Preprint 0 (2001) ?–?

Recovering Mesh Geometry from a Stiffness Matrix

Andreas Stathopoulos

Department of Computer Science, College of William and Mary, Williamsburg, Virginia
E-mail: andreas@cs.wm.edu

Shang-Hua Teng

Department of Computer Science, University of Illinois at Urbana-Champaign, Champaign, Illinois,
E-mail: steng@cs.uiuc.edu

We introduce the following class of mesh recovery problems: Given a stiffness matrix A and a PDE, construct a mesh M such that the finite-element formulation of the PDE over M is A. We show, under certain assumptions, that it is possible to reconstruct the original mesh for the special case of the Laplace operator discretized on an unstructured mesh of triangular elements with linear basis functions. The reconstruction is achieved through a series of techniques from graph theory and numerical analysis, some of which are new and can find application in other scientific areas. Finally, we discuss extensions to other operators and some open questions related to this class of problems.

Keywords: Stiffness matrix, mesh, geometry, triangulation, angles, graph embedding, graph drawing, non linear solvers, elliptic partial differential equations

AMS Subject classification: Primary 65D18, 65H10; Secondary 65U05, 65N30

1. Introduction

A problem that has not been studied in the literature is the recovery of the mesh geometry from which a given stiffness matrix originates. In this formulation, this is an ill-posed, inverse problem, in the sense that there are many partial differential equations (PDEs) whose discretization on different embeddings of a topologically equivalent mesh yield the same stiffness matrix. An also interesting problem is the recovery of a mesh for a stiffness matrix, when the underlying PDE is given. Although still ill-posed, this problem provides more information that can be exploited for a mesh recovery.

Beyond the intellectual merit of the problem, there are several potential applications. For example, one may want to reverse engineer the actual model from a matrix, either because the mesh was discarded in an earlier phase of the simulation, or simply because it was never commonly available. However, a deeper motivation for this work is to provide useful relationships between graph theory and scientific computing.

For many problems in scientific computing, we can design efficient numerical algorithms, if we know their geometric structure. One example is the geometric multigrid method [1]. If only the stiffness matrix is known, traditional Algebraic Multigrid (AMG) [7] uses the information present in the matrix to define sophisticated intergrid transfer operators. However, if the operator and the mesh that generated the matrix are known, geometric or classical multigrid usually outperform AMG. Being able to recover the mesh from the matrix could provide new understanding of how the AMG works and it could suggest ways of enhancement. Another example is the geometric technique by Miller *et al* [6], which finds a balanced partition of a well-shaped mesh in both two and three dimensions. Both the running time and quality of their partitioning reflect the efficient use of geometric information. In [12], Spielman and Teng showed that the geometric structure of a well-shaped mesh can be used to bound and approximate eigenvalues of the discrete Laplacian matrix of the mesh. Finally, there is a host of geometric based techniques developed for the direct solution of sparse linear systems (in elimination ordering), as well as for iterative solution methods.

A second, dual motivation is that we can design efficient algorithms for graphs, if we know their numerical structure. For example, the problem we consider in this paper is closely related to problems in graph embedding and graph drawing. Applications abound in diverse fields such as circuit design, networks, human-computer interface and domain triangulations. While traditional graph embedding and drawing algorithms use only the topological information of a graph, numerical information in the stiffness matrix should help us find better quality graph embeddings. A related problem is the parameterization of triangulated surfaces, which is used for surface mesh generation and texture mapping in computer graphics. Recent approaches have been based on an angle based procedure that flattens the 3-D surface [5,10]. This procedure shares many common elements with the angle embedding algorithms discussed in this paper. Finally, difficult combinatorial problems in graph theory have elegant approximate solutions if a geometry is known.

In this paper, we take the first step in establishing this connection between the geometric structure of a matrix and numerical structure of a mesh. We consider the problem of recovering the mesh geometry for the stiffness matrix of the Laplacian operator. We show that this is possible through the solution of a non-linear system of equations, and we describe techniques to geometrically embed the solution in a stable way.

The paper is organized as follows. In Section 2 we relate the entries in the stiffness matrix with the angles of the mesh, deriving a non-linear system of equations. We also show how to determine the elements and the boundary of the corresponding mesh. Reducing the non-linear problem on the boundary nodes is thus possible in principle, but it turns out to be computationally non competitive. In Section 3 we present two numerical techniques for solving the angle equations and we give two practical approaches for finding a good initial guess. In Section 4 we give numerical results on the convergence and accuracy of the techniques developed in Section 3. In Section 5 we present several algorithms for computing the geometric embedding of a mesh from its angle sequence. Some experimental results demonstrate the improved stability of the embedding algorithms. In Section 6 we extend our techniques to the scaled and shifted Laplacian as well as to anisotropic elliptic operators with constant coefficients. In Section 7 we conclude the paper and outline some future research directions.

2. Recovering Geometry for the Discretized Laplace Operator

The Laplace operator usually provides an appropriate starting point for testing new ideas and algorithms in scientific computing. Beyond its importance in both applied mathematics and physics applications, the Laplace operator demonstrates also rich geometric structure which facilitates the methods presented in this paper. Specifically, the stiffness matrix obtained from the discretization of the Laplacian on any finite element mesh, using linear basis functions, is invariant under translation, rotation, and scaling of the mesh. Also, the entries of the stiffness matrix are associated through simple formulas to the angles in the triangulation. As a result, we can determine the geometry of the mesh by computing all the angles formed by the triangular elements. Below we describe the system of equations from which these angles are computed.

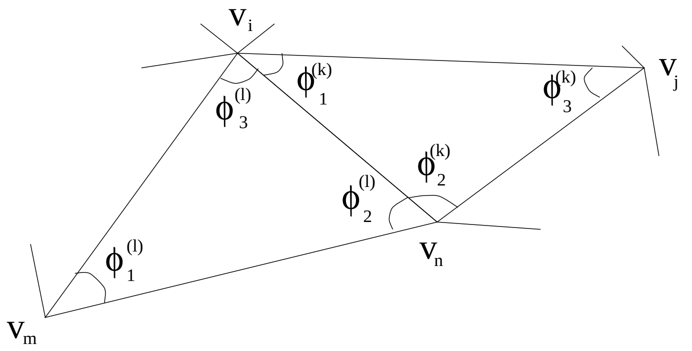

Figure 1. Two neighboring elements of a mesh.

2.1. From Stiffness Matrix to Angle Equations

Considering a triangulation of a domain, we denote the three angles formed by the $l-th$ element (triangle) $(\phi_1^{(l)}, \phi_2^{(l)}, \phi_3^{(l)})$, as shown in Figure 1. In total, there are $3m$ angles to be computed, where m is the number of triangles in the mesh. There are two possible cases; an angle can face a boundary edge, i.e., an edge that no other triangle includes, or it can face an interior edge, i.e., an edge shared by exactly two triangles. When the Laplacian operator is discretized with linear finite elements, and using Neumann boundary conditions, it is well known that the angles in the triangulation and the entries of the stiffness matrix, A, satisfy the following properties [4]:

If two angles $\phi_1^{(l)}, \phi_3^{(k)}$ face the same side (v_i, v_n),

$$\cot \phi_1^{(l)} + \cot \phi_3^{(k)} = -2A_{v_i,v_n}. \tag{1}$$

If an angle $\phi_3^{(l)}$ faces a boundary side (v_m, v_n),

$$\cot \phi_3^{(l)} = -2A_{v_m,v_n}. \tag{2}$$

There is a total of e equations from (1) and (2), where e is the number of edges in the graph. It is also interesting to note that the diagonal entries of the stiffness matrix are equal to the negative sum of the off-diagonal ones.

The above equations relate the properties of the Laplacian on the elements, but they do not enforce the fact that the angles come from a triangulation. The following m equations ensure the triangle property:

$$\phi_1^{(l)} + \phi_2^{(l)} + \phi_3^{(l)} = \pi, \quad l = 1, \ldots, m. \tag{3}$$

Finally for each interior vertex, i.e., a vertex that does not participate in a boundary edge, the sum of all the angles around it should be 2π. This ensures that all the points of the triangulation lie on the same plane:

$$\sum_{v_i \in \text{ element } (l),\ \phi_k^{(l)} \text{ angle of } v_i} \phi_k^{(l)} = 2\pi. \tag{4}$$

Therefore, if we know the vertices that form each element, as well as the boundary edges, we can set up a system involving the above equations in order to solve for the angles of the triangulation. This planarity information can be obtained by the following graph theoretic techniques.

2.2. Element and Boundary Detection

We assume that the stiffness matrix corresponds to a planar triangular mesh, i.e., each element of the mesh is a triangle. Since the stiffness matrix should reveal the connectivity of the mesh, we assume that there is always a non zero matrix entry wherever the corresponding edge exists. The case where an edge is faced by one or two $\pi/2$ angles, thus creating a zero element in the matrix, is more complicated and not examined in this paper. To ensure a unique planar embedding of the elements, we assume that the outside boundary and all holes in the domain have at least four edges. In other words, we assume that the mesh does not have triangle holes or engulfing boundary. In addition, we assume that the mesh is a tri-connected planar graph.

Given a stiffness matrix A, we can apply the planarity testing algorithm of Hopcroft and Tarjan [11] to determine all elements. The output of the planarity testing algorithm on A is an embedding of the graph G of A. It specifies for each vertex v of G the set of edges incident to v in a counter-clock wise ordering. It also specifies each face of G in a counter-clock wise ordering. Because of our assumption above, every triangular face is an element of the mesh.

If G has only one face f that contains more than three edges, then f is the boundary of the mesh of A. In general, G could have more than one non-triangular face, say $f_1, \ldots, f_h$. Only one of them can be the boundary. All other faces correspond to the holes of the mesh. For the purpose of computing the angles, however, equations (1)–(4) do not distinguish between external boundaries and holes. Even though, in a graph theoretic sense, boundaries and holes are equivalent, the geometry of the original mesh can be recovered after the angles are computed.

2.3. Number of Equations

Using the above techniques we can set up the system of non linear equations (1)–(4) that describes the angles. The question arises whether the derived Jacobian system is overdetermined or underdetermined. In this subsection, we compare the number of unknowns and the number of equations and show that the number of equations is always greater than or equal to the number of unknowns.

Suppose the mesh M of A has v vertices, e edges, m triangular elements, and h holes, with at least one non-triangular boundary or hole. Because each triangle has three angles to determine, we have $3m$ unknowns. Let f be the number of faces of a planar embedding of M, i.e., $f = m + h + 1$. Suppose in addition, that M has w interior vertices. Then we have $e + m + w$ equations: e equations from (1)–(2); m equations from (3); and w equations from (4).

Theorem 1. For any planar mesh M,

$$e + m + w \geq 3m.$$

Proof. Equivalently, we need to show $e - 2m + w \geq 0$. Using Euler's formula $f - e + v = 2$ and the fact that $m = f - h - 1$, we have:

$$\begin{aligned} e + w - 2m &= (f + v - 2) - 2(f - h - 1) + w \\ &= (v - f + w) + 2h \end{aligned}$$

We first prove by an induction on the number w of interior vertices, that we can reduce the theorem to the case where there are no interior vertices. We then prove that the theorem is true for this reduced case.

If $w = 0$, then M has no interior vertex. Now inductively assume the theorem is true for $w = w_0$. We now prove it for $w = w_0 + 1$.

Let u be an interior vertex in M. Let $u_1, \ldots, u_d$ be the neighbors of u in M given in a counter-clockwise ordering. Notice that $u_1, \ldots, u_d$ form a simple polygon. Let G' be a planar graph obtained from G by removing u and by triangulating the simple polygon formed by $u_1, \ldots, u_d$. Notice that G' has $v - 1$ vertices, w_0 interior vertices, and $f - 2$ faces. There is no change in the number of holes. By the inductive assumption, we have $(v-1)-(f-2)+w_0+2h \geq 0$, implying $v-f+(w_0+1)+2h = v-f+w+2h \geq 0$.

We now prove the theorem for the base case, when the mesh does not contain any interior vertices. Again we use induction, this time, based on the number of faces f in the mesh. If $f=2$, then the theorem is clearly true. Suppose the theorem is true for f_0 and we have a planar graph that has $f=f_0+1$ faces. There are two cases: (1) the graph contains a triangular element; (2) the graph does not contain any triangular element. In the first case, because the graph does not have an interior vertex, there must be a triangular element $T=(uzw)$ such that one of its edges, say (uz) is an edge of a non-triangular face D. We can obtain another planar graph G' be applying edge contraction to (uz). This contraction merges u and z into a new vertex and removes the face T from the graph. Because D is not triangular, the contraction simply modifies D. Therefore, G' has one less vertex and one less face than G. Hence by induction, if the theorem is true for G', it is true for G as well. In the second case, we have $4f \leq 2e$, i.e., $e \geq 2f$. By Euler's formula, we have $v = e-f+2 \geq 2f-f+2 \geq f+2$. Hence $v-f \geq 0$, proving the theorem. □

2.4. Reducing the Problem onto the Boundary

It is well known that, for the Laplace's equation, the x and y coordinate vectors for any isomorphic realization of the mesh are themselves solutions of equations with the stiffness matrix A and appropriate boundary conditions. More specifically, considering a symmetric permutation of A with interior unknowns ordered first, and the corresponding permutations of the coordinate vectors, $x = [x_I x_B]^T, y = [y_I y_B]^T$, then:

$$\begin{array}{r} \text{Interior unknowns}\, \{ \\ \text{Boundary unknowns}\, \{ \end{array} \begin{bmatrix} A_I & A_{IB} \\ A_{BI} & A_B \end{bmatrix} \begin{bmatrix} x_I & y_I \\ x_B & y_B \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ b_x & b_y \end{bmatrix}. \tag{5}$$

Following the notation of the previous section, $\dim(A_I) = w \times w$, $\dim(A_B) = n_B \times n_B$ with $n_B = v-w$, and b_x and b_y are vectors of $\Re^{n_B}$. Therefore, if the coordinates of the boundary nodes are known, we can derive all interior ones by solving the following:

$$x_I = -A_I^{-1} A_{IB} x_B. \tag{6}$$

Although not guaranteed in the worst case, the number of boundary unknowns n_B is expected to be on the order of $\mathcal{O}(\sqrt{v})$, and thus it makes sense to set up

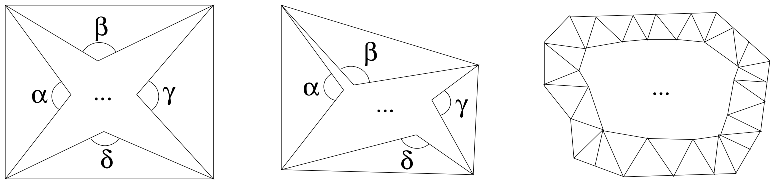

Figure 2. Boundary edges/angles do not define the body rigidly. The boundary tube is needed.

the non linear equations only for the angles required to obtain x_B and y_B. From equation (2), we can immediately obtain all the angles facing boundary sides. However, as shown in Figure 2, to define the mesh rigidly more information is needed. We introduce the notion of the tube as the set of all elements with at least one boundary vertex. The problem is now reduced into finding a geometrical embedding of the tube.

Unfortunately, a problem arises in case of holes. The graph theoretical methods described earlier identify the holes topologically, but provide no information as to where they should be embedded in relation to the outer tube. Even worse, there is no notion of outer tube because of the topological equivalence of boundaries. Unlike this reduced technique, working with all the angles in the mesh obviates this problem. Even in the absence of holes, and despite its smaller size, the reduced problem has proven harder to solve computationally. In section 6 we provide a brief discussion on the reasons for this behavior.

3. Numerical Computation of the Angles

The non linear system of equations (1)–(4) is usually solved by some form of the Newton method. The following sections discuss the computational considerations of setting up the Jacobian of this system and solving it efficiently.

3.1. Setting up the System

The first decision is the choice of unknowns. Equations (1)–(2) depend only linearly on angles, and equations (3)–(4) depend only linearly on cotangents. In solving the system we would like to keep as much linearity as possible. When few interior points exist, as in meshes with long boundaries or holes, the cotangents can be considered the unknowns, and the partial derivatives for the angles in the

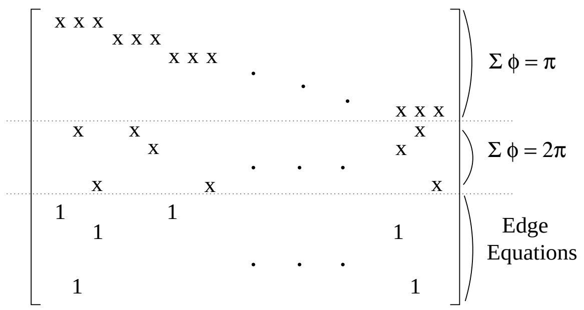

Figure 3. The sparsity structure of the Jacobian.

Jacobian become:

$$\frac{\partial \phi_i^{(l)}}{\partial \cot \phi_i^{(l)}} = \frac{-1}{1 + \cot^2 \phi_i^{(l)}}.$$

We can also consider the angles as unknowns and formulate the cotangent part of the Jacobian as follows:

$$\frac{\partial \cot \phi_i^{(l)}}{\partial \phi_i^{(l)}} = \frac{-1}{\sin^2 \phi_i^{(l)}}.$$

In both cases, the Jacobian is sparse and it has a structure that can be exploited computationally. The unknown angles are ordered by element, with angles in the same element being successive in the ordering. The first m rows contain the equations for each triangle. The following w rows contain the equations for interior vertices, and the last e rows are the equations (1)–(2). Every unknown participates in one edge equation, in one triangle equation and at most in one interior vertex equation. Therefore, the number of nonzeros in the Jacobian is at most $9m$.

If an iterative method is used to solve the Jacobian system, since a QR decomposition would generate prohibitively large amounts of fill-in, the above storage can be reduced approximately in half. Considering the cotangents as the unknowns, the Jacobian part of the last e equations contains only '1's in known positions which do not have to be stored. Similarly, when the angles are the unknowns, the first $m + w$ equations need not be stored. For reasons explained in section 3.2, we have opted to keep the cotangents as unknowns for stability reasons. In this case, the sparsity structure of the Jacobian is shown in Figure 3.

3.2. A Newton Iteration

The e rows of the Jacobian as stemming from equations (1)–(2) are linearly independent for any choice of angles or unknowns. This follows immediately from the fact each angle participates in only one equation in this group. A similar property holds for the rows stemming from the group of equations (3)–(4). Their linear independence follows from two observations: First, each angle appears only in one triangle, and therefore all equations (3) are linearly independent. Second, the equations (4) are needed to ensure that every interior point is on the same plane as its neighbors, and therefore they cannot be reproduced as a combination of the triangle equations. Equations (3)–(4) become linearly independent however, if the Jacobian of a system that represents an exact planar embedding is considered.

Despite the linear independence of the individual subblocks, and the overdetermined whole system, at solution, the Jacobian is rank deficient. This is because at solution, all points are forced onto the same plane and the conditions of interior points can be obtained from the triangle conditions and the solution of the edge equations.

However, if the angles are not converged and the approximate solution does not represent an exact planar embedding, the Jacobian is full rank. This suggests that a Newton method for this non-linear system has r-quadratic convergence [3]. Let us denote by $F(x) = 0$ the non linear equations (1)–(4), and the Jacobian at some approximate iterate x_i by $J_i \equiv J(x_i)$. We consider the Newton iteration, that solves the overdetermined Jacobian system in a least squares sense:

$$x^{i+1} = x^i - (J_i^T J_i)^{-1} J_i^T \; F(x^i). \tag{7}$$

If the angles are chosen as unknowns, we should ensure that the Newton correction to the approximate solution x^i does not violate the triangle property of the angles:

$$0 < \phi_i^{(l)} < \pi. \tag{8}$$

To avoid this constrained minimization problem formulation, we have chosen the cotangents as unknowns since they are unbounded periodic functions. In this way, we only need to enforce condition (8) when we compute the angles from the cotangents for calculating the residual $F(x)$.

Our experiments confirmed the r-quadratic convergence of this iteration for a few small meshes. A slight increase in the size or the complexity of the mesh however, causes the method not to converge. As it is typical in many areas in scientific computing, the high dimensionality of the problem shrinks the convergence region of the Newton method to a small neighborhood around the solution. We seek therefore an efficient, global converging iteration.

3.3. An Inner-Outer BiCG-Newton

When global convergence problems plague non linear iterations, a well known remedy is to solve each Jacobian system in (7) approximately with some iterative method [3,2]. Krylov subspace methods, such as GMRES and BiCG [9], are common choices for iterative methods. At each step of the outer Newton iteration, a varying number of BiCG iterations is taken on the Jacobian system. In early Newton steps, when the approximation is far from the solution, the system should not be solved very accurately, and thus a few steps of BiCG suffice. As the approximation improves, higher accuracy is demanded from the BiCG and the number of iterations increases.

The efficiency of this inner-outer scheme is well known, provided that we know how to vary the number of iterations for BiCG. Among the many ideas proposed in the literature, one by Dembo *et al.* has been quite successful [2]. According to this, instead of controlling the number of BiCG iterations explicitly, we start with a large residual threshold, decreasing it at every outer (Newton) step. Dembo *et al.* proposed that the value 2^{-i} be used as the stopping criterion for the inner iteration, where i is the number of outer iterations. Choosing this criterion is widely recognized as a hard and problem dependent task. Because of the limited convergence region of our problems, a residual threshold of about 1.1^{-i} is more beneficial. As we show later in our experimental results, this method is globally convergent and can be used for large size meshes.

The main computational requirement for this method is that we can perform a matrix-vector multiplication with the sparse Jacobian and a matrix-vector multiplication with its transpose. Since each angle appears in at most three different rows, the sparse structure of the transpose can be maintained easily.

In some problems, the computational efficiency of this inner-outer method could be improved by considering a full rank Jacobian matrix, instead of the overdetermined one used in (7). As we discussed earlier, this can achieved by picking only that number of interior vertex equations needed to obtain a square matrix. Then, the BiCG could be applied simply on a square rather than on an overdetermined system, obviating the use of the transpose of the Jacobian. Besides being more efficient computationally, the condition of this system is not squared and we expect BiCG to converge faster. The disadvantage of this approach is that the square matrix is not symmetric, and it cannot be guaranteed to be full rank for all approximate iterates.

3.4. The Initial Guess

For all the above methods, and especially for the Newton iteration, the choice of initial guess is critical to the convergence of the method. A simple and quite effective initial guess is to assign $\pi/3$ to each angle. By this assignment, all equations (3) are automatically satisfied. Another reason to use $\pi/3$ as an initial guess for angles is that most finite element meshes are well-shaped, which means that the smallest angle of each element is reasonably large.

The problem with the $(\pi/3)$-initial guess is that, usually, it is not a feasible assignment for a planar embedding of the mesh. For example, the 2π condition around interior vertices is often violated. To ensure that the initial guess gives a feasible planar embedding, we can use the *barycentric embedding.*

To generate a barycentric embedding of a planar graph, we first need to determine: (1) a boundary face; and (2) an embedding of this face. Suppose F is the chosen boundary face. We can map F in a counter-clockwise order to a convex polygon with $|F|$ sides, where $|F|$ is the number of vertices in F. In our approach, we map F to a unit regular $|F|$-gon. To determine the position of an interior vertex u, we use the following barycentric equation. Let $u_1, \ldots, u_k$ be the neighbors of u, and (x_v, y_v) be the x-y coordinates of a vertex v. For all u we have:

$$x_u = (x_{u_1} + \ldots + x_{u_k})/k$$
$$y_u = (y_{u_1} + \ldots + y_{u_k})/k.$$

As shown by Tutte [13,14], these two linear systems always have a solution. Moreover, the resulting embedding is planar. An extension to this idea is to use a weighted barycentric embedding. In the above equations, we weigh the coordinates of each neighbor u_i by its algebraic distance from vertex u, i.e., the weight of $|A_{u,u_i}|$. The equations then become:

$$x_u = (|A_{u,u_1}|x_{u_1} + \ldots + |A_{u,u_k}|x_{u_k}) / \sum_i |A_{u,u_i}|$$

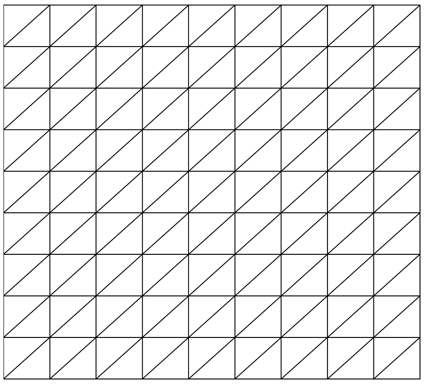

Figure 4. A 10×10 uniform square mesh with 261 edges, 162 elements, 64 interior points, and a Jacobian system of size 487×487.

$$y_u = (|A_{u,u_1}|y_{u_1} + \ldots + |A_{u,u_k}|y_{u_k}) / \sum_i |A_{u,u_i}|.$$

It is easy to show that the weighted equations have similar properties and their embedding is also planar. Because of the nature of the Laplacian operator we expect the weighted embedding to capture in a better way the relative distances in the original mesh. We can use the angles from any of these barycentric embeddings as our initial guess.

4. Numerical Experiments

To test the techniques described in this paper, we discretized the Laplacian using Neumann boundary conditions on a variety of meshes. We show convergence results from three sample meshes; one uniform square mesh, and two meshes generated by SPARSKIT [8], shown in Figures 4, 5, and 6 respectively. In the experiments, we assign $\pi/3$ as the initial guess to all angles, and we use BiCG to solve the system in iteration (7), either accurately when applying Newton's method, or approximately using the inner-outer scheme. Except for the generation of the two meshes by SPARSKIT, all other software has been written in MATLAB.

Our experiments are in agreement with the earlier discussion in this paper. The Newton method converges very fast but only for small meshes, or more precisely, when the dimensionality of the problem is not too large. As figure 7 shows, the dotted curve, depicting the convergence history of the Newton iteration, converges in four steps for the uniform grid problem. To solve the four systems, BiCG

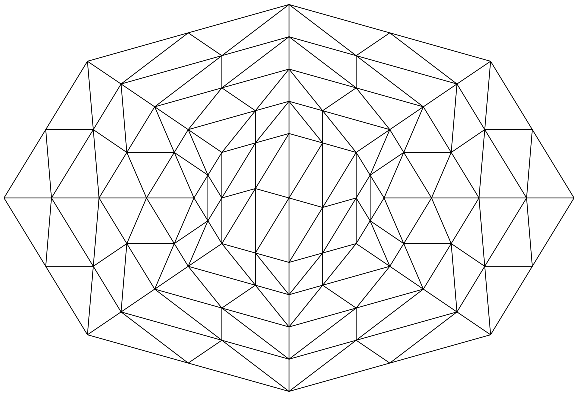

Figure 5. A SPARSKIT mesh with 85 vertices, 236 edges, 152 elements, 69 interior points, and a Jacobian system of size 457×457.

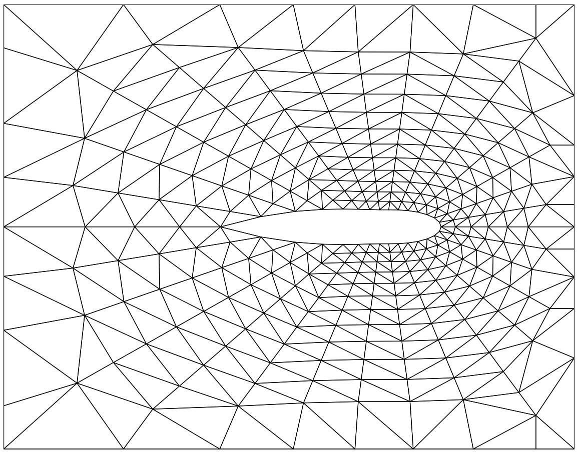

Figure 6. A SPARSKIT mesh with 330 vertices, 926 edges, 596 elements, 266 interior points, and a Jacobian system of size 1788×1789.

took a total of 350 iterations. Note that each iteration involves two matrix-vector multiplications; one with the Jacobian and one with its transpose.

The Newton iteration failed to make any progress on either of the two SPARSKIT meshes. Notice that the smaller dimension of the mesh in figure 5 does not help the Newton iteration to converge. The reason is that on the uniform mesh, all triangles have exactly the same angles and so the real dimensionality of the problem is small. The multitude of shapes in the unstructured SPARSKIT meshes sets many more hurdles to the Newton method.

On the other hand, as figures 7 and 8 show, the BiCG-Newton method

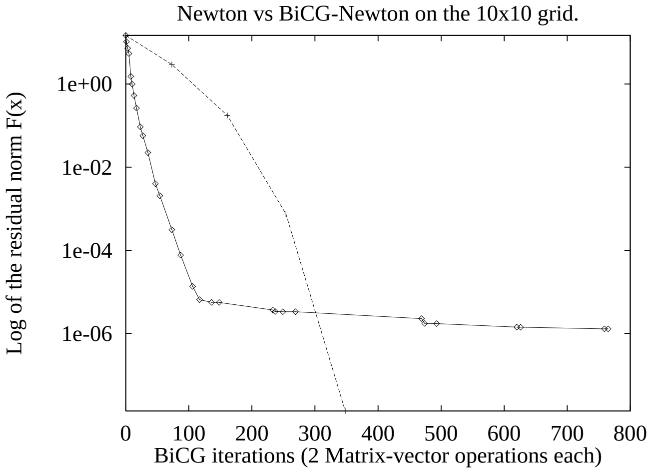

Figure 7. History of residual norm convergence of the Newton and BiCG-Newton methods on the 10×10 uniform grid. The dotted line represents the convergence of the Newton iteration. The points mark the non-linear iterations, while the x axis shows the number of BiCG iterations.

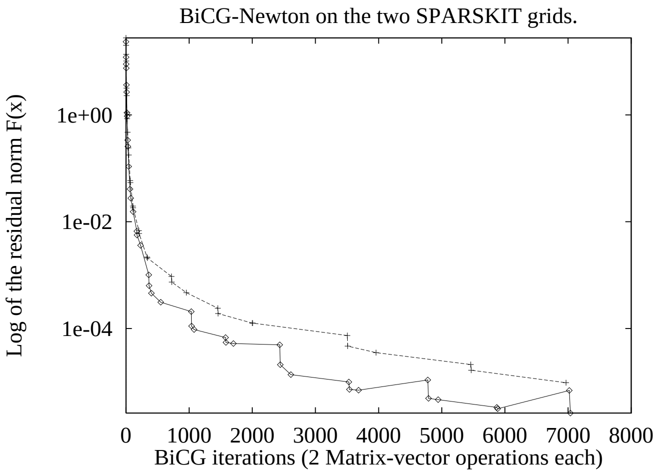

Figure 8. History of residual norm convergence of the BiCG-Newton method on the two SPARSKIT grids. The dotted line represents the convergence of the BiCG-Newton iteration for the grid of figure 6. The points mark the non-linear iterations, while the x axis shows the number of BiCG iterations.

displays a fast, global convergence, especially initially. In all the examples, BiCG-Newton reduces the residual norm by three orders of magnitude within the first 100 BiCG iterations. Convergence slows down gradually, evidence of the fact that the Jacobian becomes singular as it approaches the solution. Nevertheless, the method computed a residual norm of 10^{-6} for all cases, in reasonable time.

For large meshes, the high dimensionality of the non-linear system makes it hard to solve without an appropriate initial guess. In these cases, a weighted barycentric embedding may be needed first. In addition, the use of optimized sparse matrix codes can provide substantial improvements over the MATLAB code, and thus increase the solvable problem size.

5. From Angles to Geometric Embedding

Once we have the angles for each triangle, we can compute an embedding of the mesh; our original goal. Our basic algorithm uses breadth first search (BFS) to guide the embedding process.

Algorithm `BFS-Embedding`
Input: A topological embedding of a planar mesh M and an angle sequence for all triangular elements in the mesh.

1. Start with an arbitrary triangle $T = (u_0, v_0, w_0)$. Map the vertices of the edge (u_0, v_0), u_0 to $(0,0)$ and v_0 to $(1,0)$. Because we know the three angles of the triangle (u_0, v_0, w_0) and also its orientation, we can find the coordinates of w_0.
2. Let Q be a queue initially empty. Insert to Q the neighbor triangles of (u_0, v_0, w_0), where two triangles are neighbors if they have a common side.
3. **while** Q is not empty
   (a) Let $T = (u, v, w)$ be a triangle in Q. Note that at least one side, say (u, v), of T has already been embedded. If w has already been embedded, we do nothing. Otherwise, using the orientation and angles of T, we find the coordinates of w.
   (b) Insert the neighbor elements of (u, v, w) to Q.
4. Return the embedding.

Obviously, `BFS-Embedding` can be implemented in $O(n)$ time. We can also prove that the algorithm reconstructs the mesh correctly. First we give the following definition. An angle sequence is valid iff:

1. for each triangular element, its three angles add up to π.
2. for each interior vertex, the angles formed by all elements incident to the vertex add up to 2π.

3. in the execution of `BFS-Embedding`, any triangle T taken from the queue Q, either it has one vertex that has not been embedded, or the embedding of the three vertices satisfies its angles.

Theorem 2. If the angle sequence is valid for a planar embedding, then `BFS-Embedding` finds the correct embedding.

Proof. From tri-connectivity of the mesh, it follows that given a valid sequence of angles of the mesh, there is a unique embedding (up to dilation and rotation) that satisfies the angle sequence. Since the angle sequence is valid, the first step of the while loop is correct for the current triangle, when the third vertex has already been embedded. □

5.1. More Stable Embedding Algorithms

Although `BFS-Embedding` finds a correct embedding if the angle sequence is valid for a planar mesh, it is highly unstable. We can construct a simple example to show that alone the first two conditions of the definition are not sufficient. In solving the angle system, we usually obtain an angle sequence that is "approximately" valid, i.e., it can be viewed as a small perturbation of a valid angle sequence. Thus, it is important to have a stable algorithm to find a planar embedding from an angle sequence.

`BFS-Embedding` is not stable, largely because it incurs cumulative errors. When Condition (3) of the definition above is violated, the algorithm simply uses the current embedding for the triangle, hence, implicitly changes the angles of the element T. In turn, Condition (2) is violated at one or more of the vertices that are interior vertices. This error will propagate to subsequent triangular elements.

Figure 9 shows a manifestation of this error propagation when attempting to reconstruct the mesh from figure 6 after having converged to the angles with a residual 10^{-4}. Despite the error in individual angles being in the order of $O(10^{-3})$, `BFS-Embedding` results in local non-planar embedding. The same problem occurs on the much simpler uniform grid. When individual angles have been computed with accuracy $O(10^{-2})$, the reconstruction is still non planar as shown in figure 10.

This problem has been observed independently in [5,10], in the context of embedding a 3-D surface to a plane. The authors deal with the problem by enforcing an additional constraint on the computation of the 2-D angles. The constraint involves a product of sines of the angles of the elements around each

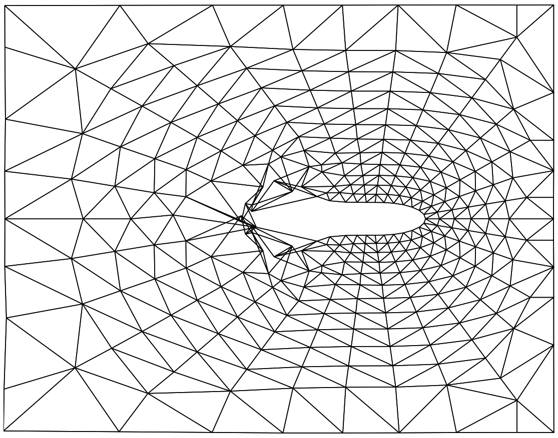

Figure 9. `BFS-Embedding` on grid from figure 6, with individual angle error $O(10^{-3})$.

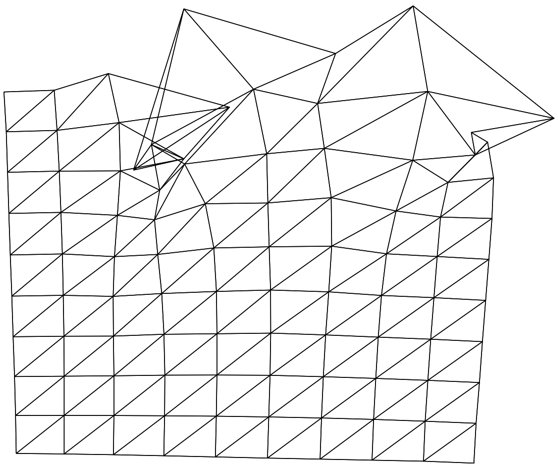

Figure 10. `BFS-Embedding` on the uniform grid, with individual angle error $O(10^{-2})$.

interior node, and it attempts to enforce appropriate lengths on the triangle edges. After the angles are obtained, a `BFS-Embedding` procedure yields the resulting mesh. However, the embedding still tends to accumulate errors because the above constraint applies directly to angles.

In our approach, we use the following two ideas to stabilize the embedding algorithm. The first idea is to define a global procedure to determine coordinates from angles. We use the following non-linear system to constrain the coordinates. Let $T = (u, v, w)$ be a triangle with vertices embedded at (x_u, y_u), (x_v, y_v), and

(x_w, y_w), respectively. Let θ_i be the angle at i, where $i \in \{u, v, w\}$. Then,

$$\Big((x_u - x_v)^2 + (y_u - y_v)^2\Big) \sin^2\theta_v = \Big((x_u - x_w)^2 + (y_u - y_w)^2\Big) \sin^2\theta_w$$
$$\Big((x_v - x_u)^2 + (y_v - y_u)^2\Big) \sin^2\theta_u = \Big((x_v - x_w)^2 + (y_v - y_w)^2\Big) \sin^2\theta_w$$
$$\Big((x_w - x_u)^2 + (y_w - y_u)^2\Big) \sin^2\theta_u = \Big((x_w - x_v)^2 + (y_w - y_v)^2\Big) \sin^2\theta_v$$

These three equations uniquely determine the side lengths of a triangle when its angles are given. If we have an initial coordinate sequence of the mesh, we can apply a few steps of a non-linear iteration to the above overdetermined system to improve the embedding. Therefore, working directly on coordinates and not on angles should alleviate the instabilities of the BFS scheme. We call this procedure `Coordinate-Smoothing`.

One way to obtain an initial coordinate sequence is to apply `BFS-Embedding`. To reduce the propagating effects of the error we introduce the following modification to the algorithm. All steps remain the same, except for the first step of the while loop, which becomes:

Algorithm `Modified BFS-Embedding`
Step 3 Modified:

3. Let $T = (u, v, w)$ be a triangle in Q. Suppose (u, v) of T has already been embedded. If w has already been embedded, we still compute the coordinates of w according to T and move w to the average of the newly computed position and the previous position. Otherwise, using the orientation and angles of T, we find the coordinates of w.

The above modification will still propagate the error if no previous embedded nodes are encountered in the BFS. In addition, `Coordinate-Smoothing` does not always correct long range losses of planarity. Despite these limitations however, our experiments in figures 11 and 12 show significant improvements over the poor embeddings of the same problems in figures 9 and 10 respectively.

6. Problems with Reducing onto the Boundary

Because of the computational expenses associated with the algorithms presented above, it is natural to seek techniques that reduce the dimensionality of the problem. In the special case of a mesh with no holes, a geometrical embedding of its boundary tube is sufficient for deriving all coordinate information using

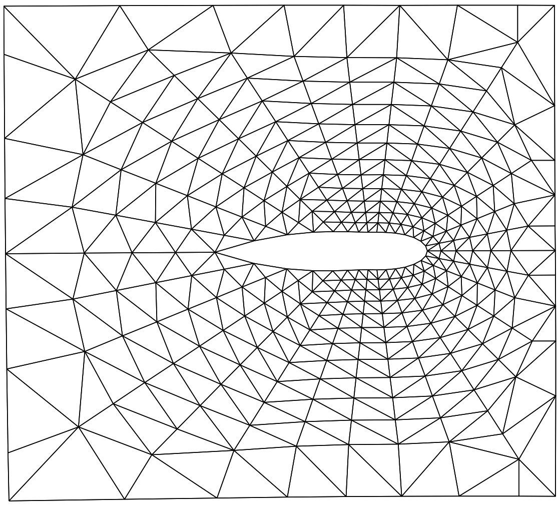

Figure 11. `BFS-Embedding` with averaging and `Coordinate-Smoothing` on grid from figure 6, with individual angle error $O(10^{-3})$.

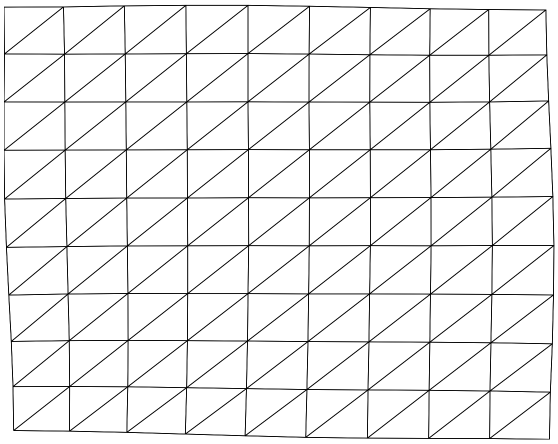

Figure 12. `BFS-Embedding` with averaging and `Coordinate-Smoothing` on uniform grid, with individual angle error $O(10^{-2})$.

equation (6). Surprisingly, the two approaches we have examined for solving this reduced problem have proven less efficient than the full angle technique.

The first approach utilizes a fixed point iteration that alternates between boundary and interior nodes, and can be described briefly in the following:

Algorithm `Boundary-Fixed-Point`

Input: A stiffness matrix A, an index of vertices on the boundary and in

the boundary tube, and an initial guess for the coordinates of the boundary vertices x_B, y_B.
while not converged

1. Using eq. (6) compute the coordinates of interior nodes x_I, y_I.
2. Considering the inner mesh without the boundary tube, use x_I and y_I to compute the angles that face the inner edges of the tube (edges that interface the tube and the inner mesh).
3. Consider the tube as a stand-alone mesh with its own stiffness matrix. The elements of this matrix are those of A corresponding to tube edges, except for the tube edges that face the inner mesh. The matrix elements of these interface edges are updated using eq. (1) and the cotangents of the angles from the inner mesh as computed in step 2.
4. Use the BiCG-Newton iteration to solve for the new angles of the boundary tube, and the Modified BFS-Embedding to obtain its node coordinates.

The computationally expensive steps of the algorithm are the solution of the linear system and the non linear iteration on the tube. For small boundaries these steps are expected to be significantly cheaper than a step of our full angle algorithm. However, our experiments with a MATLAB implementation of the above algorithm have identified several problems. The method seems to converge only if the initial guess for the boundary coordinates is almost accurate, and even in those cases, the fixed point convergence is extremely slow. The reasons can be traced in the requirement of the algorithm to move from angles to boundary coordinates at every step. These are obtained through the embedding process which is highly sensitive to angle accuracy. Thus, obtaining an adequately accurate initial guess is almost as hard as solving the non linear problem.

Moreover, the sensitivity to the embedding process continues to hold at every step. Because the elements in the tube lack a global coupling, each having a maximum of two neighboring elements, the Modified BFS-Embedding algorithm is not as effective as in the full mesh. As a result, convergence of the method occurs only if the tube angles are computed to a high accuracy.

A second approach is to use equation (6) to substitute for the interior points in the angle equations for the tube, and solve the non linear system with a Newton-like iteration. Again, we can consider the angles (or their cotangents) as unknowns, or we can formulate the system in terms of coordinates. However, with this approach the Jacobian becomes very complicated and less sparse than our full angle method. But more importantly, it requires the computation and storage of the full $w \times n_B$ matrix $A_I^{-1} A_B$. Computational alternatives exist, but all seem to have the same storage requirement. Finally, this approach demonstrates the same angle embedding sensitivity when the angles/cotangents are considered as unknowns. For the above computational reasons, and because of the limiting assumption of absence of holes, we have not pursued this second approach any further.

7. Beyond the Laplacian

In the preceding discussion we have considered the Laplacian operator with Neumann boundary conditions. The question arises whether similar techniques can be used to recover the mesh of simply modified Laplacians. For example, when the operator is scaled by some constant factor, the mesh recovered should be the same.

With a small modification, our techniques apply also in the case of a matrix $A = cL + D$, where L is the stiffness matrix of the Laplacian, c is a non zero scalar constant, and D is any diagonal matrix. The D term requires no change in the techniques, since the diagonal entries do not participate in setting up the non linear equations (1)–(4). The scalar c, however, changes the right hand sides of equations (1) and (2), which become:

$$\begin{aligned} \cot \phi_1^{(l)} + \cot \phi_3^{(k)} + 2cA_{v_i,v_n} &= 0 \\ \cot \phi_3^{(l)} + 2cA_{v_m,v_n} &= 0. \end{aligned}$$

Thus, it is necessary to include c as an additional unknown in the system of non linear equations. Since c does not multiply any unknowns, the modified Jacobian simply has an additional column with values $2A_{v_i,v_n}$ in the rows corresponding to equations (1) and (2). Our experience has been that this scalar factor is obtained within the first few iterations of our techniques.

An implication of the above generalization is that we can actually recover the mesh for an anisotropic elliptic operator of the form:

$$\frac{\partial}{\partial x}(K_x \frac{\partial}{\partial x}) + \frac{\partial}{\partial y}(K_y \frac{\partial}{\partial y}), \tag{9}$$

where K_x, K_y are scalar constants, not necessarily equal. Since one of the constants in the PDE can be factored out, it is only necessary to know their relative ratio, say K_x/K_y. It is well known, and easy to show, that when equation (9) is discretized on a given mesh, it gives rise to the same matrix as the Laplacian operator discretized on the same mesh but with the x coordinates dilated by $\sqrt{K_x/K_y}$. We can therefore apply our modified techniques that resolve any scaling factors, and after we obtain the mesh embedding, we dilate the coordinates of x to recover the original mesh of this anisotropic PDE. Notice that even if the relative ratio of the coefficients is not known, our algorithms will still compute a valid embedding of the Laplacian on a different mesh.

8. Conclusions and Future Research

Recovering the geometry of a mesh is a new, challenging problem. Our research was originally motivated by the observation that efficient numerical algorithms can be found if the geometry of a problem is known. However, many more interesting research questions have been raised during our current investigation, the solution of which is required by some applications and could potentially help many other applications.

In a more general context, geometry recovery is closely related to problems in graph embedding and graph drawing. In our research, we assume that the matrix stems from the discretization of a PDE on a 2D mesh, and we want to find an embedding that is as close as possible to the original mesh. We have shown that for the Laplacian operator, the stiffness matrix contains sufficient amount of information about the geometry of the mesh. It is possible to solve numerically for this geometric information, albeit through a non linear iteration. We have also shown that the mesh can be recovered even if the Laplacian matrix is multiplied by a scalar, or if a diagonal matrix is added to it. An anisotropic operator with constant coefficients can also be reduced to a Laplacian. There are many problems that remain open and several directions that this research can be continued.

Extensions of our results to other operators and to other boundary conditions besides Neumann, or to other element shapes and higher order finite elements are not straightforward. We would also like to extend our work to the more complex, 3D case, by designing a system of equations for the coordinates of the vertices directly. Such a formulation may also improve the stability of the 2D algorithm. Finally, more stable techniques are needed for embedding an approximately valid angle sequence. A promising direction is a multilevel algorithm that would embed the fine mesh by first embedding a hierarchical sequence of coarser meshes. What is fascinating is that there are immediate applications of this problem, and its solution seems feasible.

An interesting question is how these techniques can be used to enhance our understanding and performance of algebraic multigrid. For example, given a stiffness matrix for some PDE, we could find instead a different PDE and some mesh that yield the same stiffness matrix. A different approach would be to recognize the class of well-shaped meshes that can be used in a multilevel preconditioning scheme for a given stiffness matrix.

Throughout these questions, however, it is important not to loose sight of the fact that the general problem of mesh recovery is an ill-posed, inverse problem. Often, the quality of the mesh recovered will be determined only in the context of the reference problem.

Acknowledgments

This work has been supported by the Computational Science Cluster and the Department of Computer Science at the College of William and Mary, and the University of Illinois at Urbana-Champaign.

References

[1] T. F. Chan and J. Zou. Additive Schwarz domain decomposition methods for elliptic problems on unstructured meshes. CAM Report 95-16, Department of Math, UCLA, March 1995.

[2] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact Newton Methods. *SIAM J. Numer. Anal.*, 19: 400–408, 1982.

[3] J. E. Dennis, Jr. and Robert B. Schnabel. Numerical Methods for Unconstrained Optimization and Nonlinear Methods. Classics in applied mathematics. SIAM, 1996.

[4] E. Hinton and D. R. J. Owen. Finite Element Computations. Pineridge Press Limited, 1979.

[5] J. Liesen and E. de Sturler and A. Sheffer and Y. Aydin and C. Siefert. Efficient Computation of Planar Triangulations. In Proc. of the 10th International Meshing Round Table, (2001), accepted.

[6] G. L. Miller, S.-H. Teng, W. Thurston, and S. A. Vavasis. Automatic mesh partitioning. In A. George, J. Gilbert, and J. Liu, editors, *Sparse Matrix Computations: Graph Theory Issues and Algorithms*, IMA Volumes in Mathematics and its Applications. Springer-Verlag, 1993.

[7] J. Ruge and K. Stüben, Algebraic Multigrid (AMG), in Multigrid Methods (S. McCormick, Ed.), Frontiers Appl. Math. 5, SIAM, Philadelphia, 1987.

[8] Yousef Saad, SPARSKIT: A basic toolkit for sparse matrix computations. *RIACS, NASA Ames Research Center*, TR90-20, Moffet Field, CA, 1990.

[9] Yousef Saad. Iterative methods for sparse linear systems. PWS Publishing Company, 1996.

[10] A. Sheffer and E. de Sturler. Parameterization of Faceted Surfaces for Meshing Using Angle Based Flattening. *Engineering with Computers*, 2002, to appear.

[11] Shimon Even. Graph Algorithms. Computer Science Press, 1979.

[12] D. Spielman and S.-H. Teng. Spectral partitioning works; Planar graphs and finite element meshes *FOCS*, 1996.

[13] W. T. Tutte. Convex representations of graphs. *Proc. London Math. Soc.* 10(3): 304–320, 1960.

[14] W. T. Tutte. How to draw a graph. *Proc. London Math. Soc.* 13(3): 743–768, 1963.