SIAM J. SCI. COMPUT.
Vol. 26, No. 4, pp. 1146–1165

MULTILEVEL SOLVERS FOR UNSTRUCTURED SURFACE MESHES*

BURAK AKSOYLU†, ANDREI KHODAKOVSKY†, AND PETER SCHRÖDER†

Abstract. Parameterization of unstructured surface meshes is of fundamental importance in many applications of digital geometry processing. Such parameterization approaches give rise to large and exceedingly ill-conditioned systems which are difficult or impossible to solve without the use of sophisticated multilevel preconditioning strategies. Since the underlying meshes are very fine to begin with, such multilevel preconditioners require mesh coarsening to build an appropriate hierarchy. In this paper we consider several strategies for the construction of hierarchies using ideas from mesh simplification algorithms used in the computer graphics literature. We introduce two novel hierarchy construction schemes and demonstrate their superior performance when used in conjunction with a multigrid preconditioner.

Key words. multilevel preconditioning, multigrid, hierarchical basis multigrid, Bramble–Pasciak–Xu, computer graphics, unstructured surface mesh, surface parameterization, harmonic weights, mean value weights, mesh coarsening

AMS subject classifications. 65Y20, 65F10, 65N55, 65D18, 65M50

DOI. 10.1137/S1064827503430138

1. Introduction.

Unstructured triangle meshes which approximate surfaces of arbitrary topology (genus, number of boundaries, number of connected components) appear in many application areas. Examples range from iso-surfaces extracted [54] from volumetric imaging sources and scientific simulations [62] to surfaces produced through range scanning techniques (e.g., [10, 51]) in areas as varied as historical preservation, reverse engineering, and entertainment. These meshes can be quite detailed: 100,000 samples, i.e., point positions on the surfaces, are quite common with many datasets ranging into the millions and some even into billions [51] of samples. Processing such meshes efficiently, in particular when numerical simulation algorithms are involved, requires sophisticated solvers.

One of the most fundamental issues in the processing of such geometry is the establishment of a *parameterization*, i.e., the construction of functions from sections of the surface to regions in $\mathbb{R}^2$. For example, parameterizations are critical for the approximation of one surface by another (e.g., [49, 48, 79]), numerical simulation of the mechanics of such surfaces (e.g., [31, 5]), their resampling (e.g., [22, 50, 36]), editing [47], or just plain decoration ("texture mapping") [69].

Most parameterization algorithms define a good parameterization as one which minimizes some energy functional [22, 42, 71, 34, 32, 73, 30, 64, 44]. The energy serves to encode measures such as low distortion [68, 69], conformality [38, 15, 52, 33], area preservation [15], or elastic energy [55]. Others define the solution to satisfy barycentric coordinate conditions [24, 26], which take the original triangle shapes into account.

*Received by the editors June 17, 2003; accepted for publication (in revised form) April 14, 2004; published electronically March 11, 2005. This work was supported in part by NSF (DMS-0220905, DMS-0138458, ACI-0219979), the DOE (W-7405-ENG-48/B341492), nVidia, the Center for Integrated Multiscale Modeling and Simulation, Alias|Wavefront, Pixar, Microsoft, and the Packard Foundation.

http://www.siam.org/journals/sisc/26-4/43013.html

†Department of Computer Science, California Institute of Technology, Pasadena, CA 91125 (baksoylu@cs.caltech.edu, akh@cs.caltech.edu, ps@cs.caltech.edu).

A basic building block of all of these parameterization algorithms is the solution of sparse linear systems, which are defined relative to the input mesh, i.e., they contain as many degrees of freedom (DOFs) as vertices. Such systems tend to be ill-conditioned for large meshes, and standard iterative solver methods typically take very long to converge or even fail to converge. This comes as no surprise since the leading order terms of most of these parameterization algorithms correspond to discretizations of second-order order elliptic PDEs. For such operators, appropriate preconditioning methods have long been studied, and optimal methods are well known [37].

These preconditioners are generally based on hierarchical representations, which are easy to come by in the standard structured refinement setting (e.g., quadrasection or longest edge bisection refinement [65, 66]). However, in the cases of interest to us, we are confronted with a setting in which we are handed a very fine, unstructured mesh, and any kind of hierarchical solver machinery must address the issue of coarsening, not refinement.

1.1. Contributions. In this paper we construct effective multilevel preconditioners in the unstructured, 2-manifold (with boundary) triangle mesh setting. As example problems we consider the linear systems arising in mesh parameterization algorithms, both symmetric and nonsymmetric. Our hierarchy construction is based on mesh simplification methods that use vertex removal [19] as their primitive operations. These simplification methods guarantee a logarithmic number of levels, i.e., geometric decay of DOFs from finest to coarsest. To avoid the problems of retriangulation of polygonal holes in $\mathbb{R}^3$ we implement vertex removal through half-edge contractions [40, 18]. We propose three hierarchy constructions, which exhibit different decay rates in the number of DOFs. Two of these constructions are entirely novel and perform *exceedingly* well on problems of interest. The different hierarchies are compared vis-à-vis different multilevel preconditioning strategies. Our novel, fast decaying, unstructured *maximal independent set (MIS)* hierarchy, coupled with the multigrid (MG) preconditioner, exhibits by far the best overall runtime over a range of problem sizes.

While we were originally motivated by the need to efficiently solve the linear systems arising in the construction of smooth parameterizations for large, unstructured triangle meshes, we hasten to point out that the resulting preconditioning algorithms are applicable to general elliptic second-order problems. The so-called harmonic weights result from a finite element discretization of the Laplace–Beltrami operator, i.e., the Laplace operator subject to a given induced surface metric. Our algorithms and results thus apply to *any* problem that involves finite element approaches to the solution of the Laplace(–Beltrami) operator. This includes many geometric PDEs as used widely in image and geometry processing, as well as many physical simulations (e.g., behavior of thin flexible membranes).

1.2. Outline. In section 2 we present the mathematical framework for surface parameterization and elaborate how the linear system arises from the parameterization process. The parameterization weights of interest are *harmonic* and *mean value* weights, and we provide some background on them. The mesh coarsening process is described in section 3, together with the methods we use to construct *fast*, *medium*, and *slow* decaying hierarchies using the notion of MISs in a variety of ways. The construction of prolongation matrices is elaborated in section 4. The different preconditioners we consider are described in section 5, together with their relations to one another. To understand the observed performance better, we analyze the sparsity fill-in for the different hierarchies in section 6. Finally, section 7 documents our nu-

merical experiments and the superiority of the MG preconditioner coupled with our novel fast decaying MIS hierarchy. Conclusions in section 8 close the paper.

2. Parameterization. We begin by fixing our notation in describing the basic parameterization setup. A 2-manifold triangle mesh of arbitrary genus, possibly with boundary, is given as a simplicial complex $M = (V, E, T)$. This mesh is (typically) embedded in $\mathbb{R}^3$, i.e., each vertex $v_i \in V$ has associated with it a point position $p_i \in \mathbb{R}^3$. The point positions are extended in a piecewise linear fashion using the incidence relations given by the set of edges, $e_{ij} \in E$, and triangles, $t_{ijk} \in T$. For surfaces of arbitrary topology, not equivalent to a disk, parameterizations are typically computed by combining a sequence of individual parameterization problems for disklike subsections of the surface [22, 34, 44]. To simplify the exposition in this paper, we will consider M to be a (sub)mesh topologically equivalent to a disk.

A *parameterization* of M is a piecewise linear injective map from the embedded surface to a region $\Omega \subset \mathbb{R}^2$:

$$\psi : \mathbb{R}^3 \supset M \to \Omega \subset \mathbb{R}^2.$$

Ω may be thought of as a flattened version of the same mesh (see Figure 2.1), i.e., it shares the same combinatorial structure (V, E, T). Consequently ψ is fully specified by the values it takes on the vertices $v_i \in V$, $\psi(v_i) \in \Omega$. Note that the injectivity of ψ implies that no triangle is "flipped" under the parameterization.

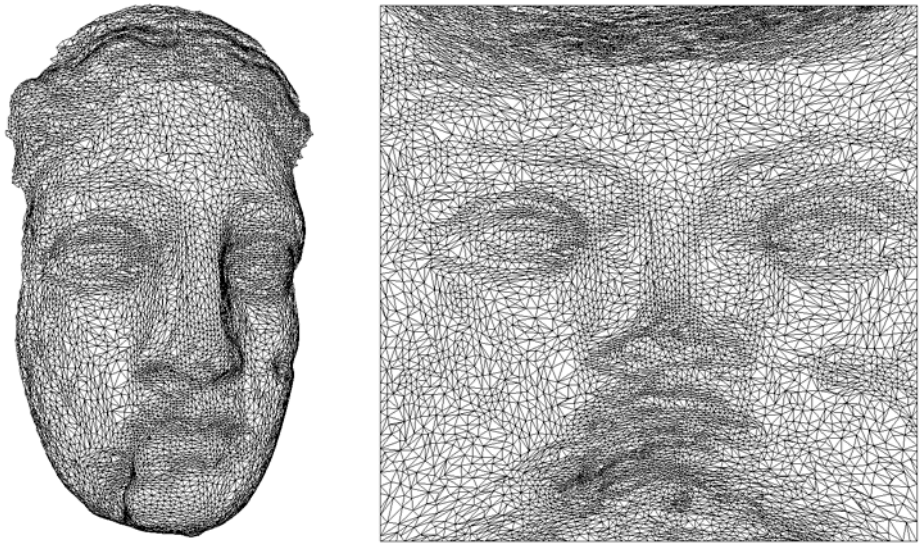

FIG. 2.1. *Igea face (model 2) and its parameterization with harmonic weights.*

Let V_I and V_B denote the interior and boundary vertices of V and 1-R_i denote the 1-*ring*, i.e., the set of edge neighbors, of v_i, 1-$R_i = \{v_j | e_{ij} \in E\}$. The commonly described approaches to solving for a ψ with desired properties all reduce to solving a sparse linear system.[1] These are defined by specifying for each vertex $v_i \in V_I$ a set

[1] In those cases in which ψ is given as the solution of some nonlinear functional, we generally get a sequence of linear systems of the same basic structure.

of weights λ_{ij}, one for each vertex $v_j \in 1\text{-}R_i$. The parameter points $\psi(v_i) \in \Omega$ are then found by solving the linear system

$$\psi(v_i) \sum_{v_j \in 1\text{-}R_i} \lambda_{ij} - \sum_{v_j \in 1\text{-}R_i \cap V_I} \lambda_{ij}\psi(v_j) = \sum_{v_j \in 1\text{-}R_i \cap V_B} \lambda_{ij}\psi(v_j), \quad v_i \in V_I. \tag{2.1}$$

This can be written as the matrix equation

$$Ax = b, \tag{2.2}$$

where $x = \{\psi(v_i)\}_{v_i \in V_I}$ is the vector of DOFs in some arbitrary order and b is the right hand side of (2.1). The matrix $A = \{a_{ij}\}_{v_i, v_j \in V_I}$ has dimension $N \times N$ with $N = |V_I|$ and entries

$$a_{ij} = \begin{cases} \sum_{v_k \in 1\text{-}R_i} \lambda_{ik}, & i = j, \\ -\lambda_{ij}, & v_j \in 1\text{-}R_i, \\ 0 & \text{otherwise.} \end{cases}$$

Note that λ_{ij} may or may not be equal to λ_{ji}, leading respectively to symmetric or nonsymmetric A. If all λ_{ij} are nonnegative, $\psi(v_i)$ lies in the convex hull of its neighbors $\psi(v_j)$ ($v_j \in 1\text{-}R_i$), which, taken together with a convex boundary mapping for all V_B, ensures the injectivity of the solution [76] (see also Floater [25] for a more recent, simpler proof). For example, the simple set of weights $\lambda_{ij} = 1$ has this property. Unfortunately these weights take none of the geometry (angles, lengths, sampling, etc.) of the original embedding into account. If the map is to have properties such as angle or area preservation, the weights must depend on the geometry of the embedded mesh.

For purposes of this article we consider two different types of weights: *harmonic* [63] and *mean value* [26]. Harmonic weights are defined by

$$\lambda_{ij} = \lambda_{ji} = \cot a_k + \cot a_l,$$

where a_k and a_l are the angles opposite e_{ij} in the two incident triangles t_{kji} and t_{ijl} (see Figure 2.2). Note that boundary edges ($v_i, v_j \in V_B$), which have only one incident triangle, do not appear, ensuring that the λ_{ij} are well defined. Boundary conditions can be of the Dirichlet type—with prescribed mappings of the V_B—or of a Neumann ("natural") type [15].

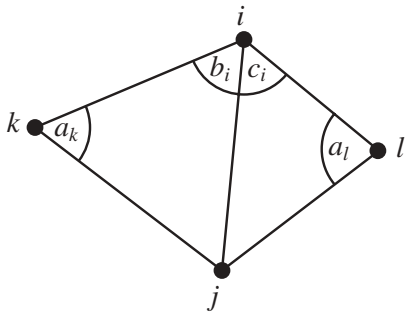

FIG. 2.2. *Angles used in the definition of the parameterization weights.*

The harmonic weights arise from the standard piecewise linear finite element approximation of the Laplace–Beltrami operator (LBO). LBO is a self-adjoint nonlinear

second-order elliptic operator given as

$$\Delta_M f = \frac{1}{\sqrt{\det\, G}} \sum_{i,j} \frac{\partial}{\partial \xi_i}\, \sqrt{\det\, G}\, g^{ij} \frac{\partial f}{\partial \xi_j},$$

where $G = (g_{ij})$ is the metric tensor and g^{ij} is defined as $G^{-1} = (g^{ij})$. LBO is the generalization of the usual Laplace operator to a smooth surface with the given metric. It is exactly the Laplace operator in the Euclidean setting. Since the underlying continuous operator is a self-adjoint second-order elliptic operator, the condition number of the discrete system behaves as $O(h^{-2})$.

Our method is applicable to a large collection of application areas where the underlying operator is LBO (or Laplace operator in flat space): image processing, signal processing, surface processing [16, 17], and the study of geometric PDEs [82, 57]. Moreover, mathematical formulations of mean curvature flow [12, 13, 21], surface diffusion flow [23, 57], gradient flow [56], and Willmore flow [58] involve LBOs. Similarly, the treatment of Riemannian surface structures relies on finite element approximations of LBO [33]. A general variational framework for surface curvature dependent energy densities can be found in [61]. A survey of existing discretizations and application areas of LBO can be found in [81].

The harmonic weights are based on the fact that harmonic functions minimize the *Dirichlet energy* [63] with $f : \Omega \to M$:

$$E(f) = \int_\Omega \|\nabla f\|_g^2,$$

where g is the metric on the surface M induced by (g_{ij}) and $\|\nabla f\|_g^2 = \text{trace}\, g(\partial f., \partial f.)$. The system arising from the harmonic weights is symmetric and positive definite [63]. However, a given triangulation may lead to negative coefficients λ_{ij} in the presence of obtuse angles. In that case the matrix may become positive semidefinite. The weights depend smoothly on the points $p_i \in \mathbb{R}^3$, i.e., the embedding.

Mean value weights are derived from an application of the mean value theorem for harmonic functions and defined as

$$\lambda_{ij} = \frac{\tan(b_i/2) + \tan(c_i/2)}{\|p_i - p_j\|}.$$

Angles used in the construction of the mean value weights do not allow a symmetric system. However, the weights are guaranteed to be positive. Taken together with a convex embedding of the boundary of M into $\partial\Omega$, this property guarantees an injective ψ. Assuming that all edge lengths are bounded away from zero, so are the λ_{ij}, which may help in the conditioning of the linear system (2.2). As in the harmonic weights case, the weights depend smoothly on the points $p_i \in \mathbb{R}^3$.

Unlike the harmonic weights, there is no established connection between the mean value weights and a continuous operator. Maybe partially due to this fact, there is no available condition number estimate in the literature for the resulting linear system. Empirically we observe that the linear systems are as badly conditioned as the ones arising from the discretization of LBO.

3. Mesh coarsening. Given that our linear systems are too large to begin with, hierarchy construction implies coarsening strategies. We borrow algorithms from the well-developed area of *mesh simplification* for this purpose and give a brief background in this section.

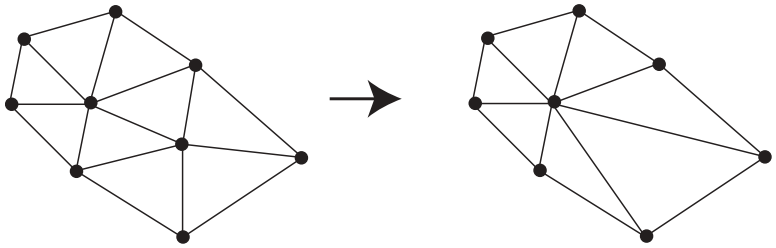

FIG. 3.1. *Edge contraction removes one vertex, three edges, and two triangles by making the end points of one edge coincident.*

The development of mesh simplification methods was initiated to deal with increasingly fine sampled surface meshes. Hoppe [40] introduced *progressive meshes*, which are built through a greedy strategy of topology preserving edge contractions [18] (see Figure 3.1), prioritized according to the geometric error introduced between the coarser mesh and the original mesh [27]. Criteria which also take triangle shape and curvatures into account were examined by Kobbelt, Campagna, and Seidel [46]. Such modifications may be particularly useful to ensure good mesh conditioning for the finite element method [4, 72].

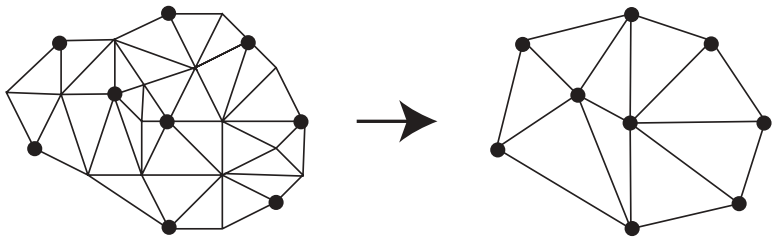

FIG. 3.2. *Marked vertices in the left mesh denote a maximal independent set. On the right, a retriangulation of the independent set, which is used in our fast MIS hierarchy construction.*

Each edge contraction removes one vertex, three edges, and two triangles. Such a sequence of edge contractions can be "played" back and forth allowing a traversal of a linear number of mesh approximations. A partial order of topological dependencies between the contractions—a given contraction cannot be undone before some neighboring contractions have been undone—can be used to induce a discrete set of "levels" of meshes between finest and coarsest [80, 41]. While the number of levels is "reasonable" in practice, no guarantees are made as to the asymptotic number of such levels. This is not surprising, since the choice of simplification order and criteria in these algorithms depends on the geometry (embedding). There are other simplification strategies such as vertex removal [70] as well as triangle removal [29].

Guarantees as to the number of levels of a fine to coarse hierarchy—and the cost of its construction—can be made by vertex removal strategies based solely on combinatorial considerations. Such methods have been employed in computational geometry for the construction of asymptotically optimal planar point location [45] and spatial geometric queries [19]. The so-called Dobkin–Kirkpatrick (DK) hierarchy is constructed by removing a maximal, *independent set* of vertices from a given triangle mesh (see Figure 3.2, left). A subset of vertices, $V_o \subset V$, is said to be independent if for any $v_i, v_j \in V_o$, $e_{ij} \notin E$. It is maximal if addition of any vertex $v_k \in V \setminus V_o$ to the set

V_o would violate its independence property. Each removed vertex leaves a polygonal hole which is subsequently retriangulated. This process removes one vertex, three edges, and two triangles per independent vertex. By repeatedly removing a maximal independent set, one arrives at a full hierarchy.

There are many different ways to implement the DK hierarchy. Selection of the maximal independent set of vertices is typically performed through some sweep-and-mark algorithm. The classical approach is purely combinatorial, i.e., it does not take any property of the embedding into account. This can lead to deteriorating aspect ratios as pointed out by Miller, Talmor, and Teng [59]. To enable control over the quality of the coarser triangulation, we prioritize half-edge contractions based on edge length (effectively removing the shortest edges first) to favor uniform triangle sizes at a given level of the hierarchy.

For a given edge we check the incident vertices for their status: unmarked, independent, or dependent. Initially all vertices are unmarked. If one incident vertex of an edge is independent, the other is marked dependent if still unmarked. Otherwise we mark an unmarked vertex as independent and the other endpoint as dependent. If both endpoints are unmarked, we favor the vertex of higher valence as independent. This biasing toward higher valence tends to result in smaller maximally independent sets of vertices. While this leads to slower decaying DK hierarchies, it also leads to faster decay in our MIS hierarchy (see below). After the marking sweep, independent vertices are removed through a half-edge contraction into one of their dependent neighbors (there is always at least one such neighbor). For the construction of prolongation operators, we also record all dependent vertices in the 1-ring of an independent vertex.

Using the Euler characteristic of a planar mesh, one can show that $|V_o| \geq c|V|$ for some $1 > c \geq c_0 > 0$ for any planar mesh with $|V| > n_0 > 0$ and c_0 independent of the particular mesh. Four-colorability of a planar graph ensures that $c_0 = 1/4$ is possible. Randomized greedy selection strategies can achieve this bound in empirical practice, though the theoretical guarantees only assert $c_0 \geq 1/24$ for such strategies. These strategies favor low valence vertices for removal. With our strategy of favoring high valence vertices, we observe an empirical $c_0 \approx .21$. Whatever the exact factor, it implies that a sequence of coarser meshes can be constructed with $J = O(\log |V|)$ levels.

The decay rate—going from fine to coarse—of the thus constructed hierarchies is much slower than the rates achieved in quadrasection refinement: .75 compared to .21. In analogy to the regular refinement setting of coarse-to-fine mesh hierarchies, one would like to use coarsening strategies that remove a larger fraction of all vertices.

Consider quadrasection refinement of a triangle mesh. In that case each old vertex is surrounded by a set of new vertices in the finer mesh: the old vertices form a maximal independent set with respect to the finer mesh. This observation suggests the construction of a hierarchy, which turns the DK removal strategy on its head. Instead of removing the maximal independent set, we form the coarser level by removing all *other* vertices. We will refer to this novel hierarchy construction as the *MIS hierarchy*.

Exchanging dependent and independent markings, the coarser level is once again built by performing half-edge contractions after a sweep-and-mark pass. Note that the prolongation list will always contain at least one vertex but may not contain more than that. In practice we found that approximately 1/3 of all removed vertices fall into this category. For such vertices the prolongation matrices have rows with a single nonzero entry. To evaluate the impact of these degenerate rows on the solver

convergence, we consider a *medium* hierarchy in between the *slow* DK and *fast* MIS hierarchies. The medium hierarchy is constructed in exactly the same way as MIS, but we prohibit the removal of vertices which have only a single entry on their prolongation list. Naturally, such a hierarchy decays slower then MIS but still faster then DK. We observed a decay rate of .51, compared to .21 and .80 for the MIS and DK hierarchies, respectively. Figure 3.3 shows the decay rate for two models. (Other models exhibit the same rates.)

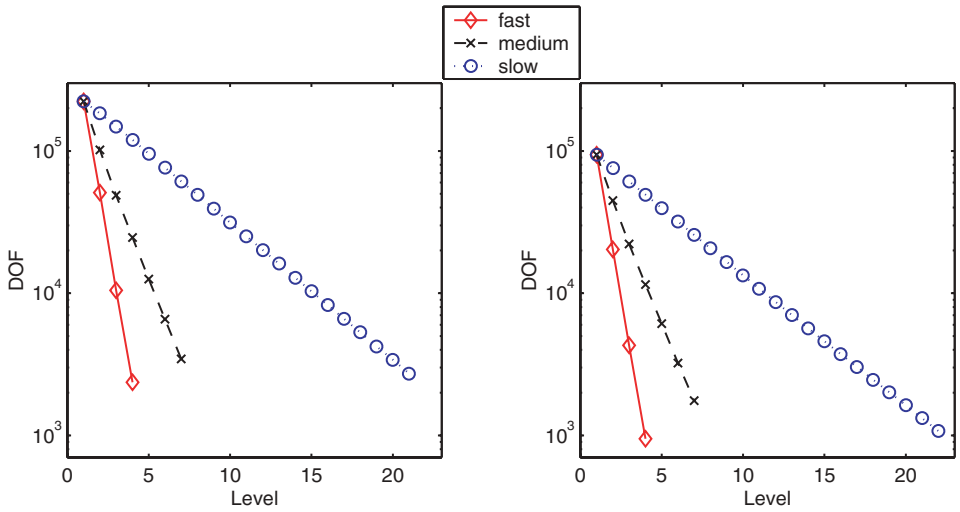

FIG. 3.3. *The rate of decay in the number of DOFs for model* 2 *(left) and model* 3 *(right). Average decay rates of* .21, .51, .80 *are observed for fast (MIS), medium (modified MIS), and slow (DK) hierarchies, respectively.*

Boundary vertices are further qualified in the removal strategy. A half-edge contraction is not performed if it would contract a boundary vertex into an interior vertex. This ensures a better representation of the boundary during coarsification. In some settings it may also be desirable to fix some boundary vertices, typically corners, as unremovable throughout the hierarchy. Other cases in which an edge contraction is not performed are those which would change the global topology of the mesh (see [18]). In some applications it may also be desirable to incorporate geometric measurements into the removal criteria. Examples include quality measures such as triangle aspect ratio or whether the coarser surface has self intersections. Such modifications can, of course, change the observed decay rates.

Alternative approaches such as geometrical coarsening of unstructured planar meshes [4, 8, 60] or algebraic coarsening techniques [43, 77] have been studied in the numerical PDE community. These techniques usually employ *agglomeration* and *aggregation* strategies. The approach taken by Bank and Xu [4] is fundamentally different from our approach, and it is unique in the sense that they force an arbitrary unstructured mesh into a nonuniform and locally refined mesh, enabling them to impose a logically nested (but not physically nested) structure on the mesh. Except for the algebraic coarsening techniques, these algorithms are generally more complicated than our approach, and it is unclear whether they can be generalized to the nonplanar setting. Purely algebraic techniques offer an alternative, albeit at the cost of giving up knowledge of the embedding in choosing the best coarsification steps.

4. Prolongation operators. We now have three different hierarchies at our disposal—slow (DK), medium (modified MIS), and fast (MIS)—and will examine their behavior vis-á-vis different multilevel preconditioning strategies. As in the standard multigrid framework, coarser representations of the finest level system matrix are formed algebraically by the triple matrix product (also known as the *variational condition*):

$$A^{(j)} = (P_j^{j-1})^T A^{(j-1)} P_j^{j-1}, \quad j = 1, \ldots, J, \tag{4.1}$$

where j corresponds to a coarser level than $j-1$ and P_j^{j-1} is the prolongation operator from level j to $j-1$. P_j^{j-1} is of size $N_{j-1} \times N_j$, where N_j denotes the number of DOFs at level j. There are many possible predictor choices which can be used for the prolongation operator: centroid, inverse edge length, Guskov [35] (divided differences), Desbrun [16], or energy minimizing [78] predictors. Gieng reports [28] that the performance of the multilevel preconditioners is not dramatically affected by the predictor choice. Hence we use the simplest one: *centroid prediction* (see Figure 4.1). (For fourth-order problems, however, the predictors of Guskov and Desbrun may be more appropriate.)

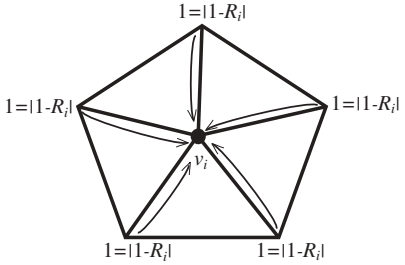

FIG. 4.1. *For this example, the prolongation matrix row corresponding to v_i has entries $1/|1\text{-}R_i| = 1/5$.*

The rows corresponding to the coarse DOFs will form an identity matrix I of size $N_{j-1} \times N_{j-1}$, and a row in R_j^{j-1} corresponding to a newly introduced DOF—or vertex v_i—will contain entries equal to $1/|1\text{-}R_i|$ (where $|1\text{-}R_i|$ is the number of DOFs in the prolongation list of v_i) in the appropriate columns. All other entries vanish.

$$P_j^{j-1} = \begin{bmatrix} I \\ R_j^{j-1} \end{bmatrix}. \tag{4.2}$$

5. Preconditioners. Most of the parameterization work published in the computer graphics literature has focused on different approaches to the formulation of the system matrix. Little effort has been devoted to effective numerical solvers, though the need to go beyond simple diagonal preconditioning has been pointed out repeatedly (see for example [52]). For example, the work of Liesen et al. [53] concentrates on a constrained minimization approach with single-level preconditioning in the form of a Krylov subspace method (for more details see [13]). In contrast, our preconditioners exploit a full multilevel hierarchy. Duchamp and coworkers [20] did use a full hierarchical approach to compute piecewise linear harmonic embeddings. They constructed *lazy wavelets* [75, 74] induced by a DK hierarchy and considered a conjugate gradient solver in the wavelet domain. Empirically, this reduced the number of iterations from

linear to logarithmic, similar to what is found when using a hierarchical basis [84] for the solution of second-order elliptic problems. In our construction we observe a constant number of iterations for the MG preconditioner as expected.

The systems arising in parameterization problems are well known to be ill-conditioned, with (bi)conjugate gradient methods often failing for systems beyond $\approx 50,000$ DOFs. Some simple hierarchical preconditioning (naïve Bramble–Pasciak–Xu (nBPX); see below) has been used since it is easy to integrate with mesh simplification approaches without the need to build coarser level system matrices (see, e.g., [47]). In our comparison of methods, we will include this strategy.

Multilevel preconditioners are usually classified as either multiplicative or additive Schwarz methods. Multiplicative preconditioners are designed to update the residual at every level throughout the multilevel hierarchy, whereas the additive ones update the residual after a full sweep of the multilevel hierarchy. We are going to employ two multiplicative Schwarz methods: the multigrid (MG) and the hierarchical basis multigrid (HBMG) [3] preconditioners. Essentially MG and HBMG are the same preconditioner, with the difference lying in the DOFs used for the smoothing iteration (see Algorithm 6.1). MG sweeps all DOFs at a given level, whereas HBMG sweeps the ones that are newly introduced at that level. Hence, HBMG is very attractive for adaptive regimes where the storage complexity is optimal and the computational complexity is close to optimal (in two spatial dimensions). Based on the decay rate of the hierarchies, different multilevel preconditioners are appropriate. Traditional MG is most appropriate for fast decaying hierarchies, while the HBMG method is more appropriate for slow decaying hierarchies. (In the traditional refinement setting, the HMBG method is used for highly adaptive refinement with only a small constant number of DOFs added when going from coarse to fine.) In either case the system is solved with the help of the (bi)conjugate gradient method as an outer accelerator. The iteration counts we report give the number of iterations of the (bi)conjugate gradient method. The computational complexities of several multilevel preconditioners were discussed in detail in [1]. Both MG and HBMG act as standard V-cycle iteration with one symmetric Gauss–Seidel iteration as smoother.

Additive versions of MG and HBMG preconditioners are the Bramble–Pasciak–Xu (BPX) [6] and the hierarchical basis (HB) [84] preconditioners, respectively. The action of the classical BPX preconditioner [1, 83] in two dimensions can be written in matrix form as

$$X_{\mathrm{B}PX} = \sum_{j=0}^{J-1} P_j S_j P_j^T + P_J (A^J)^{-1} P_J^T, \tag{5.1}$$

where $(A^J)^{-1}$ represents a coarsest level direct solve. The prolongation matrix from level J to j is denoted by

$$P_j \equiv P_j^0 = P_1^0 \dots P_j^{j-1} \in \mathbb{R}^{N_0 \times N_j}.$$

P_0^0 is the identity matrix $I \in \mathbb{R}^{N_0 \times N_0}$, P_j^{j-1} is the prolongation matrix from level j to $j-1$, and S_j is the smoother. The HB preconditioner in two dimensions can be expressed as

$$X_{\mathrm{H}B} = \sum_{j=0}^{J-1} H_j S_j H_j^T + H_J (A^J)^{-1} H_J^T, \tag{5.2}$$

where H_j are the tails of the P_j corresponding to newly introduced DOFs. In other words, with $J+1=0$, $H_j \in \mathbb{R}^{N_0 \times (N_j - N_{j+1})}$, $j = 0, \ldots, J$, is given by keeping only the columns that correspond to new DOFs (the last $N_j - N_{j+1}$ columns of P_j). Notice that the smoother S_j in (5.2) acts only on the newly introduced DOFs.

The additive method which we denote as the nBPX preconditioner is a variant of BPX. It takes only the finest level term in (5.1) and drops the smoother and the coarsest level direct solve:

$$X_{\text{naïve}} = P_J P_J^T. \tag{5.3}$$

As mentioned in section 2, the harmonic and mean value weights give rise to symmetric positive definite and nonsymmetric systems; hence the preconditioners above are coupled with conjugate gradient and biconjugate gradient methods, respectively. The solvers for the system formed by using the harmonic weights are provably guaranteed to converge. We have full-rank prolongation matrices and convergent smoothing iterations; then through the use of the variational conditions applied to the symmetric positive definite system the convergence of the solver is guaranteed [67].

All the above preconditioners are *mesh oriented* in the sense that the prolongation operators are created using the geometry and connectivity information of the mesh. An alternative would be the use of algebraic multigrid (AMG) methods [7], which do not use any geometric knowledge about the mesh, though the connectivity of the mesh enters at the finest level through the associated sparsity structure of the system matrix (2.2). AMG methods may be attractive for parameterization problems. However, the original AMG theory was developed for Stieltjes matrices (i.e., $A \in \mathbb{R}^{N_0 \times N_0}$ is symmetric positive definite and has nonpositive off-diagonal entries). While Stieltjes matrices are not a requirement for AMG to work, matrices A, which are far from being Stieltjes, may render AMG less effective [7]. We do not pursue AMG methods further here, in particular because they do not provide explicit control over the construction and quality of the coarsification hierarchy. (For some recent work employing AMG for mesh partitioning and multilevel surface editing, see the work of Clarenz et al. [9].)

6. Sparsity. When comparing different preconditioning strategies, one can evaluate their performance in terms of the number of iterations required by the solver. However, this does not tell the whole story, since the time to solution is a function of both the number of iterations and the sparsity structure of the matrices appearing in the multilevel hierarchy. In this section we take a closer look at the fill-in of the matrices involved in the actions of MG and HBMG preconditioners. (Note that the question of sparsity does not arise for the nBPX, since it does not use any matrix structures.) The sparsity will help to explain the overall superior performance of the MG preconditioner in conjunction with the fast (MIS) hierarchy.

To evaluate the sparsity we consider the average number of nonzero entries per row. We observe relatively small and uniform standard deviation of this measure, making it a good predictor of performance. In Figure 6.1 we plot the number of nonzero entries in the matrices formed by the variational conditions (4.1) for fast, medium, and slow hierarchies. For fast and medium hierarchies, fill-in increases approximately linearly with level (two models are shown; others exhibit the same behavior). The slow hierarchy fill-in is also linear with respect to level over a wide range, but it is very rapid and the curves flatten out as the matrices become dense. At the finest level, the matrices have seven nonzeros on average as a consequence of the Euler characteristic. At the coarsest level, the fast, medium, and slow hierarchies produce an average of 12, 30, and 190 nonzero entries, respectively. This easily indicates that sparsity is the

primary factor responsible for the poor performance of slow hierarchies. Also note how the medium hierarchy leads to significantly more (almost three times more) fill-in compared to the fast hierarchy.

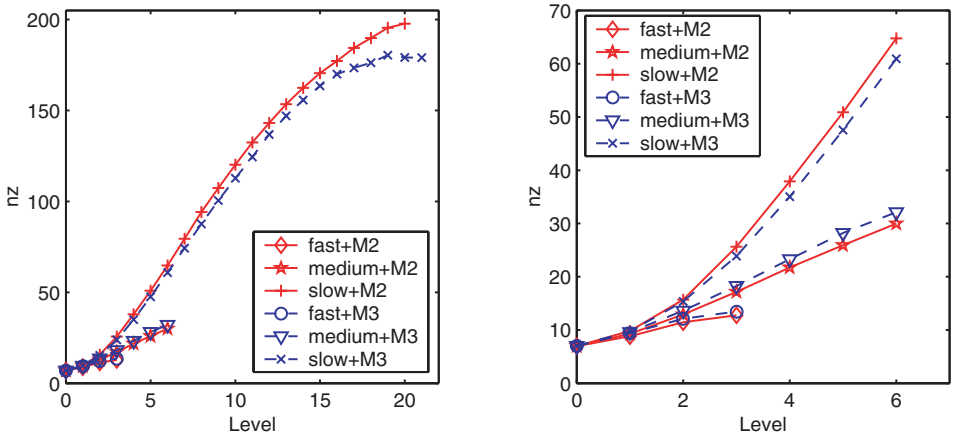

FIG. 6.1. *Sparsity of the system matrix for models* 2 *and* 3 *using fast, medium, and slow hierarchies (left). Close-up of the sparsity between levels* 0 *and* 6 *(right). Notice the rapid fill-in corresponding to the slow hierarchy.*

For slow-decaying hierarchies it is more appropriate to use the HBMG preconditioner, which only operates on some of the DOFs at each level. Let $A^{(j)}$ be represented by a two-by-two block form [2],

$$A^{(j-1)} = \begin{bmatrix} A^{(j)} & A_{12}^{(j-1)} \\ A_{21}^{(j-1)} & A_{22}^{(j-1)} \end{bmatrix}, \tag{6.1}$$

where $A^{(j)}$, $A_{12}^{(j-1)}$, $A_{21}^{(j-1)}$, and $A_{22}^{(j-1)}$ correspond to coarse-coarse, coarse-fine, fine-coarse, and fine-fine interactions, respectively. The HBMG method utilizes a change-of-basis operation resulting in the so-called *stabilized* blocks (represented with hats). Dropping the superscripts for simplicity, the blocks are expressed as

$$\begin{bmatrix} \hat{A}_{11} & \hat{A}_{12} \\ \hat{A}_{21} & \hat{A}_{22} \end{bmatrix} = \begin{bmatrix} I & R^T \\ 0 & I \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} I & 0 \\ R & I \end{bmatrix},$$

$$\begin{aligned} \hat{A}_{11} &= A_{11} + A_{12}R + R^T A_{21} + R^T A_{22} R, \\ \hat{A}_{12} &= A_{12} + R^T A_{22}, \\ \hat{A}_{21} &= A_{21} + A_{22} R, \\ \hat{A}_{22} &= A_{22}. \end{aligned}$$

In terms of these stabilized blocks, the HBMG algorithm can be interpreted as an iterative process for solving the system (2.2), where $x = P\hat{x}$, $\hat{b} = P^T b$.

ALGORITHM 6.1.

1. *smooth* $\hat{A}_{22}\hat{x}_2 = \hat{b}_2$
2. *form residual* $\hat{r}_1 = \hat{b}_1 - (\hat{A}_{11}\hat{x}_1) - \hat{A}_{12}\hat{x}_2$
3. *solve* $\hat{A}_{11}\hat{x}_1 = \hat{r}_1$
4. *prolongate* $\hat{x} = \hat{x} + P\hat{x}_1$
5. *smooth* $\hat{A}_{22}\hat{x}_2 = \hat{b}_2 - (\hat{A}_{21}\hat{x}_1)$

This can be further simplified by transforming the linear system (2.2) into the equivalent system

$$A(x - x_j) = b - Ax_j,$$

with an initial guess of x_j. In this setting, the initial guess is zero, and the HBMG algorithm recursively iterates toward the *error* with given *residual* on the right hand side. In that case the terms in parentheses in Algorithm 6.1 are zero.

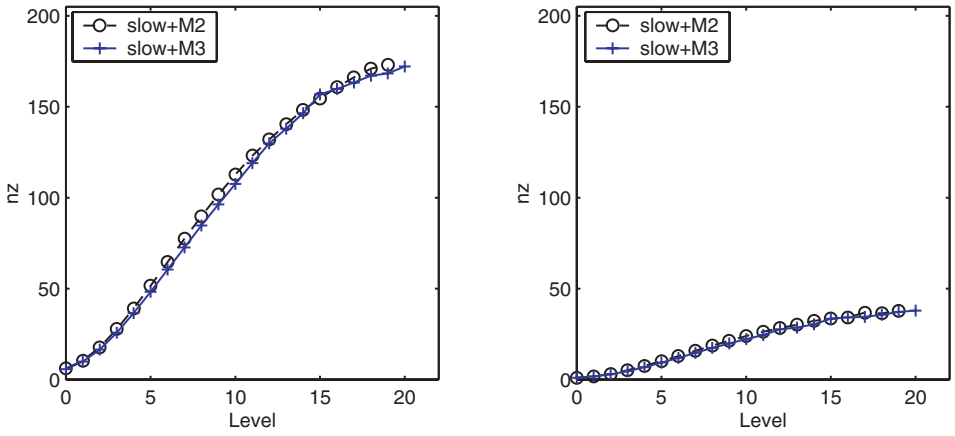

FIG. 6.2. $\hat{A}_{12}$ *sparsity (left) and* $\hat{A}_{22}$ *sparsity (right) for slow hierarchy for model* 2 *and* 3.

In the HBMG method—a multiplicative preconditioner—the residual is computed at every level (see step 2 in Algorithm 6.1), therefore the sparsity of the $\hat{A}_{12}$ block plays a crucial role. Similarly, the cost of the smoothing step is related to the sparsity of $\hat{A}_{22}$. Figure 6.2 shows the sparsity of the $\hat{A}_{12}$ and $\hat{A}_{22}$ blocks. As before, the fill-in increases linearly with level over a wide range. Eventually the matrices become quite dense so that the curves flatten out, similar to what we observed for A in the slow hierarchies. At the finest level, $\hat{A}_{12}$ contains six nonzero entries and $\hat{A}_{22}$ has only one nonzero. As the hierarchy reaches the coarsest level, HB stabilization produces 173 and 38 nonzeros in the $\hat{A}_{12}$ and $\hat{A}_{22}$ blocks, respectively. The ratio of nonzero entries in $\hat{A}_{12}$ over $\hat{A}_{22}$ is roughly 4.5 with an almost identical slope. This ratio is due to the $\hat{A}_{12}$ block having many initial nonzero entries before HB stabilization is in effect.

7. Numerical experiments. The preconditioners employed in this paper have been implemented as library extensions to the freely available Finite Element ToolKit (FEtk) [39]. (More information about FEtk can be found at http://www.fetk.org/.) The code was compiled using gcc-2.96 with O2 optimization, and all timings were taken on a 2.8GHz P4 Xeon with 4GB of RAM running Linux.

We present experiments with six different unstructured surface meshes (see the images in Figure 7.1) of varying size (see Table 7.1). Note the rapidly deteriorating condition numbers as the number of DOFs increases.

FIG. 7.1. *Models used: Igea face (model* 5*), skull (model* 6*), David face (model* 2*), and David head (models* 1, 3, *and* 4 *with descending order in the number of DOFs).*

TABLE 7.1

Collection of models used in our experiments showing the number of levels used for fast, medium, and slow hierarchies; the maximum and minimum eigenvalues; and the resulting condition number (for harmonic weights).

Model	DOFs	Fast	Med.	Slow	Min eig.	Max eig.	Cond. no.
1-David head	579649	4	8	17	1.22e−5	3.19e+4	2.60e+9
2-David face	223654	4	7	21	1.29e−4	2.26e+3	1.74e+7
3-David head	94220	4	7	22	7.61e−5	2.82e+2	3.71e+6
4-David head	46094	4	5	18	1.44e−4	2.03e+2	1.40e+6
5-Igea face	8038	3	4	16	4.23e−3	2.34e+2	5.53e+4
6-Skull	1203	4	3	2	2.70e−2	3.54e+1	1.31e+3

The goal of the numerical experiments is to determine the minimum solve time as a function of hierarchy and preconditioner used. Another factor concerns the sparse-cutoff, i.e., the level of the hierarchy at which a direct solve (such as SuperLU [14]) is performed. This is typically not the coarsest level the hierarchy creation could produce. Among all possible levels, we numerically choose the coarsest level with respect to the best solve time. The number of levels reported in Table 7.1 is obtained by the sparse-cutoff (in fast hierarchy, we typically find a cutoff after four levels). In our experiments the solver iteration was stopped when the norm of the residual fell below 5.0*e*-5. Vector entries are expressed by `double` datatypes, whereas matrix entries are of `float` type. Interestingly, going from `double` to `float` precision reduced runtimes by only 5% for the largest model, indicating that the runtime is dominated by memory latency, not bandwidth issues.

Figure 7.2 shows all nine combinations of fast, medium, and slow hierarchies with MG, HBMG, and nBPX preconditioners acting on both symmetric (harmonic weights) and nonsymmetric (mean-value weights) systems. For each preconditioner the thick line indicates the winning hierarchy. For example, MG with the fast hierarchy outperforms MG with medium or slow hierarchies. HBMG timings are comparable for fast and medium hierarchies, but asymptotically medium hierarchy timings are favorable (see also the model 1 timings in Table 7.2). HBMG is designed for adaptive meshes, and slow hierarchies are the closest to that pattern. Among the preconditioners used with the slow hierarchy, HBMG turns out to be the most effective, as expected. nBPX favors the fast hierarchy, making the fast hierarchy the best for all

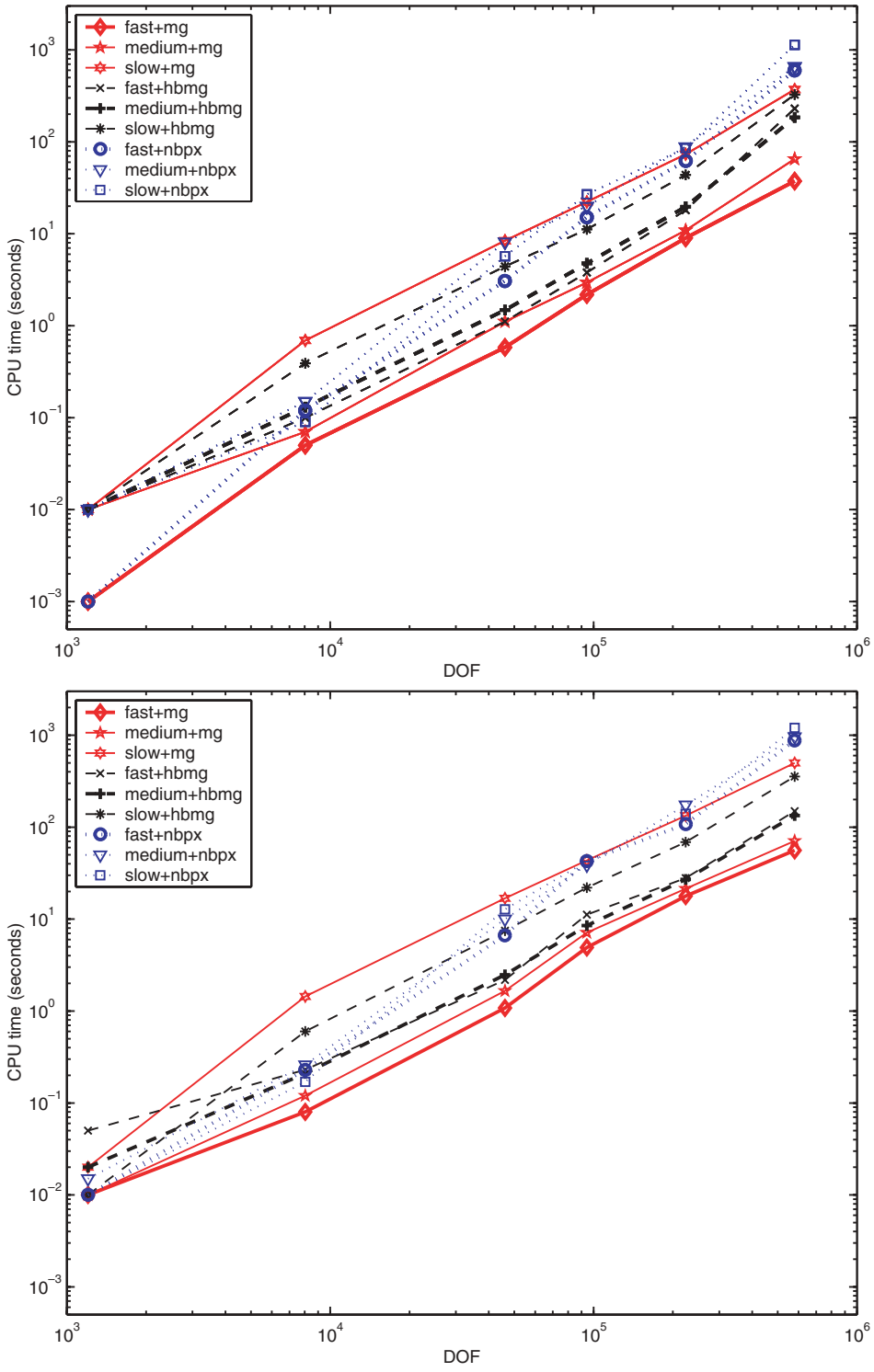

FIG. 7.2. *CPU time for symmetric (top) and nonsymmetric (bottom) systems.*

TABLE 7.2
Solve time in seconds (ascending order in time) and iteration counts for the system with 579,649 *DOFs (model* 1*).*

Hier+pcond	Sym time	Nsym time	Sym iter	Nsym iter
fast+MG	37.3	55.9	19	14
med+MG	65.0	70.7	21	11
med+HBMG	184.0	133.8	100	37
fast+HBMG	230.1	149.2	141	44
slow+HBMG	326.8	356.0	60	35
slow+MG	370.2	500.2	11	6
fast+nBPX	597.9	877.5	1156	869
med+nBPX	655.6	960.8	1171	838
slow+nBPX	1132.7	1202.3	1357	696

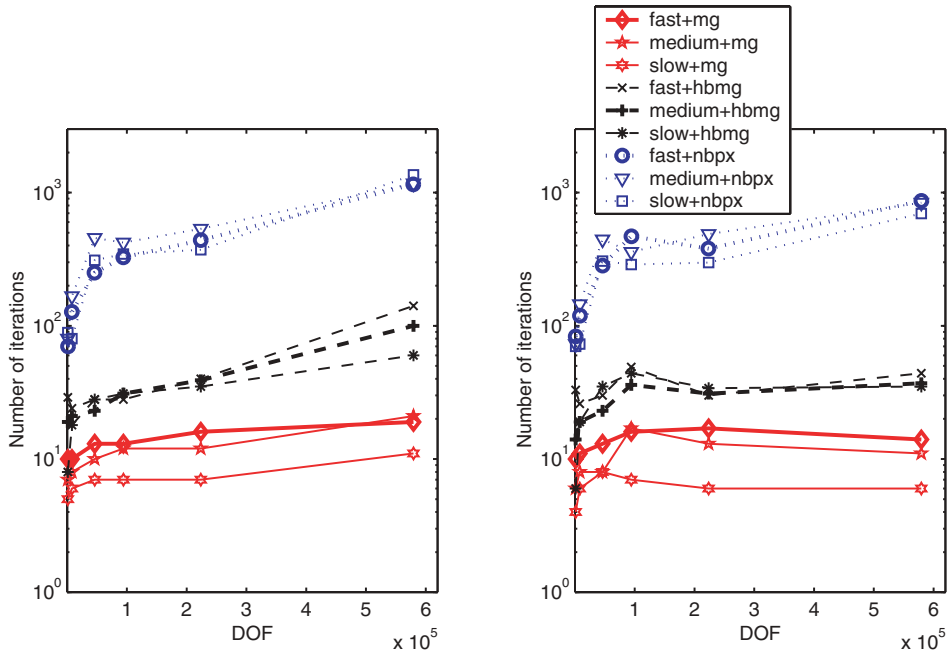

FIG. 7.3. *Iteration counts for symmetric (left) and nonsymmetric (right) systems.*

preconditioners. The least effective preconditioner turns out to be nBPX, especially with the slow hierarchy. Although nBPX is effective on small systems, it rapidly loses its performance on large systems, even though each iteration is very cheap. The main advantage of nBPX—and reason for its use—is its implementation simplicity. All matrix/vector operations can be implemented directly on the mesh datastructure with no need to construct matrices explicitly or to compute triple matrix products. With the fast hierarchy, MG becomes such an effective preconditioner that it outperforms all other preconditioners on all the models. This holds true for both symmetric and nonsymmetric systems.

The geometric decay rate in the number of DOFs produces $O(N)$ solve times for MG and $O(N \log N)$ for HBMG and nBPX, albeit with very different constants. This can be seen in the number of iterations that remain constant for MG, whereas a logarithmically growing count can be observed for HBMG and nBPX (see Figure 7.3).

This is very much the same behavior as observed for elliptic PDEs, which comes as no surprise when the parameterization is obtained by discretizing the LBO. We remark that the iteration counts of nBPX are an order magnitude larger than those for HBMG. This difference is the primary reason why sophisticated preconditioners become superior to the naïve ones even though they require significantly more work per iteration. In order to exhibit the striking performance difference between the preconditioners and the hierarchies used, we insert the results for the largest system in Table 7.2. Notice that iteration counts for the nonsymmetric system are two to three times less than the symmetric one. This is particularly noticeable for the HBMG experiments. The slow hierarchies consistently give rise to the least number of iterations for MG and HBMG preconditioners. However, the depth of the slow hierarchies creates large fill-in, as discussed in section 6. This fill-in makes slow hierarchies the most expensive to use when considering the overall timings.

8. Conclusion. Using different coarsening hierarchies, we presented a systematic performance analysis for MG, HBMG, and nBPX preconditioners on systems arising from the parameterization of unstructured surface meshes. The parameterization schemes of interest use harmonic weights and mean value weights giving symmetric and nonsymmetric systems, respectively. Iteration counts indicate that, for the same preconditioner, a better conditioning of the system is obtained by using more levels, i.e., a slow (DK) hierarchy. But since the cost of one iteration on the slow hierarchy is relatively expensive, fast hierarchies result in the best runtimes.

Our experiments consistently show that the fast hierarchy is favorable over medium and slow hierarchies for all preconditioners, with the best results achieved when using the MG preconditioner. If the slow hierarchy is chosen, the best solve time is achieved by the HBMG preconditioner. We should emphasize that we observed convergence in all the experiments and no restarts for the biconjugate gradient method. The provable guarantee of convergence (for the harmonic weights) is evidenced. In contrast, the (bi)conjugate gradient solver does not converge at all for large models when no preconditioning is used. We conclude that the preconditioners described in this paper are robust with respect to problem size.

There are a number of avenues for interesting future work. For example, a comparison of our work to AMG approaches would be of great interest. A more complete analysis of convergence properties for the mean value weights, which do not arise from the discretization of an elliptic PDE, would also be desirable. Finally, we look forward to applying our solvers to surface parameterization problems involving more than a single disklike region.

Acknowledgments. The authors would like to thank M. Holst for providing FEtk, S. Bond for his help on the components of the preconditioner code, and I. Guskov for parts of the parameterization and coarsening code. We would also like to thank them for many enlightening discussions. The David head model is courtesy of the Digital Michelangelo Project at Stanford University. The Igea and skull models are courtesy of Cyberware, Inc., and Headus, Inc., respectively.

REFERENCES

[1] B. Aksoylu, S. Bond, and M. Holst, *An odyssey into local refinement and multilevel preconditioning* III*: Implementation and numerical experiments*, SIAM J. Sci. Comput., 25 (2003), pp. 478–498.

[2] B. Aksoylu and M. Holst, *An odyssey into local refinement and multilevel preconditioning* II*: Stabilizing hierarchical basis methods*, SIAM J. Numer. Anal., submitted.
[3] R. E. Bank, T. Dupont, and H. Yserentant, *The hierarchical basis multigrid method*, Numer. Math., 52 (1988), pp. 427–458.
[4] R. E. Bank and J. Xu, *An algorithm for coarsening unstructured meshes*, Numer. Math., 73 (1996), pp. 1–36.
[5] D. Baraff and A. Witkin, *Large steps in cloth simulation*, in Proceedings of the SIGGRAPH, 1998, pp. 43–54.
[6] J. H. Bramble, J. E. Pasciak, and J. Xu, *Parallel multilevel preconditioners*, Math. Comp., 55 (1990), pp. 1–22.
[7] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed., SIAM, Philadelphia, 2000.
[8] T. F. Chan, J. Xu, and L. Zikatanov, *An agglomeration multigrid method for unstructured grids*, in Domain Decomposition Methods 10, Contemp. Math, J. Mandel, C. Farhal, and X. C. Cai, eds., AMS, Providence, RI, 218 (1998), pp. 67–81.
[9] U. Clarenz, M. Griebel, M. Rumpf, A. Schweitzer, and A. Telea, *A feature sensitive multiscale editing tool on surfaces*, Visual Computer, 29 (2004), pp. 329–343.
[10] B. Curless and M. Levoy, *A volumetric method for building complex models from range images*, in Proceedings of the SIGGRAPH, 1996, pp. 303–312.
[11] E. de Sturler and J. Liesen, *Block-diagonal and constraint preconditioners for nonsymmetric indefinite linear systems, Part* I: *Theory*, SIAM J. Sci. Comput., to appear.
[12] K. Deckelnick and G. Dziuk, *A fully discrete scheme for weighted mean curvature flow*, Numer. Math., 91 (2003), pp. 423–452.
[13] K. Deckelnick and G. Dziuk, *Mean curvature flow and related topics*, in Frontiers in Numerical Analysis, J. Blowey, A. Craig, and T. Shardlow, eds., Springer-Verlag, Berlin, 2002, 2003, pp. 63–108.
[14] J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li, and J. W. H. Liu, *A supernodal approach to sparse partial pivoting*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 720–755.
[15] M. Desbrun, M. Meyer, and P. Alliez, *Intrinsic parameterizations of surface meshes*, Computer Graphics Forum, 21 (2002), pp. 209–218.
[16] M. Desbrun, M. Meyer, P. Schröder, and A. Barr, *Implicit fairing of irregular meshes using diffusion and curvature flow*, in Proceedings of the SIGGRAPH, 1999, pp. 317–324.
[17] M. Desbrun, M. Meyer, P. Schröder, and A. Barr, *Discrete differential-geometry operators for triangulated* 2*-manifolds*, in VisMath'02, Berlin, Germany, 2002.
[18] T. K. Dey, H. Edelsbrunner, S. Guha, and D. V. Nekhayev, *Topology preserving edge contraction*, Publ. Inst. Math. (Beograd), 66 (1999), pp. 23–45.
[19] D. Dobkin and D. Kirkpatrick, *A linear algorithm for determining the separation of convex polyhedra*, J. Algorithms, 6 (1985), pp. 381–392.
[20] T. Duchamp, A. Certain, T. DeRose, and W. Stuetzle, *Hierarchical Computation of PL Harmonic Embeddings*, manuscript.
[21] G. Dziuk and J. E. Hutchinson, *Finite Element Approximations to Surfaces of Prescribed Variable Mean Curvature*, Preprint 03-01, Fakultät für Mathematik und Physik, Universität Freiburg, Freiburg, Germany, 2003 .
[22] M. Eck, T. D. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, *Multiresolution analysis of arbitrary meshes*, in Proceedings of the SIGGRAPH, 1995, pp. 173–182.
[23] J. Escher, U. F. Mayer, and G. Simonett, *The surface diffusion flow for immersed hypersurfaces*, SIAM J. Math. Anal., 29 (1998), pp. 1419–1433.
[24] M. S. Floater, *Parameterization and smooth approximation of surface triangulations*, Comput. Aided Geom. Design, 14 (1997), pp. 231–250.
[25] M. S. Floater, *One-to-one piecewise linear mappings over triangulations*, Math. Comp., 72 (2003), pp. 685–696.
[26] M. S. Floater, *Mean value coordinates*, Comput. Aided Geom. Design, 20 (2003), pp. 19–27.
[27] M. Garland and P. S. Heckbert, *Surface simplification using quadric error metrics*, in Proceedings of the SIGGRAPH, 1997, pp. 209–216.
[28] T. Gieng, *Unstructured Mesh Coarsening for Multilevel Methods*, Master's thesis, Multi-Res Modeling Group, California Institute of Technology, Pasadena, CA, 2000.
[29] T. S. Gieng, B. Hamann, K. L. Joy, G. L. Schussman, and I. J. Trotts, *Constructing hierarchies for triangle meshes*, IEEE Trans. Visualization Comput. Graphics, 4 (1998), pp. 145–161.
[30] C. Gotsman, X. Gu, and A. Sheffer, *Fundamentals of spherical parameterization for* 3*D meshes*, ACM Trans. Graphics, 22 (2003), pp. 358–363.
[31] E. Grinspun, P. Krysl, and P. Schröder, *CHARMS: A simple framework for adaptive*

simulation, ACM Trans. Graphics, 21 (2002), pp. 281–290.
[32] X. Gu, S. J. Gortler, and H. Hoppe, *Geometry images*, ACM Trans. Graphics, 21 (2002), pp. 355–361.
[33] X. Gu and S.-T. Yau, *Global conformal surface parameterization*, in Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, 2003, pp. 127–137.
[34] I. Guskov, A. Khodakovsky, P. Schröder, and W. Sweldens, *Hybrid meshes: Multiresolution using regular and irregular refinement*, in Proceedings of the Eighteenth Annual Symposium on Computational Geometry, 2002, pp. 264–272.
[35] I. Guskov, W. Sweldens, and P. Schröder, *Multiresolution signal processing for meshes*, in Proceedings of the SIGGRAPH, 1999, pp. 325–334.
[36] I. Guskov, K. Vidimče, W. Sweldens, and P. Schröder, *Normal meshes*, in Proceedings of the SIGGRAPH, 2000, pp. 95–102.
[37] W. Hackbusch, *Multigrid Methods and Applications*, Springer-Verlag, Berlin, 1985.
[38] S. Haker, S. Angenent, A. Tannenbaum, R. Kikinis, G. Sapiro, and M. Halle, *Conformal surfaces parameterization for texture mapping*, IEEE Trans. Vizualization Comput. Graphics, 6 (2000), pp. 181–189.
[39] M. Holst, *Adaptive numerical treatment of elliptic systems on manifolds*, Adv. Comput. Math., 15 (2001), pp. 139–191.
[40] H. Hoppe, *Progressive meshes*, in Proceedings of the SIGGRAPH, 1996, pp. 99–108.
[41] H. Hoppe, *Smooth view-dependent level-of-detail control and application to terrain rendering*, in IEEE Visualization, 1998, pp. 35–42.
[42] K. Hormann and G. Greiner, *MIPS: An efficient global parametrization method*, in Curve and Surface Design: Saint-Malo 1999, Vanderbilt University Press, Nashville, TN, 2000, pp. 153–162.
[43] J. E. Jones and P. S. Vassilevski, *AMGe based on element agglomeration*, SIAM J. Sci. Comput., 23 (2001), pp. 109–133.
[44] A. Khodakovsky, N. Litke, and P. Schröder, *Globally smooth parameterizations with low distortion*, ACM Trans. Graphics, 22 (2003), p. 350–357.
[45] D. G. Kirkpatrick, *Optimal search in planar subdivisions*, SIAM J. Comput., 12 (1983), pp. 28–35.
[46] L. Kobbelt, S. Campagna, and H.-P. Seidel, *A General framework for mesh decimation*, in Proceedings the of Graphics Interface Conference, 1998, pp. 43–50.
[47] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel, *Interactive multi-resolution modeling on arbitrary meshes*, in Proceedings of the SIGGRAPH, 1998, pp. 105–114.
[48] L. P. Kobbelt, J. Vorsatz, U. Labsik, and H.-P. Seidel, *A shrink wrapping approach to remeshing polygonal surfaces*, Computer Graphics Forum, 18 (1999), pp. 119–130.
[49] V. Krishnamurthy and M. Levoy, *Fitting smooth surfaces to dense polygon meshes*, in Proceedings of the SIGGRAPH, 1996, pp. 313–324.
[50] A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin, *MAPS: Multiresolution adaptive parameterization of surfaces*, in Proceedings of the SIGGRAPH, 1998, pp. 95–104.
[51] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk, *The digital Michelangelo project: 3D scanning of large statues*, in Proceedings of the SIGGRAPH, 2000, pp. 131–144.
[52] B. Lévy, S. Petitjean, N. Ray, and J. Maillot, *Least squares conformal maps for automatic texture atlas generation*, ACM Trans. Graphics, 21 (2002), pp. 362–371.
[53] J. Liesen, E. de Sturler, A. Sheffer, Y. Aydin, and C. Siefert, *Preconditioners for indefinite linear systems arising in surface parameterization*, in Proceedings of the 10th International Meshing Round Table, 2001, pp. 71–82.
[54] W. E. Lorensen and H. E. Cline, *Marching cubes: A high resolution 3D surface construction algorithm*, Computer Graphics, 21 (1987), pp. 163–169.
[55] J. Maillot, H. Yahia, and A. Verroust, *Interactive texture mapping*, in Proceedings of the SIGGRAPH, 1993, pp. 27–34.
[56] U. F. Mayer, *A numerical scheme for moving boundary problems that are gradient flows for the area functional*, European J. Appl. Math., 11 (2000), pp. 61–80.
[57] U. F. Mayer, *Numerical solutions for the surface diffusion flow in three spatial dimensions*, Comput. Appl. Math., 20 (2001), pp. 361–379.
[58] U. F. Mayer and G. Simonett, *A numerical scheme for axisymmetric solutions of curvature driven free boundary problems, with applications to the Willmore flow*, Interfaces Free Bound., 4 (2002), pp. 89–109.
[59] G. L. Miller, D. Talmor, and S.-H. Teng, *Optimal coarsening of unstructured meshes*, J. Algorithms, 31 (1999), pp. 29–65.

[60] E. Morano, D. J. Mavriplis, and V. Venkatakrishnan, *Coarsening strategies for unstructured multigrid techniques with application to anisotropic problems*, SIAM J. Sci. Comput., 20 (1998), pp. 393–415.
[61] J. C. C. Nitsche, *Boundary value problems for variational integrals involving surface curvatures*, Quart. Appl. Math., 51 (1993), pp. 363–387.
[62] S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, Appl. Math. Sci. 153, Springer-Verlag, New York, 2003.
[63] U. Pinkall and K. Polthier, *Computing discrete minimal surfaces*, Experiment. Math., 2 (1993), pp. 15–36.
[64] E. Praun and H. Hoppe, *Spherical parameterization and remeshing*, ACM Trans. Graphics, 22 (2003).
[65] M. C. Rivara, *Algorithms for refining triangular grids suitable for adaptive and multigrid techniques*, Internat. J. Numer. Methods Engrg., 20 (1984), pp. 745–756.
[66] M. C. Rivara, *Local modification of meshes for adaptive and/or multigrid finite element methods*, J. Comput. Appl. Math., 36 (1991), pp. 79–89.
[67] J. W. Ruge and K. Stüben, *Algebraic multigrid*, in Multigrid Methods, S. F. McCormick, ed., Frontiers Appl. Math. 3, SIAM, Philadelphia, 1987, pp. 73–130.
[68] P. Sander, S. Gortler, J. Snyder, and H. Hoppe, *Signal-specialized parameterization*, in Eurographics Workshop on Rendering, 2002, pp. 87–100.
[69] P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe, *Texture mapping progressive meshes*, in Proceedings of the SIGGRAPH, 2001, pp. 409–416.
[70] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen, *Decimation of triangle meshes*, Computer Graphics, 26 (1992), pp. 65–70.
[71] A. Sheffer and E. de Sturler, *Surface parameterization for meshing by triangulation flattening*, in Proceedings of the 9th International Meshing Roundtable, 2000, pp. 161–172.
[72] J. R. Shewchuk, *What is a good linear element? Interpolation, conditioning, and quality measures*, in Proceedings of the 11th International Meshing Roundtable, 2002, pp. 115–126.
[73] O. Sorkine, D. Cohen-Or, R. Goldenthal, and D. Lischinski, *Bounded-distortion piecewise mesh parameterization*, in IEEE Visualization Conference, 2002, pp. 355–362.
[74] W. Sweldens, *The lifting scheme: A construction of second generation wavelets*, SIAM J. Math. Anal., 29 (1997), pp. 511–546.
[75] W. Sweldens and P. Schröder, *Building your own wavelets at home*, in Wavelets in Computer Graphics, Course Notes, ACM Proceedings of the SIGGRAPH, 1996, pp. 15–87.
[76] W. T. Tutte, *How to draw a graph*, Proc. London Math. Soc., 13 (1963), pp. 743–767.
[77] P. Vanek, J. Mandel, and M. Brezina, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, Computing, 56 (1996), pp. 179–196.
[78] W. L. Wan, T. F. Chan, and B. Smith, *An Energy-Minimizing Interpolation for Robust Multigrid*, Tech. report, Department of Mathematics, UCLA, Los Angeles,1998.
[79] Z. Wood, M. Desbrun, P. Schröder, and D. Breen, *Semi-regular mesh extraction from volumes*, in IEEE Visualization Conference, 2000, pp. 275–282.
[80] J. C. Xia and A. Varshney, *Dynamic view-dependent simplification for polygonal models*, in IEEE Visualization Conference, 1996, pp. 327–334.
[81] G. Xu, *Convergent discrete Laplace–Beltrami Operators over Triangular Surfaces*, Tech. report, ICMSEC, China, 2003.
[82] G. Xu, Q. Pan, and C. Bajaj, *Discrete Surface Modeling Using Geometric Flows*, Tech. report, Dept. of Computer Sciences, University of Texas, Austin, 2003.
[83] J. Xu and J. Qin, *Some remarks on a multigrid preconditioner*, SIAM J. Sci. Comput., 15 (1994), pp. 172–184.
[84] H. Yserentant, *On the multilevel splitting of finite element spaces*, Numer. Math., 49 (1986), pp. 379–412.