J. Li, S. Teng, and A. Ungor, Biting ellipses to generate anisotropic mesh, *Proc 8th International Meshing Roundtable*, Lake Tahoe, CA, 1999, pp 97–109.
25. D. Mavriplis, Directional coarsening and smoothing for anisotropic Navier-Stokes problems, *Elect Trans Numer Anal* 6 (1997), 182–197.

26. P. L. George, Gamanic3d, Adaptive anisotropic tetrahedral mesh generator, Technical Report, INRIA, 2002.
27. S. Yamakawa and K. Shimada, Anisotropic tetrahedral meshing via bubble packing and advancing front, *Int J Num Method Engrg* 57 (2003), 1923–1942.
28. N. Hitschfeld, L. Villablanca, J. Krause, and M. Rivara, Improving the quality of meshes for the simulation of semiconductor devices using Lepp-based algorithms, *Int J Num Methods Eng* 58 (2003), 333–347.
29. M. C. Rivara and P. Inostroza, Using longest-side bisection techniques for the automatic refinement of Delaunay triangulations, *Int J Numer Method Engrg* 40 (1997), 581–597.
30. J. Shewchuk, Delaunay Refinement Mesh Generation, Ph.D. Thesis. Computer Science Dept Carnegie Mellon University, 1997.
31. M. Adams, Evaluations of the three unstructured multigrid methods in 3D finite element problems in solid mechanics, *Int J Numer Method Eng* 55 (2002), 519–534.
32. S. Borm and R. Hiptmair, Analysis of tensor product multigrid, *Numerical Algorithms* 26 (2001), 219–234.
33. W. Hackbusch, The frequency decomposition multigrid method, part I: Application to anisotropic equations, *Numer Math* 56 (1989), 229–245.
34. D. Mavriplis, Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes, NASA/CR-1998-206910, 1998.
35. W. Mulder, A high resolution Euler solver based on multigrid semi-coarsening and defect correction, *J Comput Phys* 100 (1992), 91–104.
36. Y. Saad, *Iterative methods for sparse linear systems*, PWS Publishing Company, 1996.
37. M. Brezina, A. Cleary, R. Falgout, V. Henson, J. Jones, T. Manteuffel, S. McCormick, and J. Ruge, Algebraic multigrid based on element interpolation, *SIAM J Sci Comp* 22 (2000), 1570–1592.
38. T. Chan, L. Zikatanov, and J. Xu, An agglomeration multigrid method for unstructured grids, *Cont Math* 218 (1998), 67–81.
39. J. Ruge and K. Stuben, Algebraic multigrid, in *Multigrid methods*, Frontiers in applied mathematics, Philadelphia, SIAM, 1987, pp 73–130.
40. U. Trottenberg, C. W. Oosterlee, and A. Schuller, *Multigrid*, Academic Press, 2000.
41. P. Vanek, M. Brezina, and J. Mandel, Convergence of algebraic multigrid based on smoothed aggregation, *Numer Math* 88 (2001), 559–579.
42. Q. Chang and Z. Huang, Efficient algebraic multigrid algorithms and their convergence, *SIAM J Sci Comput* 24 (2002), 597–618.
43. Z. Huang, G. Lei, and X. Liu, High-order PCG method solving complex systems, *Chinese J Comput Phys* 17 (2000), 401–406.
44. Z. Huang and Q. Chang, Gauss-Seidel-type multigrid methods, *J Comput Math* 21 (2003), 421–434.