NUMERICAL METHODS FOR PROBLEMS WITH MOVING MESHES

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Suzanne Michelle Shontz

January 2005

© Suzanne Michelle Shontz 2005

ALL RIGHTS RESERVED

NUMERICAL METHODS FOR PROBLEMS WITH MOVING MESHES

Suzanne Michelle Shontz, Ph.D.

Cornell University 2005

The first part of the thesis presents a new mesh warping framework for tetrahedral meshes based upon weighted laplacian smoothing. We start with a 3D domain that is bounded by a triangulated surface mesh and has a tetrahedral volume mesh as its interior. We then suppose a surface mesh movement is prescribed and use an algorithm within our framework to update the interior nodes. First, a set of local weights is determined for each interior node that describes its position in terms of its neighbors' positions. After a boundary transformation is applied, a linear system of equations based upon the weights is solved to determine the interior nodes' final positions. The three steps comprise the linear weighted laplacian smoothing (LWLS) framework. We present two methods within this framework. The first, LBWARP, uses a log-barrier approach to compute the weights. The second, FEMWARP, is based upon the finite element method. We study mesh invertibility and prove LWLS algorithms yield exact results for affine mappings and a theorem for general mappings within the context of FEMWARP. We prove another theorem on the

resulting mesh. We test the algorithms' robustness and present some numerical results. Finally, we use FEMWARP to study the movement of the canine heart.

The second part of the thesis proposes a solution procedure for the finite element discretization of a static nonlinear compressible Mooney-Rivlin hyperelastic solid. We consider the case in which the boundary condition is a large prescribed deformation, so that mesh tangling becomes an obstacle for straightforward algorithms. Our procedure involves two geometric operations to untangle the mesh: solution of a linear system to obtain initial guesses for interior positions followed by the Freitag-Plassmann mesh untangling algorithm. Then, we take Newton iterations to converge to a mechanical equilibrium. The Newton iterations are safeguarded by a line-search similar to one used in optimization. Our computational results indicate that the algorithm is much more robust (i.e., able to tolerate much larger deformations) and much faster than a straightforward Newton-continuation procedure.

Biographical Sketch

Suzanne Michelle Shontz was born on January 4, 1976 in Cedar Falls, Iowa to Gary and Margaret Shontz. Soon, brother, Robert, and five years later, sister, Roxanne, joined the family. It was there that Suzanne grew up. At the age of 14, she participated in the National Spelling Bee.

Soon afterwards, her family moved to rural Cedar Falls. This meant a new life and a new school system. Suzanne participated in many high school activities including sports, music, speech, and the talented and gifted program. Twice she was chosen to participate in All-State Band as a clarinetist. Suzanne graduated as valedictorian of the Class of 1994 from Dike-New Hartford High School.

In the fall of 1994, Suzanne enrolled in classes at the University of Northern Iowa. There she majored in mathematics and chemistry and minored in physics. At UNI, Suzanne participated in (and served as an officer of) many academic organizations. She participated in summer mathematics research programs at the Geometry Center at the University of Minnesota, the Univeristy of Kentucky, and Cornell University. She co-founded the Mathematics Club her junior year and received the College of Natural Science's Dean's Student Award in 1998. She graduated summa cum laude in 1999, the first graduate to receive the Purple and Old Gold Award for meritorious scholarship in more than one major. That year, she also received an Honorable Mention for the Alice Schafer Prize in Mathematics which is awarded by the Association for Women in Mathematics.

In 1999, Suzanne moved to Ithaca, New York to pursue her doctorate degree in Applied Mathematics at Cornell University as a National Physical Science Consortium Fellow. In 2002, she completed masters degrees in Applied Mathematics and Computer Science. At Cornell, Suzanne participated in the College of Engineering Graduate Student Association where she held various offices. She also was a member of Graduate Christian Fellowship and a book club for Christian women. During the summers, Suzanne held internships at Sandia National Laboratories-CA, Argonne National Laboratory, and Sandia National Laboratories - NM. She also taught the Introduction to Scientific Computing class at Cornell. Now, five years later, she has achieved her goal and finished her Ph.D.

Her next step will be to move to Minneapolis, Minnesota, where she will begin a postdoctoral position in the Department of Computer Science and Engineering at the University of Minnesota. Under the direction of Yousef Saad, she will do research in parallel numerical linear algebra as applied to material science.

To my parents, Gary and Margaret, for their love and guidance throughout the years. Thanks, Mom and Dad!

To my brother and sister for their love and encouragement throughout life. Thanks, Robert and Roxanne!

Thanks to each of you for your support during my graduate school career at Cornell University!

Acknowledgements

There are many people who helped me throughout graduate school and in completing this thesis. Without them this thesis would not be a reality. I would first like to thank my advisor, Steve Vavasis. Steve's insight, guidance, encouragement, support, and enthusiasm were all invaluable and greatly appreciated. I would also like to thank the other members of my special committee: Tom Coleman, Lars Wahlbin, and Robert Gilmour, Jr. They provided me with invaluable feedback and advice along the way. Tom Coleman sparked my interest in optimization, and Robert Gilmour, Jr. really kindled my interest in cardiology. Lars Wahlbin taught me a greater appreciation for finite element analyses. Charlie Van Loan served as a proxy for my A-exam and has been the source of invaluable advice on teaching and academic career mentoring. Jon Kleinberg was the one who taught me to think algorithmically and was an example of an excellent lecturer. Steve Strogatz recruited me to come to Cornell and was an amazing lecturer. Tim Healey was my temporary advisor and provided me much (often comical!) insight into the academic world; he continues to give me good advice on academia.

There are many people who made this research possible. Owen Faris and Eliot

McVeigh from the Laboratory of Cardiac Energetics at the National Institutes of Health provided us with the moving canine heart data. Lori Freitag-Diachin and Patrick Knupp of Lawrence Livermore National Laboratory and Sandia National Laboratories, respectively, provided us with many of the 2D and 3D test meshes. We benefited from conversations with Graham Bailey, Al Schatz, and Cristina Patron of Cornell; Tiberiu Tomita, formerly of Cornell; and especially Harry Kesten and Katerina Papoulia of Cornell.

I spent the majority of my summers during graduate school working in the government labs. I spent time at Sandia National Laboratories in Livermore, CA; Argonne National Laboratory in Argonne, IL; and Sandia National Laboratories in Albuquerque, NM. While at the labs, I worked with Patty Hough, Vicki Howle, Lori Freitag-Diachin, Todd Munson, and Patrick Knupp. Collectively, they have taught me a lot about research in computational science and engineering. I've throughly enjoyed working with them and have much appreciated their mentoring and experience. I look forward to continued collaborations with these individuals. Thanks to the Computational Sciences and Mathematics Research Department at Sandia National Laboratories - CA, the Mathematics and Computer Science Division at Argonne National Laboratory, and the Parallel Computing Sciences Department at Sandia National Laboratories-NM for hosting me during these internships. I've met many great computational scientists during my time spent in these departments.

I've also been fortunate to have received great mentoring from two members of the Association for Women in Mathematics as part of their Mentor Network. Although I have yet to meet her, Misha Kilmer provided me with mentoring on

academic careers and excellent feedback on my job application materials. Anne Hall taught me about industrial careers and hosted me for a visit of the General Electric plant in Milwaukee, WI.

During my time as an undergraduate at the University of Northern Iowa, I was fortunate to have received encouragement, support, mentoring, and inspirational teaching from many professors. Thanks to those faculty members in the Mathematics and Chemistry Departments who touched my life in this way. Special thanks to Tim Hardy for being the one to first spark my interest in numerical analysis - my chosen field of research.

As an undergraduate, I had the good fortune to participate in no fewer than three summer research programs. The first one was at the Geometry Center at the University of Minnesota in Minneapolis, MN. There I worked with Jesus DeLoera and Rick Wicklin who are both alumni of the Center for Applied Mathematics at Cornell. There I learned that I loved research which helped solidify my determination to go to graduate school. I also studied with Peter Perry and John Hubbard at the University of Kentucky and Cornell University, respectively, in the National Science Foundation's Research Experiences for Undergraduates programs. I also did research in analytical chemistry with Curt Hanson at the University of Northern Iowa. Thanks to all of these individuals for my first introductions to research.

Many people have made my graduate years at Cornell memorable. Dolores Pendell provided excellent administrative support and was always very knowledgeable about graduate life at the university. Thanks to my classmates - Siddharth Alexander, Marcel Blais, Ariel Cintron-Arias, Jaime Barrera, Bill Dirks, Sharad Goel,

Johnny Guzman, Carla Martin, Jason Martin, Virginia Pasour, Jesus Rodriguez, and Mike Singer - without whom it would have been difficult to survive classes and graduate school. You were a fun group with which to study and socialize. Thanks also to Katharyn Boyle, Tara Small, Bess Chen, Erin Iesulauro, Prashant Mehta, Jay Henniger, and Deena Schmidt for fun conversations and for hanging out; I've enjoyed your friendship. Special thanks to Cristina Patron, Kathleen Vogel, and Eric Hayduk for all of the conversations, smiles, encouragement, and for spending time with me throughout graduate school; your friendship has been invaluable to me.

I've been fortunate to participate in a few organizations during my time at Cornell. Thanks to the members of the Engineering Graduate Student Association at Cornell for organizing many activities in which graduate students could participate. Additional thanks to the members of Graduate Christian Fellowship at Cornell for good Bible studies, fellowship, music, and the women's group. Thanks to the Christian women's reading group; I enjoyed our discussions and your friendship. I'd also like to thank everyone at Ithaca Baptist Church for your support and friendships during my time at Cornell. Finally, thanks to those who played volleyball with me at Cornell for giving me a good way to blow off steam (and to have lots of fun doing it!).

Most importantly, I would like to thank my family for always being there for me and for supporting me every step of the way. They have been very encouraging to me in my pursuit of a Ph.D. I hope they can understand how much their love and support means to me.

Finally, there is the inevitable need for funding. Throughout graduate school, I have been supported by a fellowship from the National Physical Science Consortium. As members of the consortium, Sandia National Laboratories - California and Cornell University provided my stipend and tuition, respectively. Patty Hough and Tammy Kolda served as my NPSC Fellowship mentors and provided me with lots of valuable advice. In addition, this work and research has been supported in part by NSF through ACI-0085969.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Moving meshes arise in cardiology, computer graphics, animation, and crash simulation, among other applications in science and engineering. With moving meshes, the mesh is updated at each step in time due to a moving domain boundary, thus resulting in potentially drastically varying mesh quality from step to step. One problem that can occur at each timestep is element inversion. We focus on maintaining good-quality tetrahedral meshes throughout the mesh warping process in this thesis.

It is well-known that poor quality elements affect the stability, convergence, and accuracy of finite element and other solvers because they result in poorly conditioned stiffness matrices and poor solution approximation [45]. If well-shaped elements are not the result of updating the mesh boundary, the mesh quality must be improved by topological or geometrical means after each time step.

Research has shown that mesh smoothing (or r-refinement) methods can be applied to improve the quality of a mesh. These methods adjust the positions of the vertices in the mesh while preserving its topology.

Laplacian smoothing is the most popular method for node-based mesh smoothing. In an iterative manner, it repositions the vertices of the mesh by moving each interior node to the geometric center of its neighbors. It is often used because it is computationally inexpensive and is very easy to implement. However, the method has several undesirable properties. One of them is that the method is not guaranteed to work, i.e., sometimes it inverts mesh elements. A second drawback is that if the algorithm is not run to convergence, the resulting mesh depends upon the order in which the nodes are smoothed.

A related type of smoothing, namely Winslow smoothing, is more resistant to mesh folding due to the requirement that the logical variables be harmonic functions. This method was introduced for structured meshes by Winslow in [52]. In [27], Knupp extends this idea to unstructured quadrilateral meshes; he used a finite-difference method for his derivative computations. Others have used the finite element method for the derivative computations on unstructured meshes; for example, Knupp cites a proprietary DOE paper by Tipton to which the authors do not have access. (See the references at the end of [27].)

Many others have created extensions to Winslow's method. One type of extension has been to generate the meshes using a variational approach; for example, see Brackbill and Saltzman [5], Russell and co-workers [8] and [22], and Thompson et al. [47]. Dvinsky [10] uses the theory of harmonic functions to generate a harmonic map between the physical domain and the logical domain in order to generate the mesh. Meshes that have been generated according to this method possess regularity and smoothness properties. Li et al. [30] also introduced a moving mesh method based on harmonic maps; their method has two parts: a solution algorithm and a mesh-redistribution algorithm. In doing so, they are able to obtain the desirable properties of both the h-method and r-method for finite elements.

Other, more accurate methods for r-refinement are possible. Most of these methods are based upon optimization. Optimization-based methods are used with the goal of guaranteeing an improvement in the mesh quality by minimizing a particular mesh quality metric. Their main drawback, however, is their computational expense. Examples of optimization-based methods for r-refinement can be found in the following papers: [15], [14], [13], [16], [12], [17], [1], [2], [37], [7], [50], and [54]. For a theory of algebraic mesh quality metrics see [28].

To address the above issues, Baker [3] developed a three-step method for the metamorphosis of tetrahedral meshes. Each cycle of the method involves a combination of r-refinement, mesh coarsening, and mesh enrichment to adapt the mesh. The first step in the cycle is to move the interior nodes as far as possible using r-refinement while avoiding element inversion. The second step is to remove the poorly shaped elements in the mesh using mesh coarsening. The final step is the addition of elements to improve the mesh quality by mesh refinement. This dynamic procedure was shown to be more cost-effective than just r-refinement. One disadvantage of this technique is that it comes with no theoretical guarantees as it is difficult to analyze because each cycle is a combination of three very different techniques.

We study a different mesh warping problem where the connectivity of the mesh is not allowed to change, which is important for some applications. The problem that we address is as follows: Given a 3D domain, bounded by a triangulated surface mesh, and given an interior volume mesh composed of unstructured tetrahedra, suppose the triangulated surface mesh is displaced. Is there an algorithm to move the nodes of the volume mesh so that it continues to conform to the surface mesh and to be a good quality mesh? Seeking simplicity, preservation of the mesh's combinatorial structure, theoretical guarantees, and low computational expense, in Chapter 2, we develop a linear weighted laplacian smoothing (LWLS) framework for mesh warping in two and three dimensions.

In Chapter 3, we propose a solution procedure for the finite element discretization of a particular mechanical model from continuum mechanics. The continuum mechanical model we study in this thesis can be described as follows [21]. Let $B_0 \subset \mathbf{R}^d$ be an undeformed body whose boundary is ∂B_0. Here d, the space dimension, is 2 or 3. Suppose that a pure Dirichlet problem is considered, i.e., a continuous injective function $\phi : \partial B_0 \to \mathbf{R}^d$ is given that prescribes a new position of the boundary. Let ∂B denote the deformed boundary, i.e., $\partial B = \phi(\partial B_0)$. The problem is to find a continuous extension of ϕ that maps all of B_0 to a set $B \subset \mathbf{R}^d$ satisfying the following conditions. For a point $\mathbf{X} \in B_0$, let $\mathbf{x} = \phi(\mathbf{X})$. Let F be the deformation gradient, i.e., $F = d\mathbf{x}/d\mathbf{X}$. The *Green strain* is defined to be $E = (F^T F - I)/2$. Let $\Psi(F)$ be the *strain energy function.* Of interest to us is the compressible Mooney-Rivlin model for the strain energy of hyperelastic solids which is the following:

$$\Psi(F) = c(J-1)^2 - k\ln(J) + c_1(I_1 - 3) + c_2(I_2 - 3),$$

where c, c_1, c_2, and $k > 0$ are material constants and I_1, I_2, and I_3 are three particular matrix invariants of E. We seek to find a deformed mesh for a prescribed boundary deformation which satisfies the static equilibrium condition in the absence of body forces. The variational form of this condition is given by

$$\int_{B_0} \frac{\partial \Psi}{\partial F} : \mathrm{Grad}\, \delta \mathbf{u}\, dV = 0,$$

where $\delta\mathbf{u}$ is a test function that vanishes on the boundary. We consider the case in which the boundary condition is a large prescribed deformation, so that mesh tangling becomes an issue for straightforward algorithms. Our solution procedure involves the use of two geometric operations to untangle the mesh. The first of them is FEMWARP which we describe in Section 2.3. The second is the Freitag-Plassmann untangling algorithm which we describe in Section 3.3. After the mesh is untangled, we take safeguarded Newton iterations to converge to a mechanical equilibrium.

Chapter 2

A Linear Weighted Laplacian Smoothing Framework for Warping Tetrahedral Meshes*

2.1 Linear Weighted Laplacian Smoothing (LWLS) Framework

Here we describe the 2D linear weighted laplacian smoothing framework; it extends to 3D in the obvious way.

The first step within the LWLS framework is to determine a set of local weights for each interior node which describes each interior node in terms of the positions of

*Portions of this chapter were submitted as an article entitled, "A linear weighted laplacian smoothing framework for warping tetrahedral meshes", to the SIAM Journal of Scientific Computing in July 2004.

its neighbors. To this end, let (x_i, y_i) denote the x- and y-coordinates respectively of the ith interior node in the initial mesh. In addition, let the x- and y-coordinates of its adjacent vertices be given by $\{(x_j, y_j) : j \in N_i\}$, where N_i denotes the set of neighbors of node i. We denote the set of weights by w_{ij}, where w_{ij} is the weight of node j on interior node i. Each set of local weights satisfies the following set of equations:

$$x_i = \sum_{j \in N_i} w_{ij} x_j, \tag{2.1}$$

$$y_i = \sum_{j \in N_i} w_{ij} y_j, \tag{2.2}$$

$$\sum_{j \in N_i} w_{ij} = 1. \tag{2.3}$$

Below we propose two distinct methods, LBWARP and FEMWARP, for computing the w_{ij}'s.

We first establish some necessary notation and terminology. Let b and m represent the numbers of boundary and interior nodes, respectively. Next, define x_B and y_B to be vectors of length b that contain the initial x- and y-coordinates of the boundary nodes. Similarly, let x_I and y_I be vectors of length m that contain the initial x- and y-coordinates of the interior nodes. Then $[x_B,\ y_B]$ and $[x_I,\ y_I]$ contain the original positions of the boundary and interior nodes respectively. Next, define the weighted Laplacian matrix, L, for a weighted graph $G(V; E; w)$ as follows:

$$L(i,j) = \begin{cases} -w_{ij} & \text{if } i \neq j, \\ \sum_{k \in V} w_{ik} & \text{if } i = j, \end{cases}$$

where $w_{ij} = 0$ if $(i,j) \notin E$. The boundary nodes are assumed to be numbered last.

Denote by $A = [A_I,\ A_B]$ the matrix that is derived from the weighted Laplacian matrix by deleting its last b rows. Note that A_I contains all of the weights corresponding to the interior neighbors and is $m \times m$. In addition, A_B contains all of the weights corresponding to the boundary neighbors and is $m \times b$. Linear equations (2.1)–(2.2) then can be expressed

$$A_I[x_I,\ y_I] = -A_B[x_B,\ y_B]. \tag{2.4}$$

The second step is to apply a user-supplied transformation to the boundary of the mesh. We denote the new positions of the boundary nodes by $[\hat{x}_B,\ \hat{y}_B]$.

The final step is to solve a linear system of equations for the new positions of the interior nodes. The linear system we use is the same as in (2.4), except it has a new right-hand side vector which accounts for the new positions of the boundary nodes. Thus, we solve (2.5) for $[\hat{x}_I,\ \hat{y}_I]$:

$$A_I[\hat{x}_I,\ \hat{y}_I] = -A_B[\hat{x}_B,\ \hat{y}_B]. \tag{2.5}$$

There are two main advantages to methods within the LWLS framework. First, if a continuous deformation of the boundary is given, then methods within the framework are valid for computing the resulting trajectory that specifies the movement of the interior nodes. In addition, these trajectories will be continuous. This is vital for some applications where continuity of motion is required. A second big advantage is that sparse matrix algorithms may be used to solve (2.5). The sparsity structure is apparent, since, on average, an interior node has six neighbors in 2D, whereas a typical 2D mesh may have hundreds, thousands, or even millions of nodes.

Within this framework, we present two methods for mesh warping in two and three dimensions. The first method, LBWARP, is based on a log-barrier method. For this method, each interior node is represented as a convex combination of its neighbors. Thus, the local weights are guaranteed to be positive, which means that nodal positions are being interpolated. A full description of the LBWARP method is given in Section 2.2.

The second method, FEMWARP, is based on the finite element method. The weights computed by this method can be rescaled so that they also sum to 1. The FEMWARP method often computes negative weights, especially in 3D. This corresponds to the extrapolation of nodal positions. As a general principle, interpolation is preferred over extrapolation in numerical analysis, although it is not certain whether or not this is important within this context. The FEMWARP method is described fully in Section 2.3.

Each of the methods has its advantages and disadvantages. The main advantage of the LBWARP method is that it always interpolates the values of the nodal positions, which is not always the case with FEMWARP. On the other hand, it is more straightforward to compute the weights for the FEMWARP method, as the LBWARP method requires that an optimization problem be solved to determine each set of local weights. In addition, because the matrix A_I is symmetric positive definite (rather than an M-matrix), the computation of the new positions for the interior nodes also proceeds at a faster rate for FEMWARP. Another big advantage of FEMWARP is that it is somewhat easier to develop the theory of this method.

In Section 2.4, we prove a theorem that gives sufficient conditions for when a mesh is able to resist inversion by a transformation. We also study mesh inversion within the context of the LWLS framework. In particular, we prove that all algorithms within the LWLS framework yield exact results for affine boundary transformations. In addition, we prove a theorem for more general mappings within the context of FEMWARP. We also prove a lower bound for the LBWARP weights. In Section 2.5, we prove that for LBWARP, the resulting mesh is the same as the converged mesh obtained from the local version of weighted laplacian smoothing and from the Gauss-Seidel algorithm. This result also holds for FEMWARP and for any LWLS method when the latter two methods converge. In Section 2.6, we test our algorithms on several types of mesh deformations. In Section 2.7, we apply FEMWARP to study the motion of the beating canine heart under normal conditions. In Section 2.8, we summarize our work and identify directions for future research.

2.2 Log-Barrier Mesh Warping (LBWARP)

Our first mesh warping algorithm within the new framework is based upon a log-barrier method. The first step in the algorithm is to generate a set of local weights for each interior node that represent the relative distances of the node to each of its neighbors. We use an interior point method from nonlinear programming in order to generate these weights. The rest of the algorithm follows the LWLS framework described in Section 2.1: We apply a transformation to the boundary nodes. Using these new positions for the boundary nodes and the sets of weights from the original mesh, we solve a system of linear equations to determine new positions for the interior nodes. We now give a more detailed description of the weight computation.

Here we describe a nonlinear programming method for computing the weights for 2D meshes. Note that our method can be extended to 3D in the straightforward manner; however, we have not done extensive testing of our 3D implementation. In addition, this thesis analyzes primarily the 2D version.

In order to find the set of weights w_{ij}, where w_{ij} is the weight of node j on interior node i, we use the log-barrier function from nonlinear programming to formulate the following optimization problem for each i:

$$\begin{aligned} \max_{w_{ij}, j \in N_i} & \quad \sum_{j \in N_i} \log(w_{ij}) && (2.6) \\ \text{subject to } w_{ij} &> 0, && (2.7) \\ \sum_{j \in N_i} w_{ij} &= 1, && (2.8) \\ x_i &= \sum_{j \in N_i} w_{ij} x_j, && (2.9) \\ y_i &= \sum_{j \in N_i} w_{ij} y_j. && (2.10) \end{aligned}$$

We note that the objective function together with the constraints form a strictly convex optimization problem for which there is a unique optimum. The optimum can thus be found by an interior point method. The use of convex optimization in mesh smoothing is not a new idea. Amenta et al. developed a framework for formulating mesh smoothing problems as quasi-convex programs so that they can be optimized as generalized linear programming problems [1]. In addition, Freitag et al. developed methods for local mesh smoothing and untangling using mesh quality metrics with convex function levels sets in [12] and [17].

By starting with an initial feasible point in the interior of the space of all possible weights, we are able to solve (2.6)-(2.10) using the projected Newton method. This method can be applied to solve optimization problems of the following form:

$$\begin{array}{rl} \min\limits_{x} & f(x) \\ \text{subject to} & Ax = b. \end{array}$$

The optimality conditions for the above problem are:

$$\nabla f(x) - A^T\lambda = 0 \tag{2.11}$$

$$b - Ax = 0. \tag{2.12}$$

Here λ denotes the vector of Lagrange multipliers. Note that this is a system of nonlinear equations in x and λ; thus Newton's method can be used to solve for x and λ. However, if the initial point is feasible (i.e., $b - Ax_0 = 0$) then p_k must satisfy $Ap_k = 0$ where $x_{k+1} = x_k + p_k$, i.e., p_k is the Newton direction at iteration k. This implies that $p_k = Zv_k$ for an $(n - m)$-dimensional vector v_k, where Z is a basis for the null space of A. It is easy to see that the resulting equation for p_k is given by

$$p_k = -\left(H^{-1} - H^{-1}A^T[AH^{-1}A^T]^{-1}AH^{-1}\right)\nabla f(x_k), \tag{2.13}$$

where $H = \nabla^2 f(x_k)$ [34].

In order to compute an initial feasible point in the interior of the solution space, we choose three of the interior node's adjacent vertices and write (x, y) as a convex combination of the positions of the three nodes. This yields three positive weights; call them w_1, w_2, and w_3. In order to find a set of m positive weights, the rest of the w_{ij} are initialized to ϵ, a small positive constant. Then once ϵ has been specified, linear equations (2.8) through (2.10) can be solved for new values of w_1, w_2, and w_3. For small values of ϵ, all of the constraint equations above will be satisfied (using the new values for the weights). An inexact linesearch is used to make ϵ as large as possible so that w_1, w_2, and w_3 are still reasonably sized, i.e., each of them should be greater than some small tolerance.

Once the weights have been generated, as in the general LWLS framework, the input surface deformation is applied to move the boundary nodes to new locations. The final step is to use the positions of the boundary nodes and the sets of weights to simultaneously determine the final positions for the interior nodes by solving a linear system of equations. The linear equations that must be solved are given in (2.5).

We first prove that the resulting linear system has a unique solution.

Theorem 1. *The linear system (2.5), which expresses the position of each interior node as a convex combination of its neighbors, has a unique solution.*

Proof. To argue that we have uniqueness when LBWARP is used to generate the weights, we classify A_I as an M-matrix which is defined as follows:

An $n \times n$ real matrix A is said to be an *M-matrix* if $a_{ij} \leq 0$ for all $i \neq j$ and $A^{-1} \geq 0$.

To see that A_I is an M-matrix, we make use of the following characterization theorem:

Theorem 2. *[51] If $A \in \mathbf{R}^{n \times n}$ satisfies $a_{ii} > 0$, $a_{ij} \leq 0$ for all $i \neq j$, is weakly row diagonally dominant (i.e., $Ae \geq 0, \neq 0$, where e is the vector of all 1's), and is irreducible, then A is an M-matrix.*

Because the mesh is connected and a positive weight is associated with each edge, we see that A_I is irreducible. Also, A_I is diagonally dominant, as its diagonal entries are 1, and its off-diagonal entries are negative and sum to a number in $[-1, 0]$ in each row. Since there is at least one interior node adjacent to a boundary node, $A_I e_I \neq 0$, where e_I is a vector of all 1's of length m. Therefore, A_I satisfies the definition of an $M-$matrix by the above argument [36]. Thus, A_I is invertible, and (2.5) has a unique solution. $\square$

We next solve (2.5) via Gaussian elimination. Partial pivoting is not necessary, because A_I is weakly diagonally dominant. This is a major advantage, as sparse matrix algorithms are much more efficient without pivoting.

Because the nonzero pattern is symmetric, we can first apply the symmetric minimum degree ordering to the appropriate matrix in an attempt to improve the speed of the computations. The reader is referred to [19] for discussion on the computer solution of sparse linear systems. GMRES could also be used to solve the linear system instead of Gaussian elimination in an attempt to further improve the algorithm's speed although we have not tried this. See [38] for details of the GMRES algorithm and other iterative methods for solving linear systems of equations.

2.3 Finite Element Mesh Warping (FEMWARP)

Our second mesh warping algorithm within the new framework uses the finite element method to generate the weights during the first step. These weights are different from the LBWARP weights in that they are not guaranteed to be positive. We now give a more detailed description of this algorithm.

Here we describe a piecewise-linear finite element method for computing the weights for 2D meshes. This method can also be extended to 3D in the straightforward manner. However, this thesis analyzes primarily the 2D version.

Let N be the number of triangles in the mesh. In order to determine $\{w_{ij}\}$, we use a piecewise-linear finite element method to first compute the finite element discretization of the following:

$$\begin{aligned} \Delta u &= 0 \text{ on } \Omega \\ u(x,y) &= (x,y),\ \forall (x,y) \text{ on } \partial\Omega. \end{aligned}$$

This is done by first computing the global stiffness matrix A, where $A(i,j) = \int_\Omega \nabla\phi_i \cdot \nabla\phi_j \, d\Omega$, where ϕ_i is a piecewise-linear shape function which satisfies $\phi_i(v_i) = 1$ and $\phi_i(v_j) = 0, \forall j \neq i$. Second, A_I and A_B are extracted from A. To this end, we set $A_I = A(1:m, 1:m), A_B = A(1:m, m+1:m+b)$. Note that symmetry in A can be used to reduce both the number of computations and amount of memory required.

Denote by $\hat{A} = [A_I,\ A_B], x = [x_I;\ x_B]$, and $y = [y_I;\ y_B]$. Then, $\hat{A}[x,\ y] = \mathbf{0}$. Furthermore, we note that for $i = 1, \ldots, m$, the entries in the ith row of $\hat{A}$ (and thus also of A) satisfy the following equations:

$$\hat{A}(i,i)x_i = \sum_{j \in N_i} -\hat{A}(i,j)x_j \tag{2.14}$$

$$\hat{A}(i,i)y_i = \sum_{j \in N_i} -\hat{A}(i,j)y_j. \tag{2.15}$$

Thus, if we set $w_{ij} = \frac{-\hat{A}(i,j)}{\hat{A}(i,i)}, \forall i \neq j$, we obtain the desired set of weights.

Once the weights have been generated, the input surface deformation is applied to move the boundary nodes to new locations. The final step is to use the positions of the boundary nodes and the sets of weights to simultaneously determine the final positions for the interior nodes by solving the linear system of equations given in (2.5).

We note that this method was proposed by Baker in [3]. However, he rejected the method and chose not to develop it since he thought that a method based upon linear elasticity would be more successful, as the displacements for the x and y variables are computed in a coupled manner. We will describe linear elasticity in Section 3.1 and will show in Section 2.6 it is sometimes the case that FEMWARP outperforms a linear elasticity solver.

We first prove that the resulting linear system has a unique solution.

Theorem 3. *The linear system* (2.5)*, which expresses the position of each interior node as a linear combination of its neighbors, has a unique solution.*

Proof. It follows from [25] that A is a symmetric, positive-definite matrix. Thus, A_I is symmetric, positive-definite (and hence invertible) since it is the $m \times m$ principal submatrix of A. Note that A_B plays the role of Dirichlet boundary conditions. □

Because the above system is positive-definite, it is next solved via a Cholesky decomposition and two triangular solves. Because a Cholesky decomposition is being used, symmetry is being taken advantage of, and pivoting is not necessary.

Consider the following application of the FEMWARP Method. Figure 2.1 shows the three steps of the mesh warping process. Note that the original mesh is shown in the top graph. A rather large deformation is then applied to the boundary, and the result is shown in the middle graph. Finally, the interior nodes are moved to new positions as is shown in the bottom graph. The final mesh is a valid mesh, that is, it has no inverted elements. This figure shows that the method demonstrates

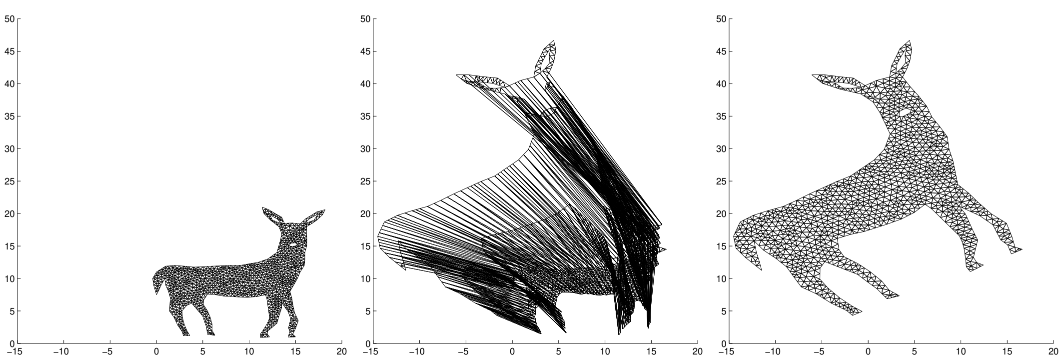

Figure 2.1: The Three Stages of the FEMWARP Algorithm [18]

promise for use as a mesh untangler in the context of the mesh warping problem.

However, the method will not always be successful. If the boundary itself becomes tangled (self-intersecting) under the deformation, then there is no possibility that FEMWARP could recover an untangled mesh. But even if the boundary is not tangled, FEMWARP may still fail as indicated by our computational experiment in Figure 2.2. Note that the shaded elements in the right box of Figure 2.2 have been inverted. In this case, the transformation applied to the boundary is too large, and it became impossible for FEMWARP to recover an untangled mesh. The same behavior is seen in LBWARP.

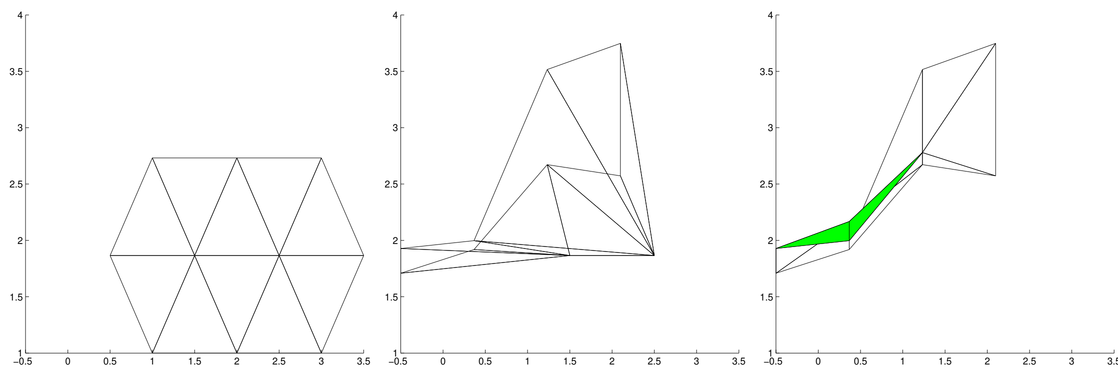

Figure 2.2: Two Patch Mesh Undergoing Large Deformation

Because FEMWARP and LBWARP are not guaranteed to work for all types of boundary transformations, we seek to determine what conditions will guarantee that the methods will be successful. Here we develop the theory of FEMWARP since it is easier to analyze. In order to meet this goal, we first prove a theorem about mesh invertibility and then prove a theorem on sufficient conditions for FEMWARP to be successful on the mesh warping problem.

2.4 Theory of Mesh Invertibility

In this section we consider sufficient conditions to guarantee that elements are not inverted in the mapped mesh. The material in this section focuses on the 2D case, but we also indicate results that extend to 3D. First, we prove a theorem about mesh invertibility in the case when a known transformation $f : \mathbf{R}^2 \to \mathbf{R}^2$ is applied to all the nodes. Then we prove a similar theorem for the case when f is applied only to the boundary and the FEMWARP algorithm is used to position interior nodes (i.e., the situation under consideration).

In order prove Theorem 4 below, we first prove the following useful lemma, which has appeared in other forms in the previous literature (e.g., [32]). This lemma extends to 3D as well if "angle" in 3D means solid angle.

Lemma 1. *Let K be a triangle with vertices $v_1, v_2, v_3 \in \mathbf{R}^2$. Let $A(K) = [v_2 - v_1; v_3 - v_1]$, and define the aspect ratio of K, $a(K)$, by $\frac{1}{a}$, where a is the minimum angle given in radians. Then $\frac{1}{4}a(K) \leq \kappa(A(K)) \leq 64a(K)$, where $\kappa(A(K))$ denotes the condition number of $A(K)$.*

Remark 1: There are several different ways to define aspect ratio. Knupp considers a framework for comparing the definitions in [28].

Sketch of Proof: First, we assume that the smallest angle of K is at v_1 and that $\overrightarrow{v_2v_1}$ is longer than $\overrightarrow{v_3v_1}$. The second step in the proof is to compute $\|A(K)\|$ and $\|A(K)^{-1}\|$ and show that the quantities are related to the area and edge lengths of K. Throughout the proof we use $\|\cdot\|_\infty$ for our choice of norm. The result is that $\|A(K)^{-1}\| \leq \frac{\sqrt{2}}{\text{area}(K)} l_{\max}$, where $l_{\max}$ is the length of the longer of the edges $\overrightarrow{v_1v_2}$

and $\overrightarrow{v_2v_3}$. Similarly, it can be shown that $\|A(K)\| \leq 2\sqrt{2}l_{\max}$. Putting this together, we obtain that $\kappa(A(K))$ is bounded above by $2\frac{l_{\max}^2}{\text{area}(K)}$. Similarly, it can be shown that $\kappa(A(K))$ can be bounded below by $\frac{1}{2\text{area}(K)}l_{\min}l_{\max}$. Recall the product of an edge with its altitude is always twice the area of K. Using the relationships between side-lengths, we can rewrite our bounds as $\frac{l_{\min}}{h_{\min}} \leq \kappa(A(K)) \leq \frac{8l_{\min}}{h_{\min}}$. Now define θ to be the angle at v_1. Because we assumed that the smallest angle is at v_1, the longest edge is $\overrightarrow{v_2v_3}$. Therefore, $\frac{l_{\min}}{h_{\min}} = (\sin\theta)^{-1}$ and $a(K) = \frac{1}{\theta}$. So, we obtain $\frac{1}{\sin\theta} \leq \kappa(A(K)) \leq \frac{8}{\sin\theta}$. Note that $0 < \theta \leq \frac{\pi}{3}$ because θ is the smallest angle. Using basic calculus, we see that for $\theta \in [0, \frac{\pi}{3}]$, $\frac{1}{\theta} \leq \frac{1}{\sin\theta} \leq \frac{2}{\theta}$. Therefore, $a(K) = \frac{1}{\theta} \leq \frac{1}{\sin\theta} \leq \kappa(A(K)) \leq \frac{8}{\sin\theta} \leq \frac{16}{\theta} = 16a(K)$, for this specific case. It is easy to show that in the more general case, the constants differ by at most a factor of 4, i.e., $\frac{1}{4}a(K) \leq \kappa(A(K)) \leq 64a(K)$, which proves the lemma.

We are now ready to prove a theorem about mesh resistance to invertibility in the case that a transformation is applied to all nodes. This theorem is stated for 2D but extends to 3D.

Theorem 4. *Suppose that $f : \mathbf{R}^2 \longmapsto \mathbf{R}^2$ is bijective and differentiable on the entire mesh, with ∇f nonsingular. Suppose f is applied to all nodes of the mesh. Then no triangle in the mesh is flipped by f if inequalities* (2.18) - (2.20) *below hold.*

Proof. For simplicity, assume for the rest of this section that $\det(\nabla f) > 0$ on the whole mesh since the case $\det(\nabla f) < 0$ is handled symmetrically. We give sufficient conditions for the noninvertibility of the mesh elements. Let K be a triangle in the mesh with vertices v_1, v_2, and v_3.

Note that K is flipped by f if and only if

$$\frac{\begin{vmatrix} (v_2 - v_1)_x & (v_2 - v_1)_y \\ (v_3 - v_1)_x & (v_3 - v_1)_y \end{vmatrix}}{\begin{vmatrix} (f(v_2) - f(v_1))_x & (f(v_2) - f(v_1))_y \\ (f(v_3) - f(v_1))_x & (f(v_3) - f(v_1))_y \end{vmatrix}} < 0. \tag{2.16}$$

Using Taylor's Theorem, we obtain:

$$\begin{aligned} f(v_2) &= f(v_1) + \nabla f(v_1)(v) \\ &+ \frac{1}{2} v^T \begin{pmatrix} \frac{\partial^2 f}{\partial x^2}(\epsilon, \mu) & \frac{\partial^2 f}{\partial x \partial y}(\epsilon, \mu) \\ \frac{\partial^2 f}{\partial x \partial y}(\epsilon, \mu) & \frac{\partial^2 f}{\partial y^2}(\epsilon, \mu) \end{pmatrix} v, \end{aligned}$$

for some (ϵ,μ) lying on the line connecting v_1 and v_2. Here $v = \begin{bmatrix} (v_2 - v_1)_x \\ (v_2 - v_1)_y \end{bmatrix}$.

Substituting the above into (2.16) we obtain an inequality of the following form giving necessary and sufficient conditions for invertibility:

$$\frac{\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}}{|\nabla f(v_1)| \begin{vmatrix} a_{11} + \gamma_1 & a_{12} + \gamma_2 \\ a_{21} + \gamma_3 & a_{22} + \gamma_4 \end{vmatrix}} < 0 \tag{2.17}$$

So, Lemma 2.7.1 in [20] applies to (2.17) once we have sufficient conditions for the hypotheses to hold. Note that the portion of the lemma we used states that if

A is invertible, if $M = A + \sigma A$ satisfies $\|\sigma A\| \leq \varepsilon \|A\|$, and if $\varepsilon\kappa(A) < 1$, then M is nonsingular. Here $\kappa(A)$ denotes the condition number of A.

Using algebra and Lemma 1 above, we find that the following conditions ensure the hypotheses of the above lemma are satisfied:

$$\frac{1}{|\nabla f(v_1)|}\left|\frac{\partial^2 f_1}{\partial x^2}(\epsilon,\mu)\right| \leq \frac{2}{195}\frac{h_{\min}}{a(K)h_{\max}^2} \tag{2.18}$$

$$\frac{1}{|\nabla f(v_1)|}\left|\frac{\partial^2 f_1}{\partial x \partial y}(\epsilon,\mu)\right| \leq \frac{1}{195}\frac{h_{\min}}{a(K)h_{\max}^2} \tag{2.19}$$

$$\frac{1}{|\nabla f(v_1)|}\left|\frac{\partial^2 f_1}{\partial y^2}(\epsilon,\mu)\right| \leq \frac{2}{195}\frac{h_{\min}}{a(K)h_{\max}^2}, \tag{2.20}$$

where $h_{\min}$ and $h_{\max}$ are the smaller and larger altitudes (respectively) from v_2 and v_3. Similar conditions are obtained for f_2. □

Note that this proof generalizes to higher dimensions and is thus applicable to unstructured mesh generation in 3D.

Next, we prove a theorem for FEMWARP, which is formally reminiscent of Theorem 4.

Theorem 5. *Suppose that $f : \mathbf{R}^2 \longmapsto \mathbf{R}^2$ is bijective and differentiable on the entire mesh, with $\nabla f(\mathbf{x})$ nonsingular. Let F be a conformal mapping from the unit disk, D, to the meshed region, R. In addition, assume that there exist positive constants α, β, γ, and δ such that $\|\nabla F(\mathbf{x})\|_2 \leq \alpha$, $\|\nabla^2 f(\mathbf{x})\|_2 \leq \beta$, $\|\nabla F(\mathbf{x})^{-1}\|_\infty \leq \gamma$, and $\|\nabla f(\mathbf{x})^{-1}\|_\infty \leq \frac{1}{2\delta}$. Consider transforming the mesh by applying f only to the boundary nodes of the triangulation, and then using FEMWARP to position the interior nodes. Then no triangles are flipped provided that $20\alpha\beta\gamma h + O(h^2) \leq \delta h_{\min}$,* *where h and $h_{\min}$ are the maximum edge-length and minimum altitude, respectively, of any triangle in the quasi-uniform mesh.*

Remark 1: The condition in Theorem 5 is analogous to the hypotheses in Theorem 4, namely, a condition for invertibility that relates the first and second derivatives of f. There is a qualitative difference, however, between the two theorems. Theorem 4 states that as long as f is bijective, then for a sufficiently refined mesh (with bounded aspect ratio), there will be no inversions. On the other hand, Theorem 5 indicates that the condition for occurrence of inversions in the FEMWARP algorithm depends on f and the mesh aspect ratio but not so much on the level of refinement.

Remark 2: The assumptions on F are unrealistic in many cases; however, the theorem gives an idea of the important factors controlling inversion of mesh elements. They are element quality and bounding the size of the second derivative in terms of the size of the first derivative.

Proof. In considering how to prove the theorem, let us fix a particular triangle K in the original mesh, centered at (x_c, y_c) with vertices v_1, v_2, v_3. As noted above, the positions of the nodes in the updated mesh are obtained from solving $A_I[\hat{x}_I,\ \hat{y}_I] = -A_B[\hat{x}_B,\ \hat{y}_B]$, for $[\hat{x}_I,\ \hat{y}_I]$ given that $[\hat{x}_B,\ \hat{y}_B]$ are obtained from the original $[x_B,\ y_B]$. In fact, each two-element row of $[\hat{x}_B,\ \hat{y}_B]$ is f applied to the corresponding row of $[x_B,\ y_B]$. Let W_K be the $2 \times m$ matrix with 1's in the $(1, v_2)$ and $(2, v_3)$ positions and -1's in the $(1, v_1)$ and $(2, v_1)$ positions. Then $W_K[\hat{x}_I,\ \hat{y}_I]$ is the 2×2 matrix, call it $A(\hat{K})$, whose determinant must be positive to avoid inversion.

Thus,

$$A(\hat{K}) = -W_K A_I^{-1} A_B [\hat{x}_B,\ \hat{y}_B] = -W_K A_I^{-1} A_B \begin{bmatrix} f(x_{B1}, y_{B1}) \\ f(x_{B2}, y_{B2}) \\ \vdots \\ f(x_{Bb}, y_{Bb}) \end{bmatrix}, \qquad (2.21)$$

where $(x_{B1}, y_{B1}), \ldots, (x_{Bb}, y_{Bb})$ is an explicit enumeration of the nodes of $[x_B,\ y_B]$. Next, expand f as a Taylor series about (x_c, y_c). The Taylor expansion has the form

$$f(x,y) = f(x_c, y_c) + \nabla f(x_c, y_c)\left((x,y) - (x_c, y_c)\right) + \tilde{f}(x,y),$$

where $\tilde{f}(x,y) = f(x,y) - f(x_c,y_c) - \nabla f(x_c,y_c)\left((x,y) - (x_c,y_c)\right)$.

Claim. $\tilde{f}$ depends only on $\nabla^2 f$, i.e., if f, g satisfy $\nabla^2 f = \nabla^2 g$, then $\tilde{f} = \tilde{g}$.

Proof. Let $\nabla^2 f$ be given. Define $\mathbf{x} = (x,y)$ and $\mathbf{x_c} = (x_c, y_c)$. We construct f from $\nabla^2 f$ using line integrals, i.e.,

$$\begin{aligned} f(\mathbf{x}) &= f(\mathbf{x_c}) + \int_0^1 \nabla f(t\mathbf{x} + (1-t)\mathbf{x_c}) \cdot (\mathbf{x} - \mathbf{x_c}) dt \\ &= f(\mathbf{x_c}) + \nabla f(\mathbf{x_c})\,(\mathbf{x} - \mathbf{x_c}) \\ &\quad + \int_0^1 \int_0^1 \Big(\nabla^2 f(st\mathbf{x} + s(1-t)\mathbf{x_c} + (1-s)\mathbf{x_c}) \\ &\qquad\qquad \cdot (\mathbf{x} - \mathbf{x_c}) \cdot t \cdot (\mathbf{x} - \mathbf{x_c}) \Big)\ ds\ dt. \end{aligned} \qquad (2.22)$$

From this formula, it is clear that $\tilde{f}(\mathbf{x})$ depends only upon $\nabla^2 f(\mathbf{x})$. □

We substitute the Taylor expansion into (2.21) to obtain

$$A(\hat{K}) = -\Bigg(W_K A_I^{-1} A_B \begin{bmatrix} f(x_c, y_c) + \nabla f(x_c, y_c)\left((x_{B1}, y_{B1}) - (x_c, y_c)\right) \\ f(x_c, y_c) + \nabla f(x_c, y_c)\left((x_{B2}, y_{B2}) - (x_c, y_c)\right) \\ \vdots \\ f(x_c, y_c) + \nabla f(x_c, y_c)\left((x_{Bb}, y_{Bb}) - (x_c, y_c)\right) \end{bmatrix}$$
$$+ W_K A_I^{-1} A_B \begin{bmatrix} \tilde{f}(x_{B1}, y_{B1}) \\ \tilde{f}(x_{B2}, y_{B2}) \\ \vdots \\ \tilde{f}(x_{Bb}, y_{Bb}) \end{bmatrix}\Bigg). \tag{2.23}$$

Note that the first term is an affine mapping of x_B, y_B, whereas the second term is a finite-difference on the finite element solution to $\nabla^2 u = 0$, where $u(x, y) = \tilde{f}(x, y)$ for $(x, y) \in \partial R$, where u is the perturbation and R is the meshed region.

We can further simplify (2.23) by writing the first two terms of the Taylor expansion of f as $v^T + [x,\ y]L$, where L is a 2×2 matrix and v^T is 2-vector, i.e., in the general form for an affine mapping, where L and v^T depend on the choice of triangle K. (The dependence on K is not explicitly denoted.) Thus,

$$f(x, y) = v^T + [x, y]L + \tilde{f}(x, y).$$

We substitute this into (2.23) to obtain

$$A(\hat{K}) = -W_K A_I^{-1} A_B \Bigg(e_B v^T + [x_B,\ y_B]L$$
$$+ \begin{bmatrix} \tilde{f}(x_{B1}, y_{B1}) \\ \vdots \\ \tilde{f}(x_{Bb}, y_{Bb}) \end{bmatrix}\Bigg), \tag{2.24}$$

where e_B is vector of all 1's of length b. We can analyze the sum of the first two terms on the right-hand side explicitly using the following lemma.

Lemma 2. *Suppose the boundary transformation applied to the initial mesh is an affine mapping. Then if any algorithm within the LWLS framework is used to reposition the interior nodes, the resulting mesh is the same as if the affine mapping were applied to all nodes in the initial mesh.*

This lemma extends to 3D as well.

PROOF. Continuing to use the above notation, we note that the positions of the interior nodes in the deformed mesh are given by

$$[\hat{x}_I,\ \hat{y}_I] = -A_I^{-1}A_B([x_B,\ y_B]L + e_B v^T). \tag{2.25}$$

In order to show that affine mappings yield exact results with any algorithm within the framework, we want to show that (2.25) is the same as:

$$[\hat{x}_I,\ \hat{y}_I] = [x_I,\ y_I]L + e_I v^T. \tag{2.26}$$

Observe that the equivalence of (2.25) and (2.26) would follow immediately from:

$$A_I([x_I,\ y_I]L + e_I v^T) = -A_B([x_B,\ y_B]L + e_B v^T). \tag{2.27}$$

Thus, it remains to check that (2.27) holds.

Because the weights for each interior node sum to 1, $Ae = \mathbf{0}$; i.e., $A_I e_I + A_B e_B = \mathbf{0}$. Hence $(A_I e_I + A_B e_B)v^T = \mathbf{0}$. Also, because $[x_I,\ y_I]$ and $[x_B,\ y_B]$ denote the original positions of the nodes, we know that $A_I[x_I,\ y_I] + A_B[x_B,\ y_B] = \mathbf{0}$ by (2.4). So, $(A_I[x_I,\ y_I] + A_B[x_B,\ y_B])L = \mathbf{0}$.

Putting these together, we see that

$$(A_I[x_I,\ y_I] + A_B[x_B,\ y_B])L + (A_I e_I + A_B e_I)v^T = \mathbf{0}. \tag{2.28}$$

Therefore, (2.27) holds, and the lemma is proven. □

Substituting the result of this lemma into (2.24) and using the fact that $W_K e_I = \mathbf{0}$ (by definition of W_K) yields

$$\begin{aligned} A(\hat{K}) &= W_K[x_I, y_I]L \\ &\quad - W_K A_I^{-1} A_B \begin{bmatrix} \tilde{f}(x_{B1}, y_{B1}) \\ \vdots \\ \tilde{f}(x_{Bb}, y_{Bb}) \end{bmatrix}. \end{aligned} \tag{2.29}$$

Observe that this equation is in a form analogous to the proof of Theorem 4: the matrix whose determinant we must take is written as the product of a determinant in the original mesh (namely $W_K[x_I,\ y_I]$) by the first derivative of f locally (namely, L) plus a remainder term. We now wish to argue that the linear term dominates the remainder term under suitable assumptions.

Lemma 2 allows us to partition any LWLS algorithm into an affine part and a perturbation. To do this, write $[\hat{x}_B,\ \hat{y}_B] = [\bar{x}_B,\ \bar{y}_B] + [\tilde{x}_B,\ \tilde{y}_B]$, where $[\bar{x}_B,\ \bar{y}_B]$ and $[\tilde{x}_B,\ \tilde{y}_B]$ denote the affine and perturbed parts of $[\hat{x}_B,\ \hat{y}_B]$. Similarly, $[\hat{x}_I,\ \hat{y}_I] = [\bar{x}_I,\ \bar{y}_I] + [\tilde{x}_I,\ \tilde{y}_I]$. Then $[\tilde{x}_B,\ \tilde{y}_B] = \begin{bmatrix} \tilde{f}(x_{B1}, y_{B1}) \\ \vdots \\ \tilde{f}(x_{Bb}, y_{Bb}) \end{bmatrix}$, and the equation $A_I[\tilde{x}_I,\ \tilde{y}_I] + A_B[\tilde{x}_B,\ \tilde{y}_B] = \mathbf{0}$ is a discretization of Laplace's equation on the mesh domain, in which the boundary condition is a Dirichlet condition of $\tilde{f}(x, y)$ at the boundary nodes. The quantity under consideration is

$$-W_K A_I^{-1} A_B [\tilde{x}_B,\ \tilde{y}_B].$$

Let u_h be the piecewise-linear interpolant of $\tilde{x}_I$ on the whole mesh. The analysis for $\tilde{y}_I$ is the same. Also, let $\tilde{J}_K$ denote the following scaled Jacobian on K:

$$\tilde{J}_K = \begin{bmatrix} \frac{(v_2-v_1)_x}{\chi} & \frac{(v_2-v_1)_y}{\chi} \\ \frac{(v_3-v_1)_x}{\chi} & \frac{(v_3-v_1)_y}{\chi} \end{bmatrix},$$

where χ is the maximum edge-length in K. Define m to be the m-vector of the evaluation of the finite element solution at the nodal points. Then $m(i) = u_h(v_i)$. Thus

$$\begin{aligned} -W_K A_I^{-1} A_B \tilde{x}_B &= W_K m \\ &= \begin{bmatrix} u_h(v_2) - u_h(v_1) \\ u_h(v_3) - u_h(v_1) \end{bmatrix} \\ &= \begin{bmatrix} (v_2 - v_1) \cdot \nabla u_h \\ (v_3 - v_1) \cdot \nabla u_h \end{bmatrix} \\ &= \begin{bmatrix} (v_2 - v_1)_x & (v_2 - v_1)_y \\ (v_3 - v_1)_x & (v_3 - v_1)_y \end{bmatrix} \begin{bmatrix} \frac{\partial u_h^1}{\partial x} \\ \frac{\partial u_h^2}{\partial x} \end{bmatrix} \\ &= \chi \begin{bmatrix} \frac{(v_2-v_1)_x}{\chi} & \frac{(v_2-v_1)_y}{\chi} \\ \frac{(v_3-v_1)_x}{\chi} & \frac{(v_3-v_1)_y}{\chi} \end{bmatrix} \begin{bmatrix} \frac{\partial u_h^1}{\partial x} \\ \frac{\partial u_h^2}{\partial x} \end{bmatrix} \\ &= \chi \tilde{J}_K \nabla u_h|_K. \end{aligned} \tag{2.30}$$

This demonstrates that each entry of $-W_K A_I^{-1} A_B [\tilde{x}_B,\ \tilde{y}_B]$ (by definition of W_K)

is a finite difference of two entries of $\tilde{f}$ at neighboring mesh nodes of triangle K centered at x_c, y_c.

Our goal is to determine a bound for $\|\nabla u_h|_K\|_\infty$. To this end, we write $\|\nabla u_h|_K\|_\infty = \|\nabla u|_K\|_\infty + \|(\nabla u_h - \nabla u)|_K\|_\infty$, where u is the solution to Laplace's equation on R with Dirichlet boundary condition $u(x,y) = \tilde{f}_1(x,y)$ on ∂R. Here $\tilde{f}_1$ denotes the first component of $\tilde{f}$. Similar notation will be used to denote the first component of various functions. Define $\hat{f} = \tilde{f}_1 \circ F$, where F is a conformal mapping from the unit disk, D, to the meshed region, R. Take $F(r,\theta) = (x_c, y_c)$. In order to determine an upper bound for $\|\nabla u_h|_K\|_\infty$, we determine $\hat{u}$, harmonic, such that $\hat{u} = \hat{f}$ on ∂D. Then we compute an upper bound for $\|\nabla \hat{u}\|_\infty$ from a Poisson integral analysis. Finally, we relate $\|\nabla \hat{u}\|_\infty$ to $\|\nabla u|_K\|_\infty$ via the formula $\nabla u(F(r,\theta)) = \nabla \hat{u}(r,\theta)(\nabla F(r,\theta))^{-1}$.

The Poisson integral we are interested in is

$$\hat{u}(r,\theta) = \frac{1}{2\pi}\int_0^{2\pi} \frac{(1-r^2)\hat{u}(1,\phi)}{1 - 2r\cos(\phi-\theta) + r^2} d\phi. \tag{2.31}$$

Differentiating (2.31) with respect to r and θ yields

$$\begin{aligned}
\frac{\partial \hat{u}}{\partial r} &= \frac{1}{2\pi}\int_0^{2\pi} \left(\frac{(-2r)\hat{u}(1,\phi)(1-2r\cos(\phi-\theta)+r^2)}{(1-2r\cos(\phi-\theta)+r^2)^2} - \right. \\
&\qquad \left. \frac{(1-r^2)\hat{u}(1,\phi)(-2\cos(\phi-\theta)+2r)}{(1-2r\cos(\phi-\theta)+r^2)^2} \right) d\phi, \\
\frac{\partial \hat{u}}{\partial \theta} &= \frac{1}{2\pi}\int_0^{2\pi} \frac{(1-r^2)\hat{u}(1,\phi)(-2r\sin(\phi-\theta))}{(1-2r\cos(\phi-\theta)+r^2)^2} d\phi.
\end{aligned}$$

Let d be the distance from $(1,\phi)$ to (r,θ) in polar coordinates. Then $d = (1 + r^2 -$

$2r\cos(\phi-\theta))^{\frac{1}{2}}$, and

$$\begin{aligned}\frac{\partial \hat{u}}{\partial r} &= \frac{1}{2\pi}\int_0^{2\pi}\Bigg(\frac{(-2r)\hat{u}(1,\phi)(1-2r\cos(\phi-\theta)+r^2)}{d^4} - \\ &\qquad \frac{(1-r^2)\hat{u}(1,\phi)(-2\cos(\phi-\theta)+2r)}{d^4}\Bigg)d\phi, \\ \frac{1}{r}\frac{\partial \hat{u}}{\partial \theta} &= \frac{1}{2\pi}\int_0^{2\pi}\frac{(1-r^2)\hat{u}(1,\phi)(-2\sin(\phi-\theta))}{d^4}d\phi.\end{aligned}$$

Note that $\nabla \hat{u} = \left(\frac{\partial \hat{u}}{\partial r}, \frac{1}{r}\frac{\partial \hat{u}}{\partial \theta}\right)$ in rotated coordinates.

We are interested in the case when $\hat{u}(1,\phi) = \tilde{g}(1,\phi) = (\tilde{f}_1 \circ F)(1,\phi)$. To this end, set $\hat{u}(1,\phi) = \tilde{g}(1,\phi) = \tilde{f}_1(F(1,\phi))$.

Substituting $\tilde{g}(1,\phi)$ into the appropriate integrals, we obtain

$$\begin{aligned}\frac{\partial \hat{u}}{\partial r} &= \frac{1}{2\pi}\int_0^{2\pi}\Bigg(\frac{\tilde{g}(1,\phi)((-2r)(1-2r\cos(\phi-\theta)+r^2)}{d^4} - \\ &\qquad \frac{(1-r^2)(-2\cos(\phi-\theta)+2r))}{d^4}\Bigg)d\phi \\ \frac{1}{r}\frac{\partial \hat{u}}{\partial \theta} &= \frac{1}{2\pi}\int_0^{2\pi}\frac{\tilde{g}(1,\phi)(1-r^2)(-2\sin(\phi-\theta))}{d^4}d\phi.\end{aligned}$$

Our goal is to get estimates on the size of these integrals. To this end, we obtain bounds on each term in their integrands as follows. First, we see that

$$\begin{aligned}|\tilde{g}(1,\phi)| &= |\tilde{f}_1(F(1,\phi))| \\ &\leq \max \|\nabla^2 f(\mathbf{x})\|_2 \|F(1,\phi) - \mathbf{x_c}\|_2^2 \text{ by } (2.22) \\ &\leq \max \|\nabla^2 f(\mathbf{x})\|_2 \alpha \|(1,\phi) - F^{-1}(\mathbf{x_c})\|_2^2 \\ &= \max \|\nabla^2 f(\mathbf{x})\|_2 \alpha d^2 \\ &= \alpha\beta d^2.\end{aligned}$$

Let $\tau(r,\theta)$ denote the other factor in the numerator of (2.32). Then

$$\begin{aligned}
\tau(r,\theta) &= (-2r)(1-2r\cos(\phi-\theta)+r^2)-(1-r^2)(-2\cos(\phi-\theta)+2r) \\
&= -4r+2r^2\cos(\phi-\theta)+2\cos(\phi-\theta) \\
&= -4r+(2r^2+2)\cos(\phi-\theta) \\
&= -4r+(2r^2+2)\left(1-\sin^2\left(\frac{\phi-\theta}{2}\right)\right). \qquad (2.32)
\end{aligned}$$

The last equality holds as $\cos(\phi-\theta)=\cos\left(2\frac{(\phi-\theta)}{2}\right)=1-2\sin^2\left(\frac{(\phi-\theta)}{2}\right)$. Note that (2.32) is equal to

$$2(r-1)^2-(2r^2+2)\sin^2\left(\frac{\phi-\theta}{2}\right).$$

Define $s=1-r$. Then

$$\begin{aligned}
2(r-1)^2 &= 2s^2 \\
&\leq 2d^2
\end{aligned}$$

and

$$(2r^2+2)\sin^2\left(\frac{\phi-\theta}{2}\right) \leq 4\sin^2\left(\frac{\phi-\theta}{2}\right).$$

In determining an upper bound for $\sin^2\left(\frac{\phi-\theta}{2}\right)$, there are two cases to consider. First, we show that $|\sin(\phi-\theta)|\leq d$, as this result will be used in Case 1 below. To see this, notice that

$$0\leq(r-\cos(\phi-\theta))^2.$$

Hence,

$$\sin^2(\phi-\theta)\leq\sin^2(\phi-\theta)+(r-\cos(\phi-\theta))^2=d^2.$$

Therefore,

$$|\sin(\phi-\theta)| \leq d.$$

Case 1: $|\phi-\theta| \leq \frac{\pi}{2}$

It follows from the above inequality that

$$\left|2\sin\left(\frac{\phi-\theta}{2}\right)\cos\left(\frac{\phi-\theta}{2}\right)\right| \leq d.$$

In this case, $\left|\cos\left(\frac{\phi-\theta}{2}\right)\right| \geq \cos\left(\frac{\pi}{4}\right) = \frac{\sqrt{2}}{2}$. Thus,

$$\left|\sin\left(\frac{\phi-\theta}{2}\right)\right|\frac{\sqrt{2}}{2} \leq \left|\sin\left(\frac{\phi-\theta}{2}\right)\cos\left(\frac{\phi-\theta}{2}\right)\right| \leq \frac{d}{2}.$$

This implies that

$$\left|\sin^2\left(\frac{\phi-\theta}{2}\right)\right| \leq \frac{1}{2}d^2.$$

Case 2: $|\phi-\theta| \geq \frac{\pi}{2}$

Then

$$\left|\sin\left(\frac{\phi-\theta}{2}\right)\right| \leq 1.$$

In addition, $d \geq 1$, as $\cos(\phi-\theta) \leq 0$ for $|\phi-\theta| \geq \frac{\pi}{2}$. So,

$$d \geq \left|\sin\left(\frac{\phi-\theta}{2}\right)\right|.$$

Hence,

$$d^2 \geq \left|\sin^2\left(\frac{\phi-\theta}{2}\right)\right|.$$

Putting the two cases together, we see that

$$\left|\sin^2\left(\frac{\phi-\theta}{2}\right)\right| \leq d^2.$$

Hence,

$$\left|2(r-1)^2 - (2r^2+2)\sin^2\left(\frac{\phi-\theta}{2}\right)\right| \leq 6d^2.$$

Let $\rho(r,\theta)$ denote the second term of the numerator in (2.32). Finally, we wish to determine an upper bound for $|\rho(r,\theta)|$. Observe that

$$\begin{aligned}
|\rho(r,\theta)| &= 2|\sin(\phi-\theta)|\cdot|1-r^2| \\
&= 2|\sin(\phi-\theta)|\cdot|2s-s^2| \\
&\leq 2|\sin(\phi-\theta)|\cdot|2s| \\
&\leq 2|\sin(\phi-\theta)|\cdot 2d \\
&\leq 4d^2.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\frac{\partial\hat{u}}{\partial r} &\leq \frac{1}{2\pi}\int_0^{2\pi}\frac{\max\|\nabla^2 f(\mathbf{x})\|_2\alpha d^2 6d^2}{d^4}\,d\phi = 6\alpha\beta, \\
\frac{1}{r}\frac{\partial\hat{u}}{\partial\theta} &\leq \frac{1}{2\pi}\int_0^{2\pi}\frac{\max\|\nabla^2 f(\mathbf{x})\|_2\alpha d^2 4d^2}{d^4}\,d\phi = 4\alpha\beta.
\end{aligned}$$

We conclude that $\|\nabla\hat{u}(r,\theta)\|_\infty \leq 6\alpha\beta$.

For a sufficiently refined mesh containing piecewise-linear elements, we note that

$$\|u-u_h\|_{W^1_\infty(\Omega)} \leq Ch\|u\|_{W^2_\infty(\Omega)} = O(h),$$

according to Corollary 7.1.12 in [6]. Thus, it suffices to approximate $\|\nabla u_h|_K\|_\infty$ by $\|\nabla u|_K\|_\infty$.

Finally, we relate $\nabla \hat{u}(r,\theta)$, where $(r,\theta) = F^{-1}(\mathbf{x_c})$, to $\nabla u(\mathbf{x_c})$ via the formula $\nabla u(\mathbf{x_c}) = \nabla \hat{u}(r,\theta)(\nabla F(r,\theta))^{-1}$. This yields

$$\begin{aligned} \|\nabla u_h|_K(\mathbf{x_c})\|_\infty &= \|\nabla u|_K(\mathbf{x_c})\|_\infty + O(h) \\ &= \|\nabla \hat{u}(r,\theta)(\nabla F(r,\theta))^{-1}\|_\infty + O(h) \\ &\leq 6\alpha\beta\|\nabla F(r,\theta)^{-1}\|_\infty + O(h) \\ &\leq 6\alpha\beta\gamma + O(h). \end{aligned} \tag{2.33}$$

Thus, we conclude that

$$\begin{aligned} \|W_K A_I^{-1} A_B \tilde{x}_B\|_\infty &= \|\chi \tilde{J}_K \nabla u_h|_K\|_\infty \\ &\leq \chi\|\tilde{J}_K\|_\infty (6\alpha\beta\gamma + O(h)) \\ &\leq 12\alpha\beta\gamma h + O(h^2). \end{aligned} \tag{2.34}$$

Similarly,

$$\|W_K A_I^{-1} A_B[\tilde{x}_B,\ \tilde{y}_B]\|_\infty \leq 20\alpha\beta\gamma h + O(h^2). \tag{2.35}$$

Therefore, the overall contribution from the remainder term of (2.29) is $O(h)$. We now concern ourselves with the first term in (2.29). By an argument similar to that shown in (2.30),

$$W_K[x_I,\ y_I]L = \chi \tilde{J}_K \nabla f(x_c, y_c)^T.$$

Thus, this term is $O(h)$, where the constant depends upon the first derivative of f. In order to finish the proof, we note that $\det(A(\hat{K})) > 0$ whenever

$$\begin{aligned} \left\|W_K A_I^{-1} A_B[\tilde{x}_B,\ \tilde{y}_B]\right\|_\infty &\leq \frac{\|\chi \tilde{J}_K \nabla f(x_c, y_c)^T\|}{\kappa_\infty(\chi \tilde{J}_K \nabla f(x_c, y_c)^T)} \\ &= \frac{1}{\|(\chi \tilde{J}_K \nabla f(x_c, y_c)^T)^{-1}\|_\infty}, \end{aligned}$$

according to Lemma 2.7.1 in [20]. Thus, it remains to find an upper bound for $\|(\chi \tilde{J}_K \nabla f(x_c, y_c)^T)^{-1}\|_\infty$. It is easy to show that

$$\begin{aligned} \|(\chi \tilde{J}_K \nabla f(x_c, y_c)^T)^{-1}\|_\infty &\leq \frac{1}{\text{area}(K)} \|\nabla f(x_c, y_c)^{-T}\|_\infty \chi \\ &= \frac{2}{\epsilon} \|\nabla f(x_c, y_c)^{-1}\|_\infty \\ &\leq \frac{1}{\delta \epsilon}, \end{aligned} \tag{2.36}$$

where ϵ is the minimum altitude in triangle K, as $2 \text{ area}(K) = \chi \cdot \epsilon$.

Combining (2.34) and (2.36), we see that no triangles in the mesh are flipped provided that $20\alpha\beta\gamma h + O(h^2) \leq \delta h_{\min}$. □

At this time, we have not proven a theorem like Theorem 5 for the LBWARP method. However, in this section, we prove a theorem which gives a lower bound for the LBWARP weights. In particular, we show that no weight is too close to zero as long as the mesh has bounded aspect ratio. This result will be important in contexts where positive weights are required; one such example is the setup in Section 2.6.

Theorem 6. *There is a positive-valued function $q(x)$ such that $w_{ij} \geq q(a)$, where w_{ij} is any weight computed by the LBWARP algorithm and a is the worst (largest) aspect ratio in the mesh.*

PROOF. First, for this proof we need the following claim.

Claim. Let $p = (1, 0)$ be an interior node in a bounded-aspect ratio triangulation, and suppose that all of its neighboring nodes have nonnegative x-coordinates. Let $x_{\max}$ be the maximum x-coordinate among neighbors of p. Then $x_{\max} \leq q_0(a)$, where q_0 is a fixed function and a is the mesh aspect ratio.

The proof of claim is omitted, but we give a sketch of its main idea here. Consider the triangle t_1 containing p and also the line segment starting at p and proceeding in the negative x direction. Since no neighbor of p has a negative x-coordinate, we conclude that this triangle t_1 has an altitude at most 1. Therefore, its maximum side-length is at most a, using the following definition of the aspect ratio: $AR = \frac{l_{\max}}{alt_{\min}}$, where $l_{\max}$ and $alt_{\min}$ denote the maximum side-length and minimum altitude. The triangle immediately adjacent to p, possessing one side of length at most a, therefore has maximum side-length at most a^2, and so on. The number of triangles adjacent to p is bounded by ca, where c is a universal constant, since the minimum angle of each triangle is at least $1/(c'a)$, using the alternative definition of the aspect ratio, $AR = 1/\theta_{\min}$, where $\theta_{\min}$ is the minimum angle in the triangle. (Note that the two definitions of aspect ratio are equivalent up to a constant.) Therefore, the longest side-length among neighbors of p is a^{ca}, which is the function q_0. A similar argument appears in [33].

Once the claim is established, we can return to the proof of the theorem, which is as follows. Fix an interior point p in the initial mesh with n neighbors, and let the optimal weights computed by the LBWARP algorithm be $w_1, \ldots, w_n$. (Omit the second subscript since p is fixed throughout the proof.) Let the neighbors of p be $(x_1, y_1), \ldots, (x_n, y_n)$, and let p itself be located at (x^p, y^p). Since the LBWARP problem is a strictly convex optimization problem on a bounded convex set, the Lagrange multiplier conditions are necessary and sufficient for optimality [35]. These conditions are as follows. There exist three multipliers λ, μ, ν such that

$$\begin{aligned} w_1 x_1 + \cdots + w_n x_n &= x^p, \\ w_1 y_1 + \cdots + w_n y_n &= y^p, \\ w_1 + \cdots + w_n &= 1, \\ \lambda x_1 + \mu y_1 + \nu &= 1/w_1, \\ &\vdots \\ \lambda x_n + \mu y_n + \nu &= 1/w_n. \end{aligned}$$

First, let us consider a very simple case that $\lambda = \mu = 0$. In this case, each w_i is equal to $1/\nu$. Since the sum of the w_i's is 1, this means in fact that $1/\nu = 1/n$ and hence $w_i \geq 1/n$ for all i. Furthermore, $1/n > (1/(ca))$ using the argument about the number of neighbors from the proof of the claim. This shows that $w_i > q(a)$ in this special case.

The generic case is that at least one of λ, μ is nonzero. Without loss of generality, we can assume $\lambda = 1$, $\mu = \nu = 0$. The reason this assumption is WLOG is as follows. We can apply an arbitrary transformation of the form

$$(x, y) \mapsto (x, y) \begin{pmatrix} u & v \\ -v & u \end{pmatrix} + (k_1, k_2)$$

(with at least one of u, v nonzero) uniformly to the original data $(x_1, y_1), \ldots, (x_n, y_n)$, (x^p, y^p); applying such a transformation has the effect of transforming λ, μ, ν arbitrarily as long as not both λ, μ are zero but does not change any of the aspect ratios in the original triangulation (since the above transformation is a combination of a rigid motion, a translation, and a uniform scaling).

After this transformation is applied, we can further apply a uniform translation to all the y-coordinates in the original data to force $y^p = 0$. Since $\lambda = 1$ and $\mu = \nu = 0$, we conclude from the KKT system that $x_i = 1/w_i$ so that $w_i x_i = 1$. This means that $w_1 x_1 + \cdots + w_n x_n = n$, hence $x^p = n$ so $p = (n, 0)$. Also, the relationship $x_i = 1/w_i$ means that all the x_i's are positive. Finally, the relationship $x_i = 1/w_i$ means that $\min w_i = 1/\max x_i$. In particular, in order to have a very small w_i means that there must be a very large x_i. It follows from the claim that $\max x_i \leq nq_0(a)$, hence $\min w_i \geq 1/(nq_0(a))$. Therefore, there exists a positive-valued function $q(x)$ such that all the weights are at least $q(a)$. $\square$

2.5 Theory of the Resulting Mesh

Next, we study the resulting mesh from LBWARP and compare it with the converged meshes obtained from the local weighted Laplacian smoothing and Gauss-Seidel algorithms.

Theorem 7. *The resulting mesh from the LBWARP algorithm is the same as the converged meshes from the weighted Laplacian smoothing method and the Gauss-Seidel method taken one pass at a time.*

Proof. First, we note that as proved in Theorem 1, unique positions result when using LBWARP to solve for positions of the interior nodes. Throughout this discussion we assume that the input transformation has already been applied to the boundary.

Now, we discuss the convergence of the Gauss-Seidel algorithm when applied to $A_I[\hat{x}_I,\ \hat{y}_I] = -A_B[\hat{x}_B,\ \hat{y}_B]$. It is well-known that this algorithm is guaranteed to converge for an M-matrix. (See [48] for the details.) Thus, according to Theorem 2, it will converge when applied to this system.

Next, it is easy to show the iterates produced by applying the local version of weighted Laplacian smoothing to the above linear system are the same as those obtained by applying the Gauss-Seidel algorithm. The details have been omitted.

Thus, we can compute the iterates for the local version of weighted Laplacian smoothing as follows:

$$[x_I,\ y_I]^{(k+1)} = C[x_I,\ y_I]^{(k)} + F[\hat{x}_B,\ \hat{y}_B],$$

where $C = (M_G^{-1}N_G), F = (-M_G^{-1}A_B), M_G = D + L,\ D = \text{diag}(A_I)$ and $L = \text{tril}(A_I)$, $N_G = -\text{triu}(A_I)$, and $[x_I,\ y_I]^{(0)} = [x_I,\ y_I]$.

Finally, we show that the local version of weighted Laplacian smoothing converges to the same point as LBWARP. To this end, we consider the convergence of the local version of the weighted Laplacian smoothing method. In order to study its convergence, we need to concern ourselves with

$$\lim_{k\to\infty}\left(\begin{pmatrix} C & F \\ 0 & I \end{pmatrix}^k\right). \tag{2.37}$$

Note that (2.37) is equal to

$$\begin{pmatrix} \lim_{k\to\infty} C^k & \left(\sum_{j=0}^{\infty} C^j\right) F \\ 0 & I \end{pmatrix}. \tag{2.38}$$

Now, observe that $\left(\sum_{j=0}^{\infty} C^j\right) F = (I - C)^{-1}F$. In addition, observe that A_I and $-A_B$ can be obtained from $I - C$ and F via elementary row operations since $I - C = I - M_G^{-1}N_G$ and $F = -M_G^{-1}A_B$. Thus, the desired matrices are obtained by left multiplying by M_G. This proves that $(I - C)^{-1}F = -A_I^{-1}A_B$.

The final observation to make is that the eigenvalues of C satisfy $\rho(\lambda) < 1$, which can be seen as follows. First, we note that all of the entries of C are nonnegative by definition of C. Second, we note that each row sum is less than or equal to 1 since each row is a convex combination of rows with nonnegative entries and row sums that are less than or equal to 1. In addition, there is at least one row with row sum strictly less than 1 since there is at least one interior node with a boundary neighbor. Thus, the eigenvalues of C are less than 1 in absolute value, i.e., $\lim_{k\to\infty} C^k = 0$.

Putting everything together, we see that (2.38) is equal to

$$\begin{pmatrix} 0 & -A_I^{-1}A_B \\ 0 & I \end{pmatrix}. \tag{2.39}$$

This proves the theorem, since the solution to the second linear system, $A_I[\hat{x}_I,\ \hat{y}_I] = -A_B[\hat{x}_B,\ \hat{y}_B]$, can be written as follows:

$$\begin{pmatrix} \hat{x}_I & \hat{y}_I \\ \hat{x}_B & \hat{y}_B \end{pmatrix} = \begin{pmatrix} 0 & -A_I^{-1}A_B \\ 0 & I \end{pmatrix} \begin{pmatrix} x_I & y_I \\ \hat{x}_B & \hat{y}_B \end{pmatrix}.$$

□

This result also holds for the FEMWARP algorithm, since the local weighted laplacian smoothing and Gauss-Seidel algorithms both converge for symmetric positive-definite matrices. We note that this result also holds for other algorithms within the LWLS framework whenever the local version of weighted laplacian smoothing and the Gauss-Seidel methods converge.

2.6 Numerical Tests

In order to test the robustness of LBWARP and FEMWARP, we designed three numerical tests each based upon a series of mesh deformations. These tests have been designed to mimic the basic elements of a beating heart's motion, namely compression and twisting. We study the heart's motion in Section 2.7. All of the tests involve the deformation of an annulus on each timestep. The annulus is composed of four, equally-spaced concentric rings of triangles. Its inner radius is 1, and its outer radius is 10. The initial annulus mesh is shown in Figure 2.3. For the annulus tests, we only discuss the results from the FEMWARP method, as the results obtained from LBWARP are similar.

In the first test, the inner circle was deformed via an outward translation on each iteration. In particular, its radius was increased by 0.5 each timestep. Figure 2.4 shows how the annulus mesh changes over time when FEMWARP was used to warp the mesh. The test showed that when the inner radius of the annulus became 5, too much deformation had occurred, since the mesh had inverted, and the minimum mean ratio of a triangle in the mesh, denoted by γ min, was less than 1. (See [15]

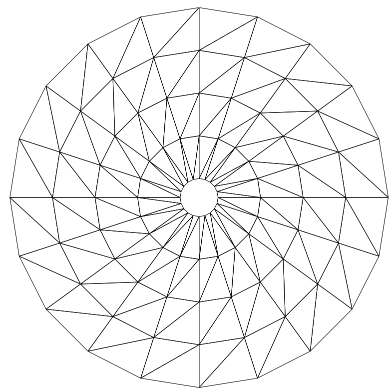

Figure 2.3: Initial Annulus Mesh

for the mean ratio definition.) However, as this test demonstrates, FEMWARP tolerates a large amount of compression is tolerated before inversion occurs.

Rotation of the inner circle occurs in the second deformation series. On each iteration, the inner circle was rotated counterclockwise 10 degrees. Figure 2.5 shows how the annulus changes over time when FEMWARP was used to warp the mesh. In each of the figures, the number of degrees the annulus has been rotated is given along with the minimum mean ratio of the triangles in the mesh. Notice that as the inner circle rotates counterclockwise, the quality of the triangles in the inner ring decreases. The mesh remained untangled until it had been rotated 70 degrees.

The goal of the third test was to combine translation and rotation of the inner circle. At each timestep, the motions of the first two tests were combined, i.e., the radius of the inner circle was increased by 0.5, and it was rotated by 10 degrees. Figure 2.6 shows the results when FEMWARP was used for the mesh warping. The new radius for the inner circle, its amount of rotation, and the minimum mean ratio

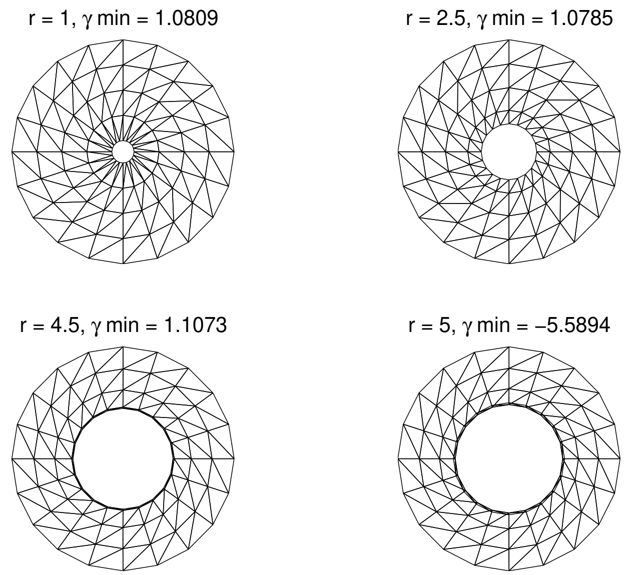

Figure 2.4: Test 1: Translation of Inner Circle of Annulus by FEMWARP

of the triangles are given. Observe that the mesh remains untangled until the radius of the inner circle is 3.5 and the circle has been rotated by 50 degrees.

From these three tests, we conclude that FEMWARP can withstand relatively large amounts of compression and twisting while resisting inversion. However, whenever the deformations either tangle the boundary or are too large, FEMWARP fails because inversion of elements occurs. The same conclusions hold for the LBWARP tests.

In order to compare the LBWARP and FEMWARP methods, we chose specific nonlinear boundary deformations in order to determine how much deformation each test mesh could withstand when warped according to each method. The nonlinear deformations we used can be written as the sum of an affine mapping plus

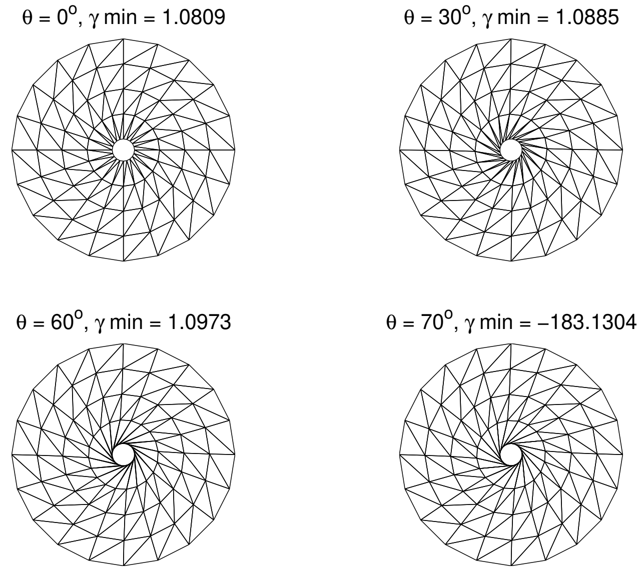

Figure 2.5: Test 2: Rotation of Inner Circle of Annulus by FEMWARP

a parameter-controlled nonlinear term. In 2D, the boundary deformation used for the tests was:

$$\begin{pmatrix} x \\ y \end{pmatrix} \longmapsto \begin{pmatrix} 2 & -1 \\ -2 & 5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \alpha \begin{pmatrix} 0.2x^2 + 0.1xy + \sin(x-\pi) \\ 0.5y^2 + 5\sin(y)\cos(x) \end{pmatrix},$$

whereas in 3D, it was:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \longmapsto \begin{pmatrix} 2 & -1 & 0 \\ -2 & 5 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \alpha \begin{pmatrix} 0.1xy \\ 0.5yz \\ 0.1x^2 \end{pmatrix}.$$

Initially, the parameter, α, controlling the nonlinear term in each boundary deformation was set to zero. Thus, the initial warping of the mesh was done in an affine manner, which we know yields a valid mesh according to Lemma 2. The value of α was then increased, whereby increasing the amount of deformation to the mesh.

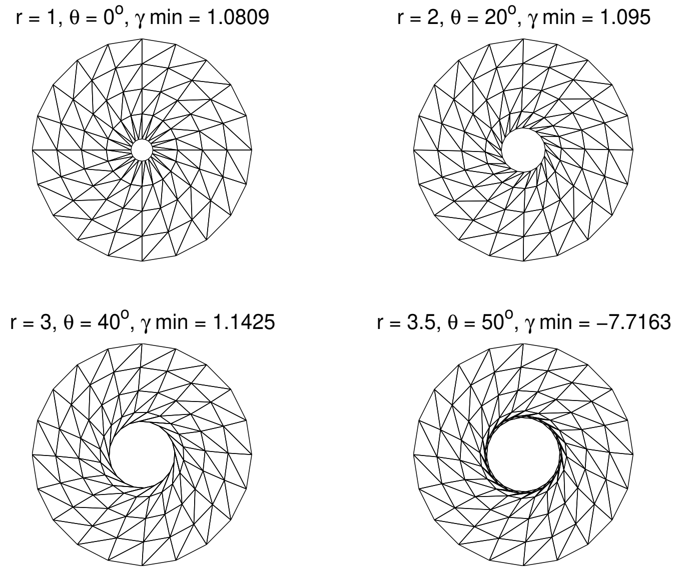

Figure 2.6: Test 3: Translation and Rotation of Inner Circle of Annulus by FEMWARP

When α became large enough, inversion eventually occurred, and its value, which we denote by $\alpha_{\max}$, was recorded at that time. Table 2.1 and Table 2.2 give the results obtained from warping various 2D and 3D meshes, respectively, according to these boundary deformations.

These results demonstrate that in 2D, LBWARP and FEMWARP often perform identically. However, it is sometimes the case that one method outperforms the other. In Table 2.1, we see that FEMWARP did better than LBWARP on the arc mesh. (We have also seen computational results that indicate that LBWARP performs better than FEMWARP at times, although those results are not included here since they were for a different boundary motion.) In 3D, the two methods often perform differently as seen in Table 2.2. For the chosen deformation, FEMWARP

Table 2.1: Values of α_{max} for LBWARP and FEMWARP on a 2D Problem [18]

2D mesh	**LBWARP**	**FEMWARP**
arc	0.2	0.9
cylinder	2	2
deer	2.6	2.6
rand400	1.8	1.8
rand1000	1.8	1.8

Table 2.2: Values of α_{max} for LBWARP and FEMWARP on a 3D Problem [29]

3D mesh	**LBWARP**	**FEMWARP**
distduct	0.005	0.022
foam	0.3	0.7
gear	1.8	3.5
hook	0.08	0.16

always outperformed LBWARP. Table 2.3 and Table 2.4 show the percentage of negative weights computed by FEMWARP for each of the test meshes. From these tables, we see that it is when the FEMWARP method computes a high percentage of negative weights that the performance of the two methods differs the most, as FEMWARP is extrapolating nodal positions under these circumstances. Counterintuitively, FEMWARP performed better than LBWARP for meshes with high percentages of negative weights.

Table 2.3: Percentages of Negative Weights Computed by FEMWARP on 2D Meshes [18]

2D mesh	**Percent Negative Weights**
arc	29.5
cylinder	0.37
deer	0.11
rand400	0
rand1000	0

Table 2.4: Percentages of Negative Weights Computed by FEMWARP on 3D Meshes [29]

3D mesh	**Percent Negative Weights**
distduct	21.8
foam	21.8
gear	24.8
hook	23.6

Finally, we wish to compare FEMWARP with a linear elasticity solver. Johnson and Tezduyar developed a linear elasticity solver for mesh warping in [24]; it fits into the LWLS framework. As was eluded to in Section 2.3, FEMWARP can often handle more deformation than a linear elasticity solver. Table 2.5 compares the performance of FEMWARP and the linear elasticity solver on the previous set of parameter-controlled nonlinear boundary deformations tests in 2D. For this experiment, Poisson's ratio, ν, was set to 0.3 in the linear elasticity solver.

Table 2.5: Values of α_{max} for FEMWARP and a Linear Elasticity Solver on a 2D Problem [18]

2D mesh	**FEMWARP**	**Linear Elasticity**
arc	0.9	0.7
cylinder	2	1.2
deer	2.6	2.4
rand400	1.8	1.3
rand1000	1.8	1.3

2.7 Application to Cardiology

We now use FEMWARP in order to study the movement of the beating canine heart under normal conditions. To do this, we obtained data from the Laboratory of Cardiac Energetics at the National Institutes of Health (NIH) [31]. We were given (x, y, z, t) data for 192 points on the inner surface of the left and right ventricles of the beating canine heart from a physiological pacing experiment. The data frames are 14.6 milliseconds apart with the first frame occurring 12 milliseconds before the pacing spike.

The first step in the simulation of the ventricular movement was to generate a mesh for the initial position of the ventricles. In order to do this, we first noted that the 192 points we were given were arranged in 8 slices with 24 points each. Thus, in order to generate the initial mesh, we decided to create a mesh for the top slice and then use FEMWARP to do the mesh warping necessary to create meshes for the remaining slices. Note that this uses FEMWARP in one and two dimensions as we describe in detail below. Once we have the meshes for all of the levels, we connect the triangular meshes into a tetrahedral mesh for the ventricles. The procedure to do this is also described below.

We now give a more detailed description of the method we used to create the initial mesh of the canine ventricles. We first used the two-dimensional quality mesh generation package called Triangle [44] to generate an initial mesh of the top slice (after projecting it into the x-y plane). Note that this yielded a good-quality mesh in the x-y plane with several additional nodes. Second, we computed the z-coordinates for the new points on the boundary of the top slice using 1D FEMWARP. Third, we used the weight-finding portion of our FEMWARP algorithm to compute the weights for the appropriate 2D linear system obtained from the x- and y-coordinates. Fourth, we determined the z-coordinates for the mesh of the top slice by forcing the z-coordinates to satisfy the appropriate 3D linear system using the 2D weights. At this point, we had the mesh for the top slice.

Our second task was to generate meshes for each of the remaining 7 slices. This was done using our FEMWARP algorithm to warp the mesh for the top slice into meshes for each of the remaining slices. In order to accomplish this, the first step was to determine the coordinates of the additional boundary nodes for the mesh of the appropriate slice. This was done using 1D FEMWARP. Then, the (x, y) coordinates of the interior nodes of that slice were determined using 2D FEMWARP. The z-coordinates for the interior nodes were found by forcing them to satisfy the appropriate 3D linear system using these weights.

The third step was to connect the triangular meshes for each of the 8 slices into one tetrahedral mesh for the canine ventricles. To do this, the corresponding triangles between two slices were connected to form a triangular prism. After a temporary mesh of triangular prisms was created, the triangular prisms were subdivided into tetrahedra using the method outlined in [9].

After the initial tetrahedral mesh was created, we checked its quality using the mean ratio mesh quality metric. Using this test, it was determined that the initial mesh was of poor quality. This was because the 8 slices were equally-spaced even though the curvature of the ventricles changes much more rapidly near the bottom of the heart. Thus, we decided to use linear interpolation in conjunction with FEMWARP in the obvious way in order to add two additional slices of nodes near the bottom of the heart. Because the curvature of the ventricles changes more rapidly near the bottom of the heart, the first additional slice was placed halfway between levels 7 and 8, and the second additional slice was placed halfway between the first new level and level 8. The resulting initial mesh is shown in Figure 2.7.

After the initial mesh was created, the heart data was used to move the 192 data points on the boundary of the mesh to their new positions The same process

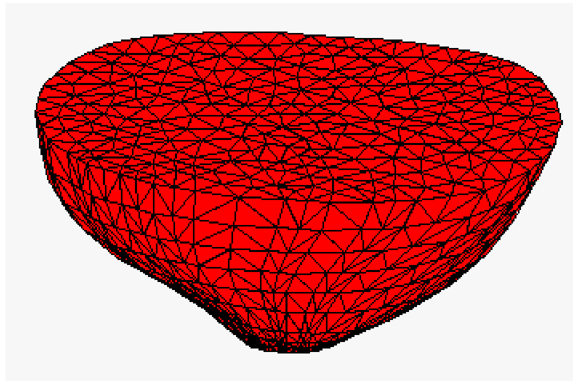

Figure 2.7: Initial Canine Ventricular Mesh

as above (i.e., FEMWARP in 1D, FEMWARP in 2D, and using 1D FEMWARP to add the two new levels of data near the bottom) was used in order to reposition the remaining boundary points. Once the boundary nodes were relocated to their positions for timestep $t = 2$, the 3D version of FEMWARP was used to move the interior nodes to their new positions for this timestep. This process was performed iteratively in order to study the movement of the heart at timesteps $t = 3, \ldots, 32$.

The simulation of the canine ventricular movement produced a series of meshes that show the ventricles twisting, expanding, and then contracting over the cardiac cycle. This is consistent with what occurs in nature. Because the dynamic range of the motion is small, it cannot be detected in single figures (separate from an animation).

During each timestep, the mean ratios of the tetrahedra in the mesh were computed. The mean ratio computations showed that the heart remained untangled throughout the entire simulation. This is not very surprising since the heart mesh is composed of elliptical rings that seem to undergo less movement on each timestep than the circular rings in the test cases. However, the motion of the heart is anisotropic which makes it difficult to predict in advance how it will tolerate deformations. Interestingly enough, the values of the minimum and average mean ratios were relatively constant across all timesteps. Only the value of the maximum mean ratio changed a significant amount. The mean ratio computations are also a good indication that the heart meshes are of sufficiently good quality for use with a numerical PDE solver that requires moving meshes.

In order to further test our mesh warping algorithm, the motion of the ventricles was exaggerated by a factor of 3. In this case, the motion was large enough to detect in separate figures and is shown in Figure 2.8. We note that the FEMWARP algorithm also performed successfully in this case, which is encouraging given the much larger deformations.

2.8 Conclusions

In summary, we developed a linear weighted laplacian smoothing (LWLS) framework for warping tetrahedral meshes. The first step in our framework is to determine a set of local weights for each interior node. Second, a user-supplied deformation is applied to the boundary notes. The third and final step is to solve a system of linear equations based upon the weights and the new positions of the boundary nodes to determine the final positions of the interior nodes.

Within this framework, we presented two methods: LBWARP and FEMWARP

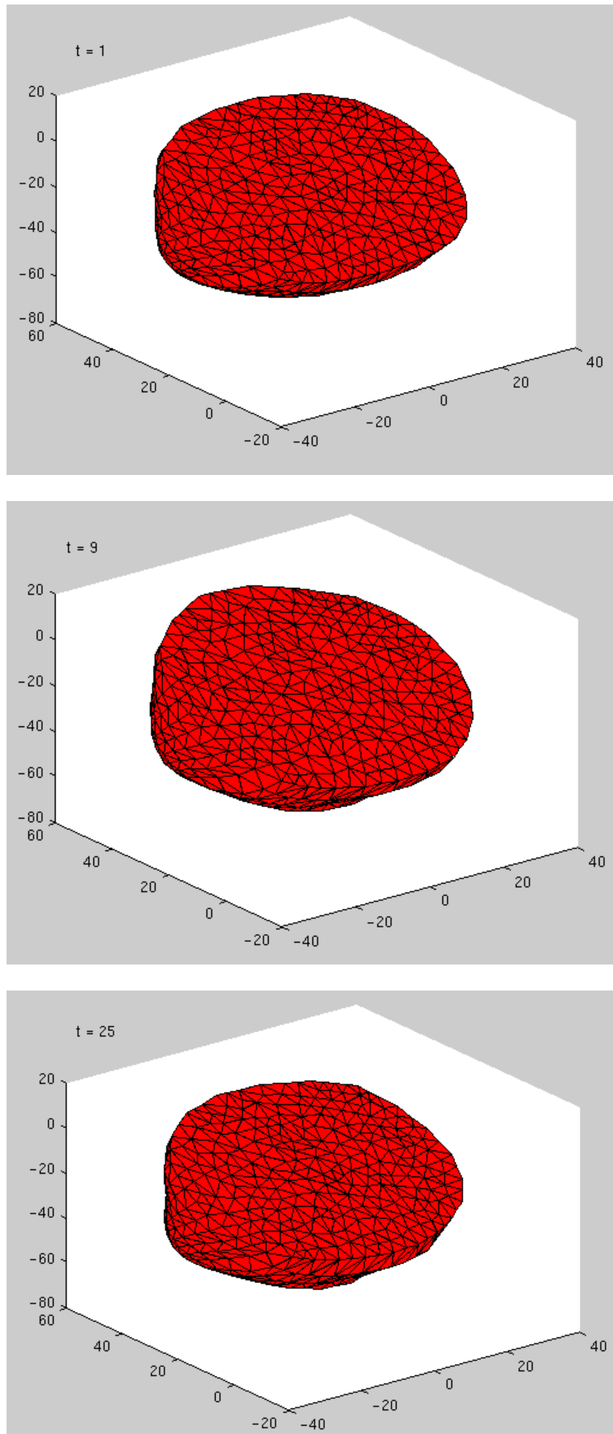

Figure 2.8: Exaggerated Movement of the Heart as it Beats: Motion for Timesteps $t = 1, 9$, and 25

based on a log-barrier method and the finite element method, respectively. Both of these methods are simple, preserve the mesh's combinatorial structure, are computationally inexpensive, and provide some theoretical guarantees for use on the mesh warping problem.

There are two main advantages to methods within the LWLS framework. First, if a continuous boundary deformation is given, then methods within the framework are valid for computing the resulting trajectory that specifies the movement of the interior nodes. In addition, these trajectories will be continuous, which is vital for applications where continuity of motion is required. Second, sparse matrix algorithms may be used to solve the linear system which determines the final positions of the interior nodes.

We proved a theorem regarding mesh invertibility and a lemma showing that all LWLS methods yield exact results for affine boundary transformations. In addition, we proved a theorem, within the context of FEMWARP, giving sufficient conditions for the mesh to resist inversion for more general mappings.

In addition, we proved a theorem that gave a lower bound for the LBWARP weights. We also showed that the resulting mesh from any of the LWLS methods is the same as the converged mesh obtained from a local version of the algorithm and from the Gauss-Seidel method whenever the latter two methods converges.

We tested the robustness of LBWARP and FEMWARP and demonstrated computationally that FEMWARP is often more successful than LBWARP at resisting inversion of mesh elements. We also used FEMWARP to study the motion of the canine ventricles under normal conditions.

We now describe some possibilities for future research. First, we would like to develop other types of mesh warping methods that may be more successful in withstanding larger deformations. For example, various models of nonlinear elasticity could be used for this purpose. It is hoped that the nonlinearity in the equations will make the warping procedure more accurate, thus making the method more successful at resisting inversions.

Second, we would like to use our canine ventricle meshes to study the bioelectricity of the heart. To do this, we will couple the heart's electrical activity with its mechanical motion (from the FEMWARP meshes). We will then use the finite element method to simulate the electricity on the beating heart. This will yield new knowledge of the heart's activity since most of the current models have not coupled the heart's electricity with its mechanical motion.

Chapter 3

A Robust Solution Procedure for Hyperelastic Solids with Large Boundary Deformation

3.1 The Problem Under Consideration

The continuum mechanical model under consideration has the following description [21]. Let $B_0 \subset \mathbf{R}^d$ be an undeformed body whose boundary is ∂B_0. Here d, the space dimension, is 2 or 3. Suppose that a pure Dirichlet problem is considered, i.e., a continuous injective function $\phi : \partial B_0 \to \mathbf{R}^d$ is given that prescribes a new position of the boundary. Let ∂B denote the deformed boundary, i.e., $\partial B = \phi(\partial B_0)$. The problem is to find a continuous extension of ϕ that maps all of B_0 to a set $B \subset \mathbf{R}^d$ satisfying the following conditions. For a point $\mathbf{X} \in B_0$, let $\mathbf{x} = \phi(\mathbf{X})$. Let F be the deformation gradient, i.e., $F = d\mathbf{x}/d\mathbf{X}$. It is assumed that F always has positive determinant. The *Green strain* is defined to be $E = (F^T F - I)/2$. Let scalar function $\Psi(F)$ be the *strain energy function*, which is assumed to be a property of the material. For this thesis, we assume that Ψ depends only on the three matrix invariants of E, which are $J = \det(F) = \sqrt{\det(2E + I)}$, $I_1 = \mathrm{trace}(F^T F) = \mathrm{trace}(2E+I)$, and $I_2 = \mathrm{trace}((F^T F)^{-1}) \det(F^T F) = \mathrm{trace}((2E+I)^{-1}) \det(2E+I)$. Further specializing the model, the strain energy is then taken to have the following form due to Mooney and Rivlin

$$\Psi(F) = c(J-1)^2 - k\ln(J) + c_1(I_1 - 3) + c_2(I_2 - 3), \tag{3.1}$$

where c, c_1, c_2, and $k > 0$ are material constants. For the $d = 2$ case, we assume that B_0 is 3D but that the $z-$ displacement is identically 0 and that the $x-$ and $y-$ displacements depend only on x and y; these are the plane strain assumptions. Thus, E has a last row and column of all zeros, and the Mooney-Rivlin formula in (3.1) is applied to this E to come up with the strain energy function for the $d = 2$ case. The condition for static equilibrium in the absence of body forces (written in minimization form) is that

$$\int_{B_0} \Psi(F(\mathbf{X}))\, d\mathbf{X} \tag{3.2}$$

is minimized among all possible choices of ϕ satisfying the boundary constraints. This condition can be rewritten in variational form: for all admissible variations $\delta\mathbf{u}$, that is, functions in the space $[H^1(B_0)]^d$ that vanish on the boundary,

$$\int_{B_0} \frac{\partial \Psi}{\partial F} : \mathrm{Grad}\ \delta\mathbf{u}\, dV = 0, \tag{3.3}$$

where $A : B = \text{trace}(AB^T)$ is used to denote the inner product of second-order tensors A and B. This model also applies to the case of isotropic linear elasticity with two changes in definitions. First, $E = ((F - I)^T + (F - I))/2$ in the case of linear elasticity. Second, $\Psi = \mu \sum_{i,j} E(i,j)^2 + \frac{\lambda}{2} \left(\sum_i E(i,i)\right)^2$, which can be written in terms of the three matrix invariants of E. The plane strain assumptions are assumed to hold in this case.

As mentioned above, we have assumed in the preceding paragraph that the problem under consideration is a pure Dirichlet problem and that there are no body forces. In Section 3.7, we make some remarks about how to generalize our method to the case when boundary conditions are a mixture of displacement and traction and when body forces are present. We should mention that our method does not appear to depend so much on the specifics of the Mooney-Rivlin model, except for the $\ln J$ term, which is quite important for our analysis. Since $dv = J\, dV$, where dv is the volume element of B and dV is the volume element of B_0, this logarithmic term resists infinite compression of the material: if a small positive volume of material in B_0 shrinks to a 0-volume set in B, then this term causes the strain energy at those points to become infinite.

We next describe the discretization of the problem under consideration [4]. We assume that B_0 is discretized with a mesh of triangles or tetrahedra. We assume that the discretization of ϕ, or alternatively the discretization of the displacement $\mathbf{u} = \phi(\mathbf{X}) - \mathbf{X}$, is piecewise linear, with the pieces of linearity being the mesh cells. (In Section 3.7, we discuss extension of our method to piecewise quadratic displacements.) This assumption implies that $\mathbf{u}$ is determined by dm real numbers, namely, the values of $\mathbf{u}$ at nodes. Recall that d is the space dimension, and let m denote the number of interior nodes of the mesh. The finite element method finds the displacement $\mathbf{u}$ such that (3.3) holds for all test functions $\delta\mathbf{u}$ in the test function space. Here, the test function space is the set of $\delta\mathbf{u}$'s that are piecewise linear and continuous and vanish on ∂B_0. The integral in (3.3) is evaluated with a quadrature rule; we have used a 6-point formula having degree 4 precision from [23] for our quadrature in 2D and a 15-point formula having degree 5 precision from [26] for 3D. These quadrature rules were chosen because they have positive weights; the Gauss points are interior; and they possess full triangular or tetrahedral symmetry, respectively. It suffices to solve (3.3) for the dm choices of $\delta\mathbf{u}$ that compose the standard basis for the test function space. This yields a system of dm nonlinear equations for dm unknowns.

The algorithmic question under consideration is how to robustly solve these nonlinear equations. In the next section, we give a summary of of the mesh tangling issue and of our proposal to overcome it. The individual steps of our algorithm are then described in more detail in Sections 2.3, 3.3, and 3.4. Our computational experiments are presented in Sections 3.5 and 3.6. Concluding remarks are presented in Section 3.7.

3.2 Mesh Tangling

The standard method for solving a system of nonlinear equations is Newton iterations. It is well known, however, that if the initial guess is far from the true solution, then Newton iterations will often diverge.

In the case of hyperelasticity with large deformation, there is a specific obstacle that may cause divergence, namely, mesh tangling. The definition of this term is that a mesh is *tangled* if the value of J defined in the previous section is 0 or negative in B_0. In the case of linear displacements, J is piecewise constant and hence this condition can be verified with a finite number of determinant computations. The matter of checking for tangling in the piecewise quadratic case is more complicated and is discussed in Section 3.7. A solution with a tangled mesh is physically invalid. Indeed, the strain energy function is undefined in this case because of the presence of the term $-k \ln J$. Note that although the strain energy function is undefined when J is negative, the Galerkin form (3.3) is still well defined, which is an anomaly that we return to below. We assume that the given problem instance has a valid solution, i.e., there is a piecewise linear function $\mathbf{u}$ satisfying the boundary conditions, as well as (3.3) for all test functions $\delta\mathbf{u}$ plus the condition that $J > 0$ on every element.

Even with this assumption, Newton's method will still often run into problems because the mesh will become tangled on intermediate steps. For example, the starting point for Newton's method is often taken to be $\mathbf{u} = \mathbf{0}$ on every interior node. If the deformation of the boundary is large, then this starting point will have tangling among most of the elements that are adjacent to the boundary.

To understand the difficulty caused by a tangled mesh, suppose that the strain energy has a single term $\Psi(J) = -k \ln J$ (for $k > 0$) on a single element and that there are no boundary constraints. If we treat J as the independent scalar variable, then Newton's method for minimizing this scalar function is

$$J^{(i+1)} = J^{(i)} - \Psi'(J^{(i)})/\Psi''(J^{(i)})$$

which simplifies in this case to

$$J^{(i+1)} = 2J^{(i)}.$$

For positive $J^{(0)}$, this iteration produces a sequence of J's tending to $+\infty$. This is to be expected since the minimum of $-k \ln J$ is indeed at $+\infty$. On the other hand, for a negative $J^{(0)}$, this iteration tends to $-\infty$, which is physically invalid.

The preceding analysis, although naive, seems to point to the following conclusion: Newton's method on the Galerkin form, when applied to a tangled mesh, has a natural tendency to make the tangling worse. We suspect that this fact is probably already known to experts in the field, although we have not been able to find it in the previous literature.

Given the conclusion in the previous paragraph, it seems of paramount importance to avoid tangling. When Newton's method fails in computational mechanics, it is standard practice to try Newton continuation, that is, to apply the load in incremental steps and use the converged solution for one step as the Newton starting point for the next step. We believe that continuation will indeed solve the mesh tangling problem if the steps are sufficiently small. Continuation, however, addresses the tangling issue only in an indirect fashion and therefore is likely to be very inefficient. Our computational experiments confirm the inefficiency of continuation.

We propose a new algorithm for getting around the mesh tangling obstacle. The basic idea is to first untangle the mesh using purely geometric methods. Once the mesh is untangled, the mechanical model is solved. "Untangling the mesh" means finding a ϕ that satisfies the Dirichlet boundary condition and also satisfies $J > 0$. Our new algorithm, which we call UBN (for "untangling before Newton") consists of three steps.

1. First, we attempt to untangle the mesh with the FEMWARP algorithm from our previous work [46]. The FEMWARP algorithm, as described in 2.3, is simple and fast as it involves solving d systems (where $d = 2, 3$) of sparse linear equations.

2. If FEMWARP fails to untangle the mesh, UBN next uses the Freitag-Plassmann mesh untangler [17] called Opt-MS, which is slower but is more reliable. This algorithm is described in Section 3.3. This algorithm is iterative and hence needs a starting point. We use the output of FEMWARP as the starting point for Opt-MS.

3. If Opt-MS cannot untangle the mesh, then UBN reports failure to solve the problem.

4. Else if Opt-MS succeeds, then we take Newton iterations to solve (3.3). The starting point for Newton is an untangled mesh produced by either step 1 or step 2. No continuation is used. On the other hand, Newton's method is safeguarded using a line search described in Section 3.4, which prevents the introduction of new tangling. The line search is based on a technique common in the interior point literature.

3.3 Optimization-based Simplicial Mesh Untangling (Opt-MS)

Opt-MS is a software package that performs optimization-based simplicial mesh untangling and mesh quality improvement [17]. To untangle the simplicial mesh, it seeks to maximize the minimum area (volume) of the elements in each local submesh in a sweeping manner.

Let $\mathbf{x}$ be the location of the *free vertex*, $\mathbf{x}_1, \ldots, \mathbf{x}_n$ the positions of its adjacent vertices, and $t_1, \ldots, t_n$ the incident triangles (tetrahedra) in the local submesh. Then the function which prescribes the minimum area (volume) of an element in the local submesh is given by $q(\mathbf{x}) = \min_{1 \le i \le n} \; A_i(\mathbf{x})$, where A_i is the area (volume) of simplex t_i. In 2D, the area of triangle t_i can be stated as a function of the Jacobian of the element as follows: $A_i = \frac{1}{2} \det(\mathbf{x}_i - \mathbf{x},\; \mathbf{x}_j - \mathbf{x})$ which is a linear function of the free vertex position; the same is true in 3D. Freitag and Plassmann use this fact to formulate the solution to $\max q(\mathbf{x}) = \max \min_{1 \le i \le n} \; A_i(\mathbf{x})$ as a linear programming problem which they solve via the simplex method. On each sweep, m linear programs are solved which sequentially reposition each interior node in the mesh. Sweeps are performed until the mesh is untangled or a maximum number of sweeps has occurred. The reader is referred to [17] for more details on the untangling algorithm.

Let *interior* and *boundary* be m- and b-vectors containing the interior and boundary indices of the mesh, respectively. The nodal ordering used in this thesis is [*interior boundary*]. The Opt-MS algorithm is terminated when either the mesh is untangled or a maximum of 200 passes has occurred.

3.4 Newton Line Search

Newton's method is often employed for solving nonlinear systems of continuously differentiable equations [11]. Let $f : \mathbf{R}^{dm} \to \mathbf{R}^{dm}$, continuously differentiable, be given. The task at hand is to find a $\mathbf{u} \in \mathbf{R}^{dm}$ such that $f(\mathbf{u}) = \mathbf{0}$. Let $\mathbf{u}_0 \in \mathbf{R}^{dm}$ be given. Then, at each iteration k, Newton's Method solves

$$J(\mathbf{u}_k)\mathbf{s}_k = -f(\mathbf{u}_k), \tag{3.4}$$

where J denotes the Jacobian of f, for the Newton step, $\mathbf{s}_k$, and performs the following update

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \mathbf{s}_k. \tag{3.5}$$

If it becomes necessary to satisfy an additional constraint, it is possible to safeguard the Newton step with the introduction of a line search. Let α_k denote the line search parameter. Then, α_k is chosen to be as large as possible such that $0 < \alpha_k \leq 1$ and $\mathbf{u}_{k+1} = \mathbf{u}_k + \alpha_k\mathbf{s}_k$ satisfies the constraint.

It is often difficult to compute the value of α_k that minimizes $f(\mathbf{u}_k + \alpha_k\mathbf{s}_k)$ and satisfies the constraint because f is often a highly nonlinear function. Thus, it is common practice in interior point methods to derive heuristics for computing α_k that allow for both ease of computation and larger steplengths [53]. One such heuristic is to choose α_k so as to stay a fixed percentage away from the boundary. We employ this heuristic in our line search below.

As was pointed out in Section 3.2, the mesh is tangled unless $J = \det(F) > 0$. Thus, we introduce a line search that enforces that $J > 0$ on each iteration of Newton's method. In particular, we begin with $J > 0$ on the zeroth iteration and choose the line search parameter α_k such that $J(\mathbf{u}_{k+1}) \geq 0.1J(\mathbf{u}_k)$ on each element so as to stay a fixed percentage away from the boundary for reasons discussed above.

Let N denote the number of elements in the mesh. Here is a pseudo-code algorithm which shows how the line search parameter is determined:

```
alpha = 1;
for index = 1 : N
  compute J_old for the element whose number is index (from u_old)
  while 1
    compute J_new for this element (from u_old+alpha*s)
    if J_new >= .1 * J_old
      break;
    end
    alpha = alpha * .9;
  end
end
```

3.5 2D Experiments

We designed a series of numerical experiments, which are based on a series of mesh deformations, in order to test the robustness of UBN and to compare it to the standard Newton continuation algorithm. The Newton continuation method is to apply the load in incremental steps and use the converged solution for one step as the Newton starting point for the next step. We perform linear continuation in the displacement variables, i.e., the u variables. We choose Δu so that the predicted displacement of an interior node on a given iteration is no more than one-third of the minimum altitude of the boundary triangles in the initial mesh; this heuristic was chosen with the goal of preventing element inversion. It has the advantage of determining the number of iterations at the beginning of the algorithm which allows the user to predict the amount of time the algorithm will take; however, at times, the Newton continuation method will take steps that are too large resulting in a tangled mesh. Alternatively, Δu may be chosen adaptively. This way of computing Δu suffers from the fact that the algorithm will run very slowly at times when the mesh elements are near inversion. We will show that the Newton continuation algorithm with fixed Δu is already significantly less efficient than UBN. Thus, we do not compute Δu adaptively.

For all of the numerical experiments in this thesis, we set the parameters in (3.1) as follows: $c = \frac{3}{2}, c_1 = \frac{3}{2}, c_2 = \frac{-1}{2}$, and $k = 2(c_1 + 2c_2) = 1$. These values were chosen to ensure that the reference configuration is stress-free and that $k > 0$.

The termination criteria for the Newton loop was that $\frac{\|F\|_2}{\text{area}(B_0)} < 10^{-10}$ in 2D and $\frac{\|F\|_2}{\text{volume}(B_0)} < 10^{-10}$ in 3D.

The 2D tests in this thesis were run on a Dell machine with a 3.60 GHz Intel Pentium 4 processor and 2 GB of RAM running Linux in a non-dedicated environment, as that was all that was available to the author. The 3D tests were run on four identical machines, also in a non-dedicated environment, with care taken to run a particular 3D problem on a single machine for both algorithms. This was done due to the longer time necessary to run the 3D problems. The UBN code was written entirely in Matlab with the exception of Opt-MS, the Freitag-Plassmann mesh untangler, which was written in C.

All of the numerical experiments in the first series involve the deformation of an annulus on each timestep. The annulus test mesh is the same as shown in Figure 2.3. For the annulus tests we discuss the results from UBN and the Newton continuation algorithm.

In the first test, the inner circle was deformed via an outward translation. In particular, we tested for tangling of the mesh at radial increments of 0.1 for both algorithms. The test showed that for UBN, the mesh remained untangled while the inner radius of the annulus was less than 9.3. Once the inner radius reached 9.3, too much deformation had occurred, since J became negative, and elements in the mesh became inverted. Figure 3.1 shows how the annulus mesh looked at each stage of the UBN algorithm; this figure is for an inner radius of 9.2. The mesh boundaries have been drawn with thicker lines so that they may easily be identified. Notice that the FEMWARP mesh is tangled; the Opt-MS mesh is untangled but is far from being a solution to the nonlinear elasticity equations. UBN yields a valid mesh in the final stage with solves these equations.

As this test demonstrates, UBN tolerates a large amount of compression before inversion occurs. In contrast, the Newton continuation method inverted elements in the mesh when the inner radius became 8. Figure 3.2 demonstrates what happens when the Newton continuation method attempts to compress the inner annulus past a radius of 7.9; in this particular example, we attempt to compress it to a radius of 9.2. A zoomed-in view of the last successful major iteration is shown in the upper, left-hand corner. The first failed Newton iteration is shown in the upper, right-hand corner; this figure details the tangling near the inner ring. The major iteration that results is shown on the bottom; this iteration produced a tangled mesh near the inner ring.

Figure 3.3 shows the untangled mesh that results from compressing the inner circle of the annulus to a radius of 7.9.

Rotation of the inner circle occurs in the second deformation series. The inner circle was rotated counterclockwise by the algorithms until it became tangled. For this experiment, we were interested in testing whether or not it was tangled at 5 degree increments. For UBN, the mesh remained untangled until it had been rotated 155 degrees. Figure 3.4 shows how the annulus changes after each stage of the UBN algorithm; this figure is for a rotation of 150 degrees. In order to aid in visualizing the rotation that has occurred, the portion of each mesh that corresponds to the upper half of the original mesh has been drawn in medium-width lines, and the portion that corresponds to the lower half has been drawn in thick lines.

This test demonstrates that UBN tolerates a large amount of twisting before element inversion occurs. In comparison, the Newton continuation method created a tangled mesh for a rotation of only 85 degrees. Figure 3.5 demonstrates how the Newton continuation algorithm fails to produce an untangled mesh for 150 degrees of rotation; this amount of rotation was chosen because it corresponds to the maximal amount of deformation for UBN.

In the upper left, a zoomed-in view of the last successful major iteration is shown. The first unsuccessful Newton iteration is again shown in the upper, right-hand corner. Again, there is tangling near the inner ring. The bottom figure shows the resulting major iteration which produced a tangled mesh near the inner ring. It is not surprising that the tangling occurs near the inner ring since this is where the boundary deformation is occurring.

Figure 3.6 shows the untangled mesh that results from rotating the inner circle of the annulus 80 degrees.

The goal of the third test was to combine translation and rotation of the inner circle. The compression and twisting motions of the first two tests were combined, and radial and angular increments of 0.5 and 5 degrees, respectively, were used when checking for tangling. For UBN, the mesh remains untangled until the radius of the inner circle is 6 and the circle has been rotated by 70 degrees. Figure 3.7 shows the results at each stage of UBN for a radius of 5.5 and a rotation angle of 65 degrees.

This tests demonstrates that UBN can tolerate a large amount of compression and twisting in combination before element inversion occurs. In comparison, the Newton continuation algorithm generated a tangled mesh for a radius of 3.5 and an angle of rotation of 45 degrees. Figure 3.8 demonstrates how the Newton continuation algorithm fails to produce an untangled mesh for an inner radius of 5.5 and 65 degrees of rotation.

For this test, the last successful major iteration is shown in the upper, left-hand corner. Note that the triangles adjacent to the inner ring are nearly degenerate. As the algorithm proceeds to the next minor iteration, we see the tangling that results in the upper, right-hand corner of the figure. The vertices of one particular inverted triangle are labeled 1, 2, and 3. Note that the triangle is nearly degenerate and has actually flipped its orientation. The bottom figure shows the tangled mesh that results at the end of the major iteration.

Figure 3.9 shows the untangled mesh that results from translating the inner circle to a radius of 3 and rotating it 40 degrees.

From these three tests, we conclude that UBN can withstand large amounts of compression and twisting while resisting inversion. However, whenever the deformations are exceptionally large, UBN fails because inversion of elements occurs. Another scenario in which UBN fails, although not seen in the above series of experiments, is when the mesh boundary becomes tangled by the deformation. The Newton continuation method also fails under both of these circumstances, although we have seen that it is less robust as it fails on significantly smaller deformations.

Another important consideration for our study is the relative timings of the two methods. Table 3.1 shows the maximal amount of deformation UBN and Newton continuation could withstand before tangling and the amount of time necessary to achieve these results. Because the Newton continuation method could not withstand as much deformation as the UBN method for any of the three test cases, we ran additional tests with the UBN method to determine the amount of time required to achieve the Newton continuation method's maximal amount of deformation. These numbers for the UBN algorithm are seen within parenthesis in the last column. For example, for the compression test, the UBN algorithm could successfully compress the annulus mesh until the inner radius was equal to 9.2. This required 41 seconds. The Newton continuation algorithm could only compress the mesh until the inner radius was 7.9; this required 489 seconds. To compress the mesh until the inner radius was 7.9 required only 29 seconds of the UBN method.

Table 3.1: Deformation Amounts and Timing Results for UBN and Newton Continuation on the Annulus Test Mesh

	UBN		**Newton continuation**	
motion	**Result**	**Time (s)**	**Result**	**Time (s)**
compression	$r = 9.2$	41	$r = 7.9$	489 (29)
twisting	$\theta = 150^o$	64	$\theta = 80^o$	90 (19)
both	$(r, \theta) = (5.5, 65^o)$	36	$(r, \theta) = (3, 40^o)$	134 (14)

These results suggest that the UBN algorithm is more robust than the standard Newton continuation algorithm for compression and twisting motions. It is also significantly faster than the Newton continuation algorithm.

An experiment was also done to determine whether or not the fact that Opt-MS was written in C rather than Matlab played an important role in making UBN significantly faster than the Newton continuation method which was written solely in Matlab. For the experiment, we ran timing tests to determine the amounts of time necessary to run FEMWARP, to run Opt-MS, to form the Jacobian matrix, and to compute the Newton step for a compression to $r = 7.9$ on the annulus. For this test, FEMWARP required 0.059 seconds, Opt-MS required 0.162 seconds to complete seven untangling passes, an average of 1.48 seconds to form the Jacobian, and an average of 0.0027 seconds to compute the Newton step. UBN solved the problem in 29 seconds and 11 minor iterations. The dominant work for one minor iteration is to form the Jacobian and to solve the Newton step. This required an average of 1.4827 seconds. FEMWARP and Opt-MS both required significantly less time than this, so we could count them each as one minor iteration. Thus, if we were to report the effort UBN spent on solving the problem in iterations rather than time, we would say that it took a total of 13 iterations. In comparison, the Newton continuation algorithm took 70 major iterations for a total of 264 minor iterations on the same problem. This required 489 seconds or 264 iterations. The difference in the number of iterations (and not the fact that Opt-MS was written in C) is what accounts for the fact that UBN is about 17 times faster than the Newton continuation method on this problem.

It is also important to determine whether or not the line search was active in UBN throughout these tests. If it was never active, the Newton continuation algorithm could have been set to run with one major iteration rather than taking incremental steps. However, this was not the case; instead, the line search was often active, and the full Newton step was not taken. For example, in the tests above involving maximal compression and twisting of the annulus, the full Newton step was taken only 31% and 32% of the time, respectively. The majority of these steps were taken when UBN neared the converged solution.

In order to further compare the UBN and Newton continuation methods, we chose a specific nonlinear boundary deformation in order to determine how much deformation each test mesh could withstand when warped according to each method. The nonlinear deformations we used can be written as the sum of an affine mapping plus a parameter-controlled nonlinear term. In 2D, the boundary deformation used for the tests was:

$$\begin{pmatrix} x \\ y \end{pmatrix} \longmapsto \begin{pmatrix} x \\ y \end{pmatrix} + \alpha \begin{pmatrix} 0.2x^2 + 0.1xy + \sin(x - \pi) \\ 0.5y^2 + 5\sin(y)\cos(x) \end{pmatrix},$$

Initially, the parameter, α, controlling the nonlinear term in each boundary deformation was set to zero. Thus, the initial warping of the mesh was done in an affine manner, which is known to yield a valid mesh. The value of α was then increased, whereby increasing the amount of deformation to the mesh. When α became large enough, inversion eventually occurred, and its value, which we denote by α_{max}, was recorded at that time. Table 3.2 gives the results obtained from warping various 2D meshes according to this boundary deformation. The *'s in the table indicate a maximal value of the deformation parameter has been reached before subsequent mesh tangling occurs. For example, the boundary of the arc mesh is tangled at $\alpha = 0.7$. Thus, the algorithm has exhibited maximal performance on this particular test.

The Newton continuation method was allowed to run until it achieved its maximal deformation or until it had run for five times as long as the UBN method, whichever came first. In all cases, the Newton continuation method was terminated when it had run for five times as long as the UBN method; the †'s in the chart indicate that the tests were terminated under this condition. Thus, the amount of deformation the Newton continuation algorithm had achieved when it was terminated is given as a percentage of the maximal UBN deformation in the third column of Table 3.2. The second column shows how long the UBN algorithm took to achieve its maximal deformation; the fourth column indicates that the Newton continuation algorithm would take more than five times as long as the UBN algorithm to achieve its maximal deformation (hence the designation $> 5x$). For example, on the arc mesh, the UBN method took 694 seconds to achieve its maximal deformation of $\alpha_{max} = 0.6$. When allowed to run for five times as long, the Newton continuation algorithm had only achieved 40% of this deformation, i.e., $\alpha = 0.24$. On two out of the five test problems, the Newton continuation algorithm had achieved less than one percent of UBN's maximal deformation when allowed to run for five times as long.

Table 3.2: Values of α_{max} and Timing Results for UBN and Newton Continuation on a 2D Problem [18]

	UBN		**Newton continuation**	
mesh	α_{max}	**Time (s)**	α_{max}	**Time (s)**
arc	0.6∗	694	$40\%^{\dagger}$	$> 5x$
cylinder	0.7	170	$6\%^{\dagger}$	$> 5x$
deer	0.3	870	$2\%^{\dagger}$	$> 5x$
rand400	1∗	222	$< 1\%^{\dagger}$	$> 5x$
rand1000	1∗	942	$< 1\%^{\dagger}$	$> 5x$

Together, these results demonstrate that in 2D, UBN drastically outperformed the standard Newton continuation algorithm on a variety of boundary deformations. On these test problems, it was both more robust and more computationally efficient.

3.6 3D Experiments

In order to further test the robustness of UBN and to compare it to the standard Newton continuation algorithm, we ran an experiment in 3D. For this experiment, relative timings of the two methods were computed. A specific parameter-controlled nonlinear boundary deformation was chosen in the spirit of Section 3.5 in order to determine how much deformation each test mesh could withstand when warped according to each method. For our experiment, the boundary deformation used was:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \longmapsto \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \alpha \begin{pmatrix} 0.1xy \\ 0.5yz \\ 0.1x^2 \end{pmatrix}.$$

Table 3.3 gives the results obtained from warping various 3D meshes according to this boundary deformation.

These results demonstrate that in 3D, UBN drastically outperformed the standard Newton continuation algorithm on a variety of geometries for a given boundary deformation. In particular, it was up to 100 times more computationally efficient than the Newton continuation algorithm. Because the UBN method is significantly more efficient than the Newton continuation algorithm, we do not compare the robustness of the two methods in 3D.

Table 3.3: Values of α_{max} and Timing Results for UBN and Newton Continuation on a 3D Problem [29]

	UBN		**Newton continuation**	
mesh	α_{max}	**Time (s)**	α_{max}	**Time (s)**
distduct	0.0058	$16,711$	22%†	$> 5x$
foam	0.38	7598	5%†	$> 5x$
gear	2.0	4938	18%†	$> 5x$
hook	0.047	$11,758$	8%†	$> 5x$

3.7 Conclusions

In summary, we developed a robust solution method for solving nonlinear elasticity equations for hyperelastic solids with large boundary deformations. The basic idea is to first untangle the mesh using purely geometric methods and second solve the mechanical model; thus, the algorithm was named UBN (for "untangling before Newton"). The first step of our algorithm is to attempt to untangle the mesh with the FEMWARP algorithm. If FEMWARP fails to untangle the mesh, UBN next uses the Freitag-Plassmann mesh untangler called Opt-MS to untangle the mesh. If Opt-MS cannot untangle the mesh, then UBN reports failure to solve the problem. Else if Opt-MS succeeds, UBN takes safeguarded Newton steps to solve (3.3).

We tested the robustness of UBN and compared it to the standard Newton continuation algorithm. We demonstrated that UBN is more successful than the

Newton continuation algorithm at resisting inversion of mesh elements under large deformations. We also demonstrated its computational greater computational efficiency.

We now describe some possibilities for future research. First, we discuss how to generalize our method to the case when boundary conditions are a mixture of displacement and traction and when body forces are present. For example, if traction boundary conditions were present, we would replace the call to FEMWARP with a call to a linear elasticity solver as described in Section 3.1 in the first pass as an attempt to untangle the mesh. In the subsequent call to Opt-MS, we would freeze the traction boundaries and then allow them to move once again within the safeguarded Newton method. The same procedure could be used when body forces are present.

Second, we discuss the extension of our method to piecewise quadratic displacements. The challenge with piecewise quadratic displacements is that checking for tangling is much more complicated, as J is not constant on the element. In particular, it is a function of both the displacement and the location on the element which makes it difficult to determine when $J > 0$ analytically. In addition, J cannot be computed at just a single point on each element. There are some separate necessary and sufficient conditions for element inversion in the literature. Let G be the Jacobian of F. Then one such necessary condition is that $\det(G)$ has the same sign (strictly positive or strictly negative) at some finite list of test points [43]. In this case, we would test for inversion at the Gauss points used for numerical quadrature; however, it is still possible that folding could occur at the corners. A complicated sufficient condition for invertibility involving the Bernstein-Bézier form of a polynomial is given in [49]. Checking that the sufficient condition is met requires running a linear programming algorithm. Salem, Canann, and Saigal have proposed sufficient conditions for quadratic triangles and tetrahedra in [39], [40], [41], and [42].

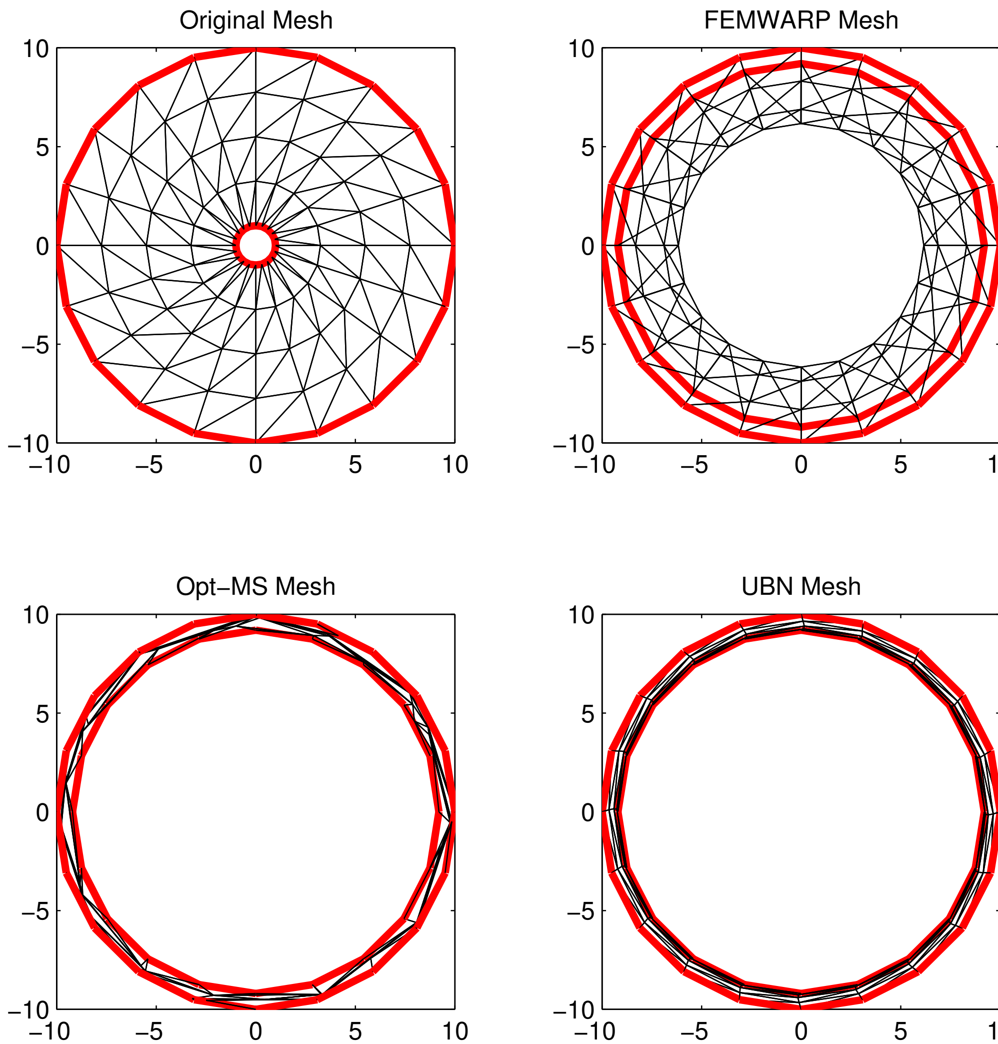

Figure 3.1: Test 1: Translation of Inner Circle of Annulus by UBN

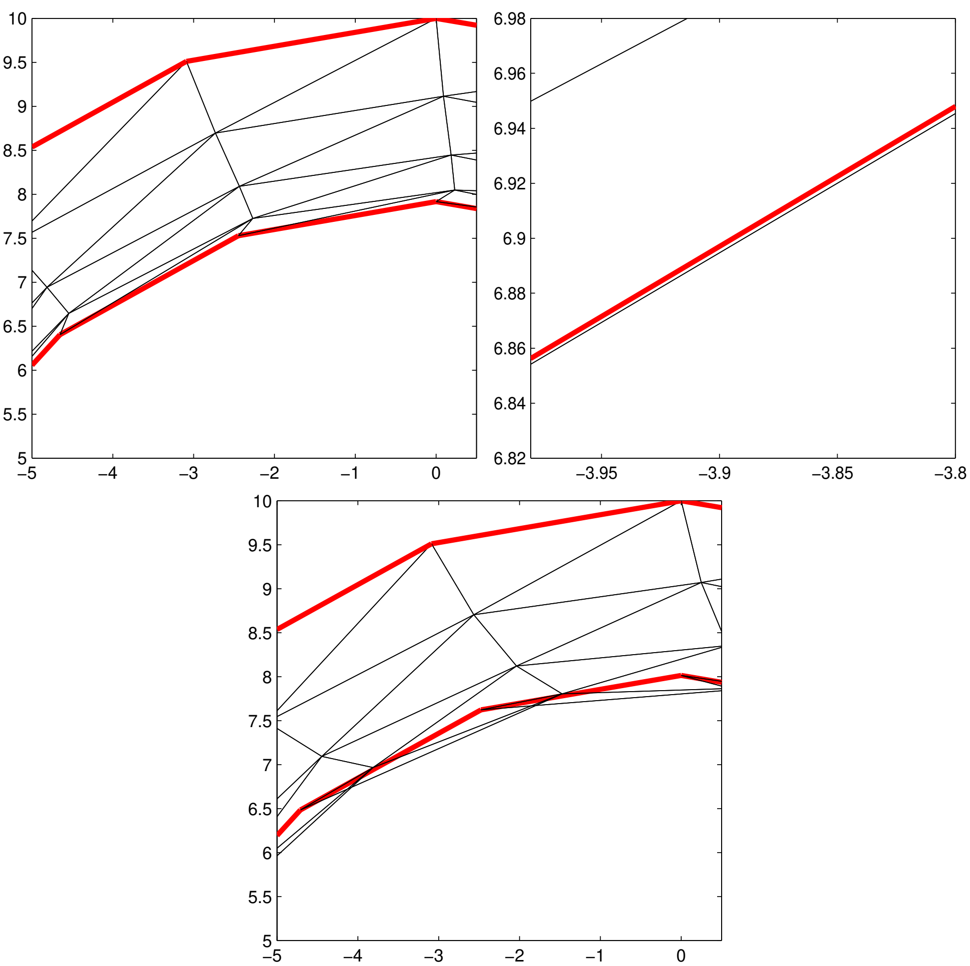

Figure 3.2: Test 1: Translation of Inner Circle of Annulus by Newton Continuation. Early Tangling is Experienced Under Large Deformation.

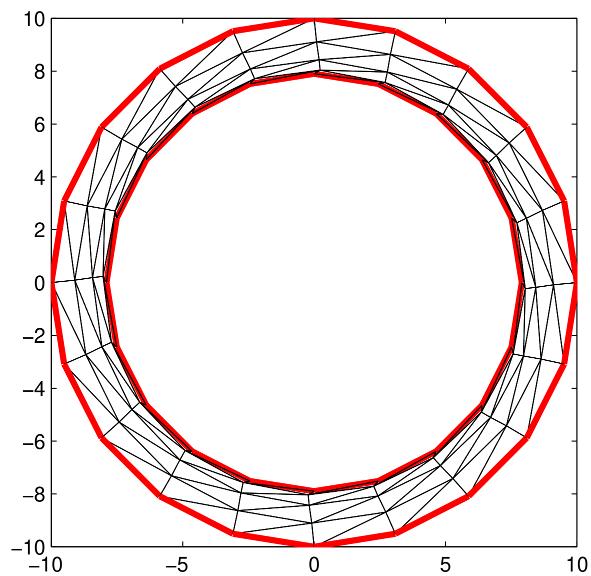

Figure 3.3: Test 1: Successful Translation of Inner Circle of Annulus by Newton Continuation

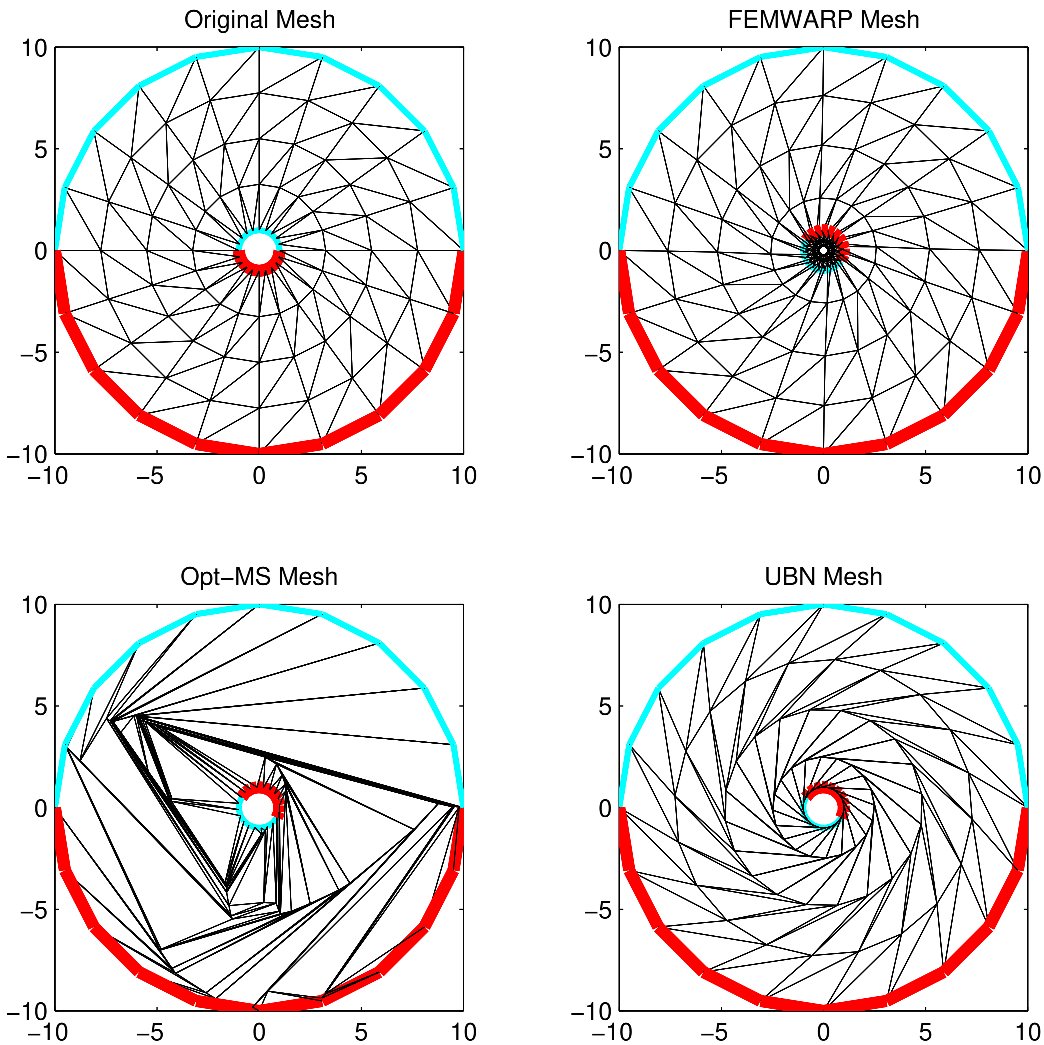

Figure 3.4: Test 2: Rotation of Inner Circle of Annulus by UBN

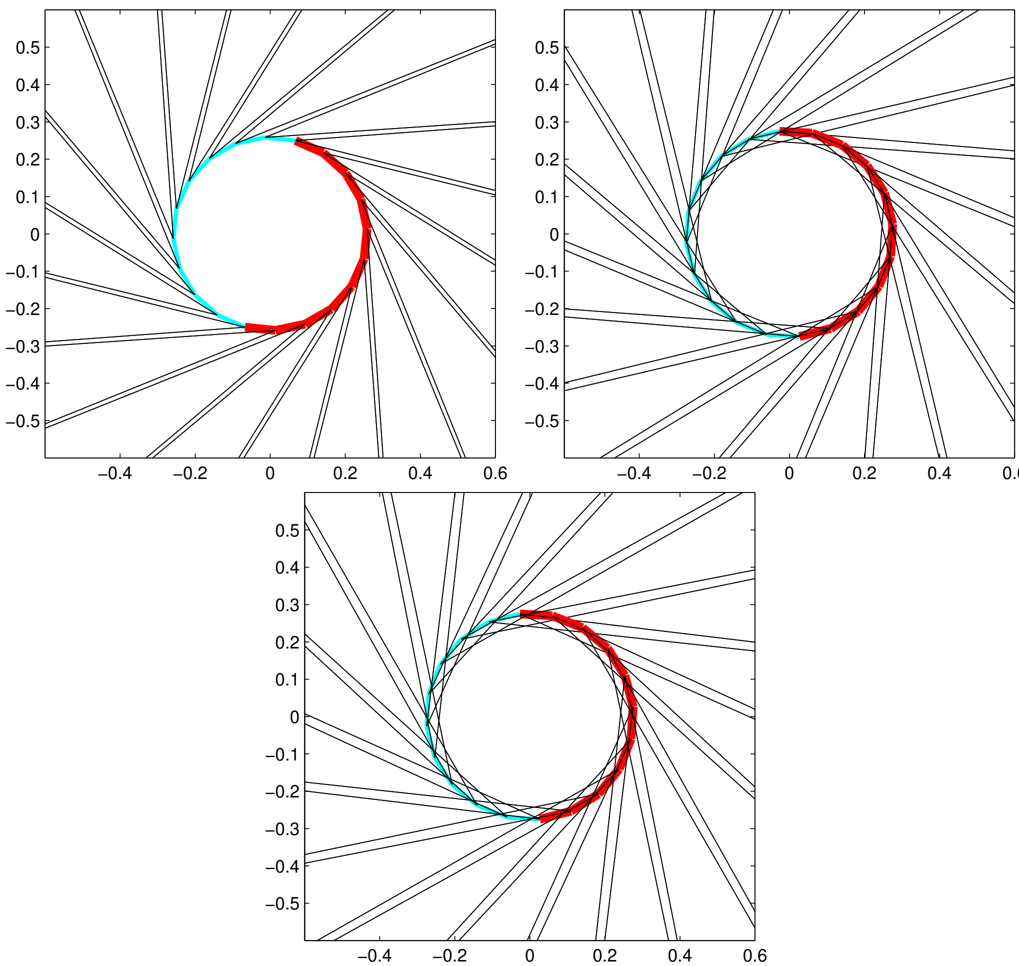

Figure 3.5: Test 2: Rotation of Inner Circle of Annulus by Newton Continuation. Early Tangling is Experienced Under Large Deformation.

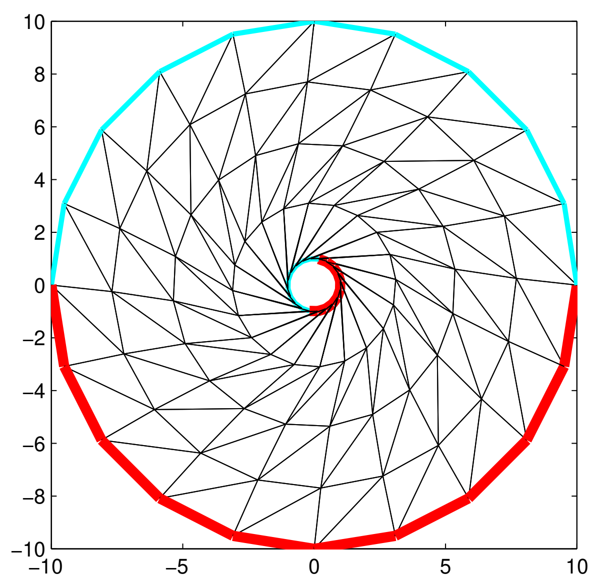

Figure 3.6: Test 2: Successful Rotation of Inner Circle of Annulus by Newton Continuation

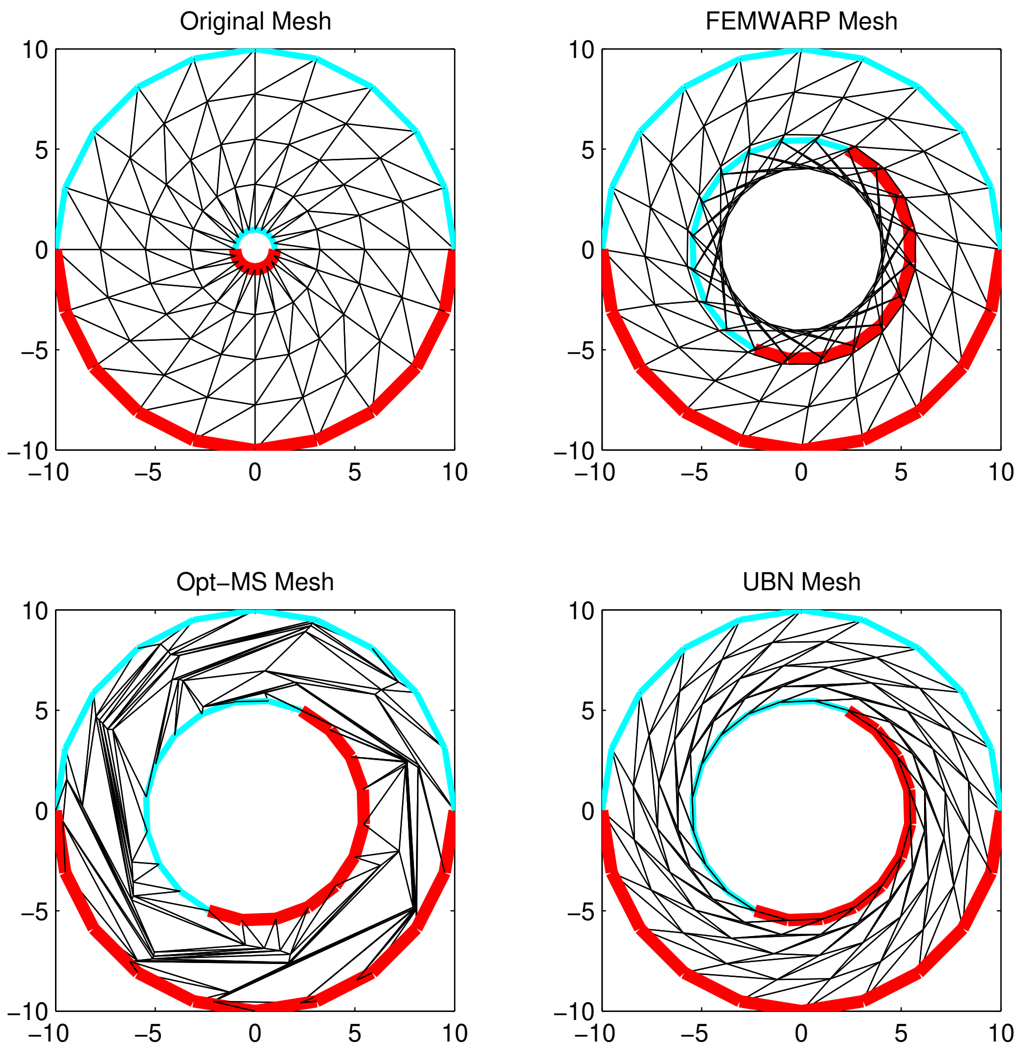

Figure 3.7: Test 3: Translation and Rotation of Inner Circle of Annulus by UBN

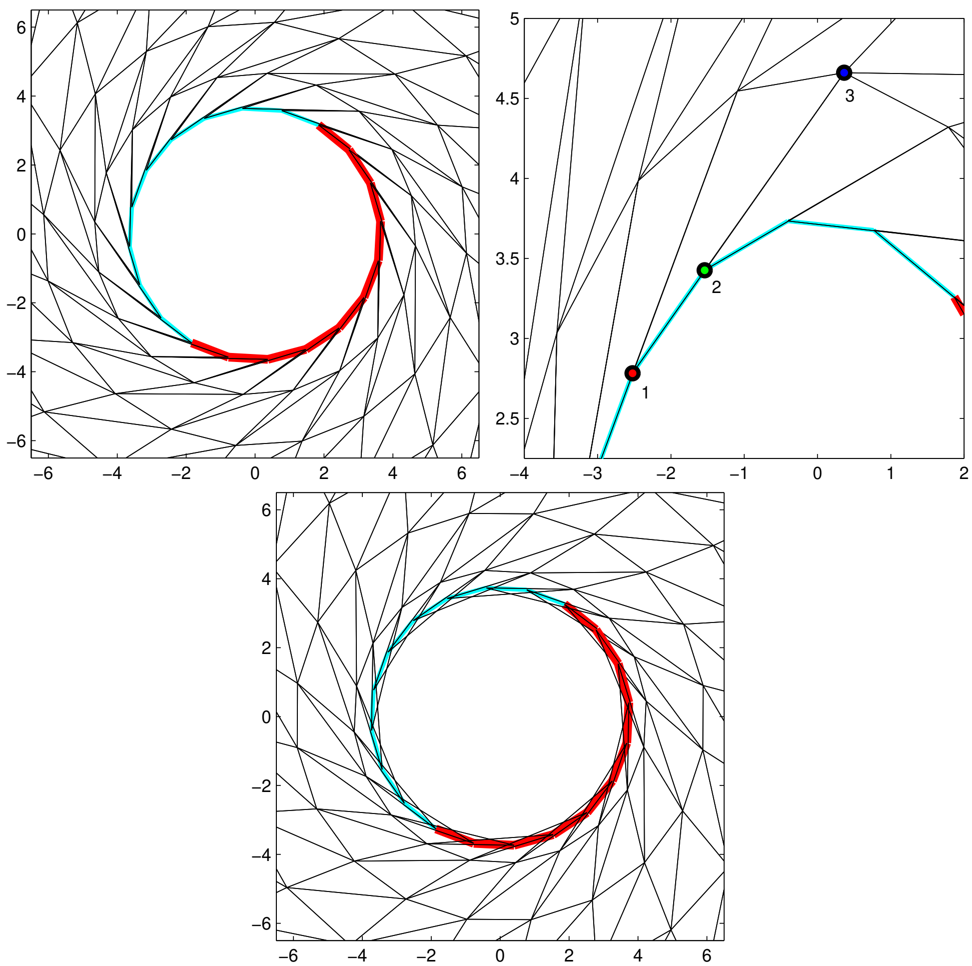

Figure 3.8: Test 3: Translation and Rotation of Inner Circle of Annulus by Newton Continuation. Early Tangling is Experienced Under Large Deformation.

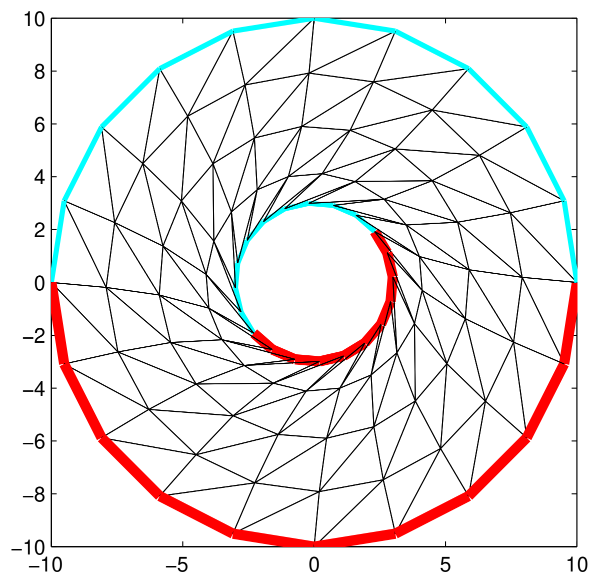

Figure 3.9: Test 3: Successful Translation and Rotation of Inner Circle of Annulus by Newton Continuation

Bibliography

[1] N. Amenta, M. Bern, and D. Eppstein, *Optimal point placement for mesh smoothing*, in Proceedings of 8th ACM-SIAM Symposium on Discrete Algorithms, 1997, pp. 528–537.

[2] E. Amezua, M. Hormaza, A. Hernández, and M. Ajuria, *A method for the improvement of 3d solid finite-element meshes*, Adv. Eng. Softw., 22 (1995), pp. 45–53.

[3] T. J. Baker, *Mesh movement and metamorphosis*, in Proceedings of the Tenth International Meshing Roundtable, Albuquerque, NM, 2001, Sandia National Laboratories, pp. 387–396.

[4] T. Belytschko, W. Liu, and B. Moran, *Nonlinear Finite Elements for Continua and Structures*, Wiley, 2000.

[5] J. U. Brackbill and J. S. Saltzman, *Adaptive zoning for singular problems in two dimensions*, J. Comput. Phys., 46 (1982), pp. 342–368.

[6] S. C. Brenner and L. R. Scott, *The Mathematical Theory of Finite Element Methods*, Springer, 1994.

[7] S. A. Canann, M. B. Stephenson, and T. Blacker, *Optismoothing: An optimization-driven approach to mesh smoothing*, Finite Elem. Anal. Des., 13 (1993), pp. 185–190.

[8] W. M. Cao, W. Z. Huang, and R. D. Russell, *A study of monitor functions for two dimensional adaptive mesh generation*, SIAM J. Sci. Comput., 20 (1999), pp. 1978–1994.

[9] J. Dompierre, P. Labbé, M. Vallet, and R. Camarero, *How to subdivide pyramids, prisms and hexahedra into tetrahedra*, in Proceedings of the 8th

International Meshing Roundtable, Albuquerque, NM, 1999, Sandia National Laboratories, pp. 195–204.

[10] A. S. Dvinsky, *Adaptive grid generation from harmonic maps on riemannian manifolds*, J. Comput. Phys., 95 (1991), pp. 450–476.

[11] J. E. Dennis, Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Society for Industrial and Applied Mathematics, 1996.

[12] L. Freitag, *On combining laplacian and optimization-based mesh smoothing techniques*, in AMD - Vol. 220 Trends in Unstructured Mesh Generation, The American Society of Mechanical Engineers, 1997, pp. 37–43.

[13] L. Freitag and P. Knupp, *Tetrahedral element shape optimization via the jacobian determinant and condition number*, in Proceedings of the 8th International Meshing Roundtable, Albuquerque, NM, 1999, Sandia National Laboratories, pp. 247–258.

[14] ——, *Tetrahedral mesh improvement via optimization of the element condition number*, Int. J. Numer. Meth. Eng., 53 (2002), pp. 1377–1391.

[15] L. Freitag, P. Knupp, T. Munson, and S. Shontz, *A comparison of optimization software for mesh shape-quality improvement problems*, in Proceedings of the Eleventh International Meshing Roundtable, Albuquerque, NM, 2002, Sandia National Laboratories, pp. 29–40.

[16] L. Freitag and C. Ollivier-Gooch, *Tetrahedral mesh improvement using swapping and smoothing*, Int. J. Numer. Meth. Eng., 40 (1997), pp. 3979–4002.

[17] L. Freitag and P. Plassmann, *Local optimization-based simplicial mesh untangling and improvement*, Int. J. Numer. Meth. Eng., 49 (2000), pp. 109–125.

[18] L. Freitag-Diachin, Personal Communication, (October 2002).

[19] A. George and J. W. H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, 1981.

[20] G. H. Golub and C. F. Van Loan, *Matrix Computations*, John Hopkins University Press, third ed., 1996.

[21] G. Holzapfel, *Nonlinear Solid Mechanics: A Continuum Approach for Engineering*, Wiley, 2000.

[22] W. Z. Huang and R. D. Russell, *Moving mesh strategy based on a gradient flow equation for two-dimensional problems*, SIAM J. Sci. Comput., 20 (1999), pp. 998–1015.

[23] T. J. R. Hughes, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Dover Publishers, 2000.

[24] A. A. Johnson and T. E. Tezduyar, *Simulation of multiple spheres falling in a liquid-filled tube*, Comput. Method. Appl. M., (1995).

[25] C. Johnson, *Numerical Solutions of Partial Differential Equations by the Finite Element Method*, Studentlitteratur, 1987.

[26] P. Keast, *Moderate-degree tetrahedral quadrature formulas*, Comput. Method. Appl. M., 55 (1986), pp. 339–48.

[27] P. Knupp, *Winslow smoothing on two-dimensional unstructured meshes*, Eng. Comput., 15 (1999), pp. 263–268.

[28] ——, *Algebraic mesh quality metrics*, SIAM J. Sci. Comput., 23 (2001), pp. 193–218.

[29] ——, Personal Communication, (October 2002).

[30] R. Li, T. Tang, and P. Zhang, *Moving mesh methods in multiple dimensions based on harmonic maps*, J. Comput. Phys., 170 (2001), pp. 562–588.

[31] E. McVeigh and O. Faris, Personal Communication, (February 2003).

[32] S. Mitchell and S. Vavasis, *Quality mesh generation in higher dimensions*, SIAM J. Comput., 29 (2000), pp. 1334–1370.

[33] S. A. Mitchell, *Mesh Generation with Provable Quality Bounds*, PhD thesis, Cornell University, 1993.

[34] S. G. Nash and A. Sofer, *Linear and Nonlinear Programming*, McGraw-Hill, Inc., 1996.

[35] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, New York, 1999.

[36] J. Ortega, *Numerical Analysis: A Second Course*, Academic Press, 1972.

[37] V. N. Parthasarathy and S. Kodiyalam, *A constrained optimization approach to finite element mesh smoothing*, Finite Elem. Anal. Des., 9 (1991), pp. 309–320.

[38] Y. Saad, *Iterative Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics, second ed., 2003.

[39] A. Salem, S. Canann, and S. Saigal, *Robust quality metric for quadratic triangular 2d finite elements*, Trends in unstructured mesh generation, 220 (1997), pp. 73–80.

[40] ——, *Mid-node admissible spaces for quadratic triangular 2d arbitrarily curved finite elements*, Int. J. Numer. Meth. Eng., 50 (2001), pp. 253–272.

[41] ——, *Mid-node admissible spaces for quadratic triangular 2d finite elements with one edge curved*, Int. J. Numer. Meth. Eng., 50 (2001), pp. 181–197.

[42] A. Salem, S. Saigal, and S. Canann, *Mid-node admissible space for 3d quadratic tetrahedral finite elements*, Eng. Comput., 17 (2001), pp. 39–54.

[43] M. Shephard, S. Dey, and M. Georges, *Automatic meshing of curved three-dimensional domains: Curving finite elements and curvature-based mesh control*, in Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations, I. Babuska, J. Flaherty, J. Hopcroft, W. Henshaw, J. Oliger, and T. Tezduyar, eds., Springer Verlag, 1995.

[44] J. Shewchuk, *Triangle: Engineering a 2d quality mesh generator and delaunay triangulator*, in Proceedings of the First Workshop on Applied Computational Geometry, New York, NY, 1996, Association for Computing Machinery, pp. 124–133.

[45] ——, *What is a good linear element? Interpolation, conditioning, and quality measures*, in Proceedings of the Eleventh International Meshing Roundtable, Albuquerque, NM, 2002, Sandia National Laboratories, pp. 115–126.

[46] S. Shontz and S. Vavasis, *A linear weighted laplacian smoothing framework for warping tetrahedral meshes*, Submitted to SIAM J. Sci. Comput., (2004).

[47] J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin, *Numerical Grid Generation*, North-Holland, 1985.

[48] R. Varga, *Matrix Iterative Analysis*, Springer-Verlag, second ed., 2000.

[49] S. Vavasis, *A Bernstein-Bézier sufficient condition for invertibility of polynomial mapping functions*, See http://www.arxiv.org, cs.NA/0308021 (2003).

[50] D. White and G. Rodrigue, *Improved vector FEM solutions of Maxwell's equations using grid pre-conditioning*, Int. J. Numer. Meth. Eng., 40 (1997), pp. 3815–3837.

[51] G. Windisch, *M-matrices in Numerical Analysis*, Teubner, 1989.

[52] A. Winslow, *Numerical solution of the quasi-linear Poisson equation in a nonuniform triangle mesh*, J. Comput. Phys., 1967 (1967), pp. 149–172.

[53] S. J. Wright, *Primal-Dual Interior-Point Methods*, Society for Industrial and Applied Mathematics, 1997.

[54] P. Zavattieri, E. Dari, and G. Buscaglia, *Optimization strategies in unstructured mesh generation*, Int. J. Numer. Meth. Eng., 39 (1996), pp. 2055–2071.