Optimization-based Methods in High-Order Mesh Generation and Untangling

Michael Stees

B.A. Computer Science, Monmouth College, 2013

Submitted to the graduate degree program in Department of Electrical Engineering and Computer Science and the Graduate Faculty of the University of Kansas in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Dr. Suzanne M. Shontz, Chairperson

Dr. Perry Alexander

Committee members

Dr. Prasad Kulkarni

Dr. Jim Miller

Dr. Weizhang Huang

Date defended: May 1, 2020

The Dissertation Committee for Michael Stees certifies
that this is the approved version of the following dissertation :

Optimization-based Methods in High-Order Mesh Generation and Untangling

Dr. Suzanne M. Shontz, Chairperson

Date approved: May 8, 2020

Abstract

High-order numerical methods for solving PDEs have the potential to deliver higher solution accuracy at a lower cost than their low-order counterparts. To fully leverage these high-order computational methods, they must be paired with a discretization of the domain that accurately captures key geometric features. In the presence of curved boundaries, this requires a high-order curvilinear mesh. Consequently, there is a lot of interest in high-order mesh generation methods. The majority of such methods warp a high-order straight-sided mesh through the following three-step process. First, they add additional nodes to a low-order mesh to create a high-order straight-sided mesh. Second, they move the newly added boundary nodes onto the curved domain (i.e., apply a boundary deformation). Finally, they compute the new locations of the interior nodes based on the boundary deformation. We have developed a mesh warping framework based on optimal weighted combinations of nodal positions. Within our framework, we develop methods for optimal affine and convex combinations of nodal positions, respectively. We demonstrate the effectiveness of the methods within our framework on a variety of high-order mesh generation examples in two and three dimensions. As with many other methods in this area, the methods within our framework do not guarantee the generation of a valid mesh. To address this issue, we have also developed two high-order mesh untangling methods. These optimization-based untangling methods formulate unconstrained optimization problems for which the objective functions are based on the unsigned and signed angles of the curvilinear elements. We demonstrate the results of our untangling methods on a variety of two-dimensional triangular meshes.

Acknowledgements

I would like to take this opportunity to thank the people who made this thesis possible. First and foremost, I would like to express my gratitude to my advisor Suzanne Shontz. She has been instrumental in my growth as a researcher and professional in the field. I would also like to thank my committee members, Perry Alexander, Prasad Kulkarni, Jim Miller, and Weizhang Huang, for their support. Thank you to all of the friends that have helped and supported me during this journey. Namely, Nick, Jocelyn, Kurt, Alex, and everyone else that I have had the pleasure of working with. I would also like to thank my family for their love and support.

I would also like to express my gratitude to Madison and Lila Self for creating the Madison and Lila Self Graduate Fellowship which funded several years of my graduate work. In addition, I would like to thank the National Science Foundation for funding my work under grants CCF 1717894 and OAC 1808553. Finally, I would like to thank the Army Research Office for funding the infrastructure which I used for performance testing under contract W911NF-15-1-0377.

Contents

List of Figures

List of Tables

Chapter 1

Introduction to Thesis

Before discussing the focus of this thesis, we first review the role of the domain partitioning (finite element mesh) in finite element calculations. Before proceeding, let us first pose the model problem:

$$\begin{aligned} -\triangle u = f(x_1,x_2), &\text{defined in } \Omega = (x_1,x_2)\colon x_1^2+x_2^2<1 \\ u = 0 &\text{ on } \partial\Omega, \end{aligned}$$

where Ω is a bounded planar domain with the boundary $\partial\Omega$, f is a given real-valued function bounded in Ω, and $\triangle u = \partial^2 u/\partial x_1^2 + \partial^2 u/\partial x_2^2$. The formal name of this model problem is the Poisson Equation with homogeneous Dirichlet boundary conditions. More specifically, this is the strong formulation of the problem. To solve this problem with the Finite Element Method, we first need to write it in the weak or variational form [16]. To do this, we multiply both sides by a test function $v \in V$ and then integrate both sides. V is defined in the following way:

$$\begin{aligned} V = \{v\colon\ & v \text{ is continuous on } \Omega, \\ & \frac{\partial v}{\partial x_1} \text{ and } \frac{\partial v}{\partial x_2} \text{ are piecewise continuous on } \Omega, \\ & \text{and } v = 0 \text{ on } \partial\Omega\}. \end{aligned}$$

The weak form thus looks as follows:

$$\int_\Omega (-\triangle u)\, v\, dx = \int_\Omega f\, v\, dx.$$

From Green's Theorem, we know:

$$\int_\Omega (\triangle u)\, v\, dx = \int_{\partial\Omega} (\partial_n u)\, v\, ds - \int_\Omega \nabla u \cdot \nabla v\, dx.$$

Substituting this information into the weak form, we get the following:

$$-\int_{\partial\Omega} (\partial_n u)\, v\, ds + \int_\Omega \nabla u \cdot \nabla v\, dx = \int_\Omega f\, v\, dx.$$

Since we know that $v \in V$, then we know that $v = 0$ on $\partial\Omega$, so we can simplify the weak form to:

$$\int_\Omega \nabla u \cdot \nabla v\, dx = \int_\Omega f\, v\, dx.$$

With our model problem in a suitable form, we now list the steps of a finite element method as follows: (1) partition the physical domain into a collection of elements; (2) rewrite the PDE in weak form; (3) calculate the element stiffness matrices; (4) assemble the global stiffness matrix and associated linear system; (5) apply the given boundary conditions; and (6) solve the resulting linear system. For the purposes of this discussion, we will focus primarily on discretizing the physical domain and omit details related to steps (2)-(6). See [16] for a discussion of the other steps. To discretize the physical domain, we construct a mesh. From our problem statement, we know that our domain is the unit disk. One choice that needs to be made at this point is the type of elements that compose the mesh. Two common choices for two-dimensional domains are triangles and quadrilaterals. Assuming we are using triangular elements, we can partition our domain by subdividing Ω into a set τ_h of n non-overlapping triangles K_i, i.e.,

$$\Omega = K_1 \cup K_2 \cup \cdots \cup K_n,$$

such that the intersection of any two triangles is an edge, a vertex, or the empty set [16]. A partition satisfying these requirements is known as a conformal mesh. In Figure 1.1, we show four possible low-order meshes that represent the unit disk, with the boundary of the disk denoted in red. While all of these meshes provide a discretization of the domain, some of them have better properties when connected to a finite element solver. In particular, it has been shown that the shape and size of the elements that compose the mesh can have a major influence on convergence and accuracy of the finite element solver because of their impact on interpolation error, conditioning of the stiffness matrices, and solution approximation [27].

Looking closely at the figure, it is clear that the mesh does not capture the smooth boundary of the disk. One obvious solution to remedy this is to add additional elements to better capture the curvature of the domain, a process known as h-refinement. In Figure 1.2, we show an example of a mesh generated by this approach. While this mesh approach better captures the curved feature of our domain, it also adds to the computational complexity of the associated finite element solver due to the additional nodes and elements.

An alternative solution to adding more elements is the use of high-order elements, that is, elements that are defined by polynomials beyond degree one. This process is known as p-refinement. These high-order elements have extra degrees of freedom that allow them to better capture curved domains via curvilinear elements. The overall objective of this thesis is to present methods for robustly generating high-order finite element meshes. To accomplish the objective, we conducted research in two areas, namely high-order mesh generation and high-order mesh untangling. Results from the first area were published in two peer-reviewed conference proceedings. With some modifications, the methods described within these two publications became the optimal convex and optimal affine methods defined within our framework in Chapter 2. Within the second area, we published two angle-based mesh untangling methods in peer-reviewed conference proceedings that are reproduced in this thesis as Chapter 3, and Chapter 4.

Our research in high-order mesh generation was inspired by techniques for mesh warping. The need for mesh warping arises in fields like computer graphics, cardiology, and a variety of time-dependent science and engineering applications where the mesh needs to be updated at each timestep due to a moving domain boundary. In addition to the applications listed above, another application of particular interest to this thesis is the moving domain boundary that occurs during the typical high-order mesh generation process when the high-order nodes are projected onto the curved boundaries of the domain. In typical mesh warping methods, the 2D formulation for the

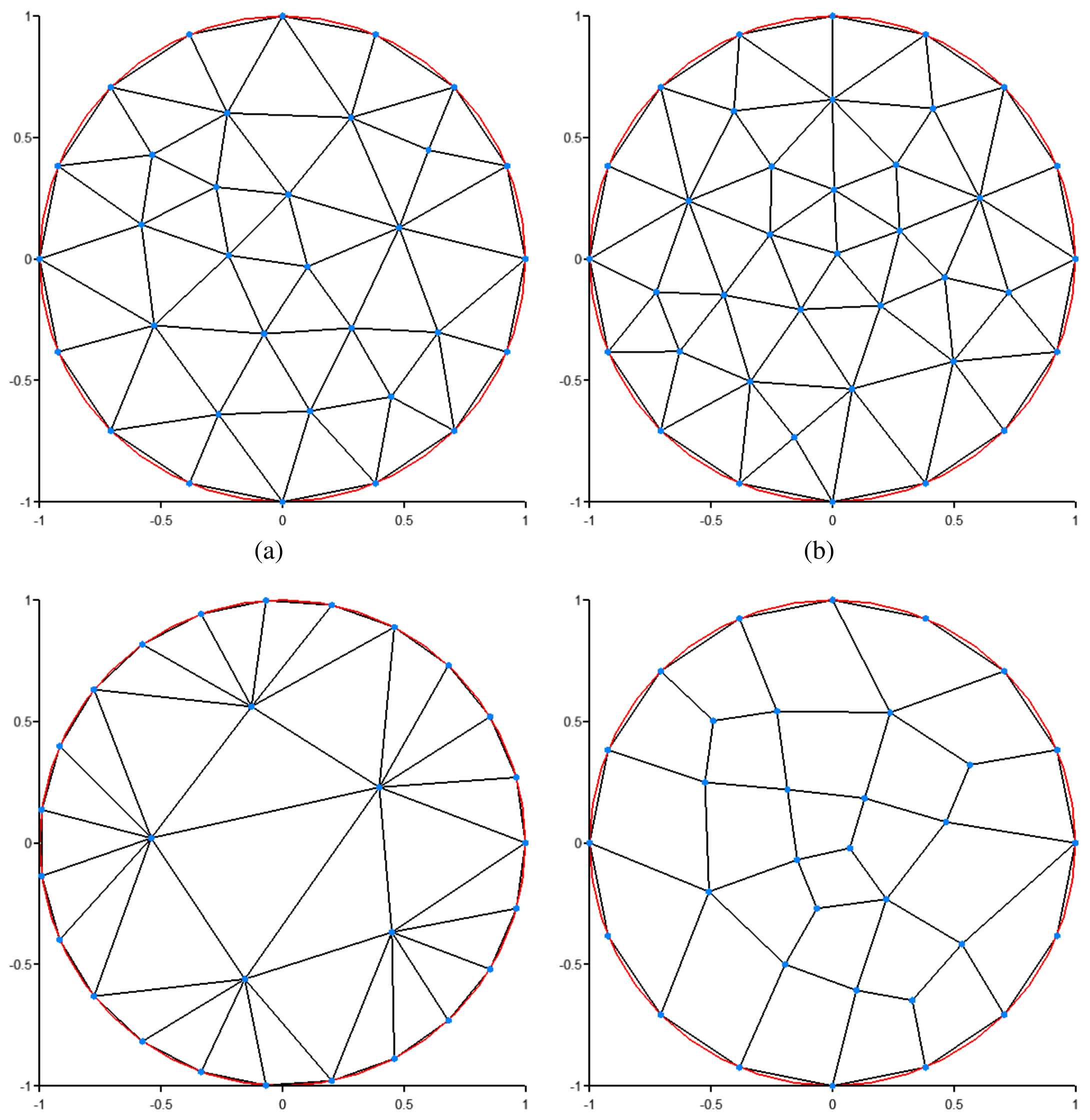


Figure 1.1: Four possible meshes that could be generated for the unit disk with the boundary of the disk shown in red.

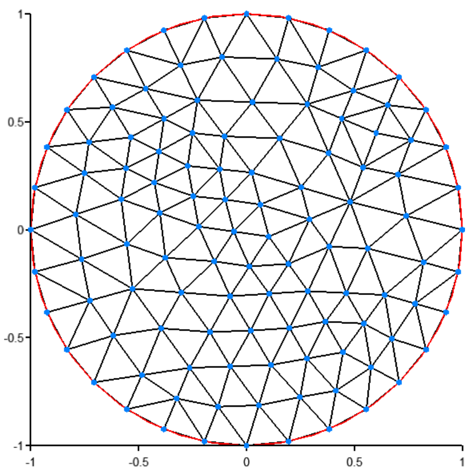


Figure 1.2: A refined mesh of the unit disk.

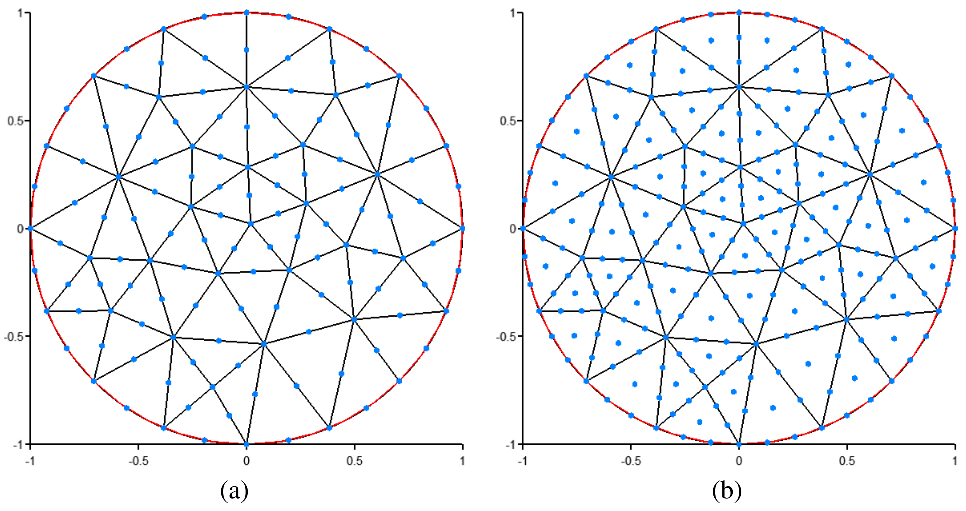


Figure 1.3: Second- and third-order meshes of the unit disk.

problem being addressed is as follows: given a two-dimensional domain with an associated triangular mesh, suppose the boundary mesh is deformed. Is there a method for moving the interior nodes that will result in a good quality mesh while leaving the mesh topology fixed? To address this question, we have developed a mesh warping framework based on optimal weighted combinations of nodal positions. The first method within our framework was inspired by the low-order Log-Barrier Based Mesh Warping (LBWARP) method developed by Shontz and Vavasis in [29]. LBWARP is a method that uses optimal convex combinations of nodal positions to dictate the movement of the interior nodes based on a user-supplied boundary deformation. The other method that we developed within our framework uses optimal affine combinations of nodal positions. In Chapter 2, we introduce our framework for high-order mesh warping and discuss the two methods within our framework and their constrained optimization problem formulations.

Our second area of research applies when the mesh warping methods described above fail to maintain a good quality mesh. This area is known as mesh untangling, and such techniques are applied to meshes with invalid (tangled) elements with the goal of moving interior mesh nodes to create valid (untangled) elements. For low-order triangles, a common approach for checking element validity is based on the orientation of the nodes. If the nodes are in counterclockwise order, then the element is valid, otherwise it is invalid. In Fig. 1.4, we show an example of an invalid patch of low-order triangles and one possible untangling of the patch. This is only one possible untangled configuration because the result will be dependent on the choice of untangling technique and the choice of mesh quality metric. For curvilinear elements, detecting tangling is generally more difficult. For a second-order element, tangling can occur due to the edges of the element intersecting at places other than a node. We show an example of this in Fig. 1.5, where the invalid element is shown in red. For element orders beyond second-order, the placement of the face nodes also plays a role in element validity. For this reason, while there are numerous techniques for untangling low-order meshes, high-order mesh untangling techniques generally rely on validating a mapping from a reference element to a physical element. In Fig. 1.6, we show the mapping $\phi(\xi, \eta)$ from the second-order unit triangle in reference space to the second-order triangle

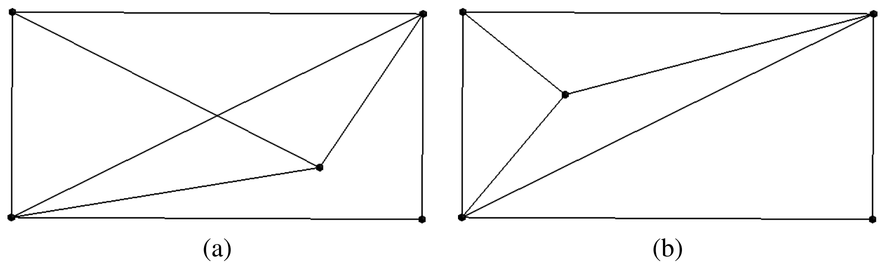


Figure 1.4: A tangled low-order patch (a) and one possible untangled version of the patch (b).

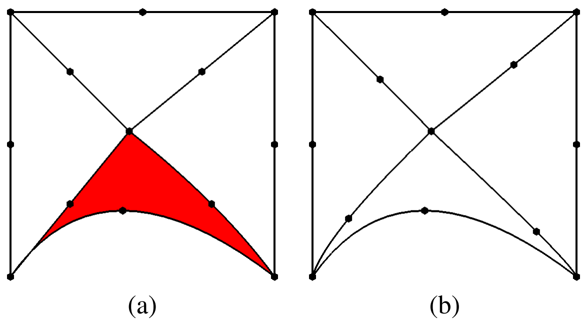


Figure 1.5: A tangled patch of 2nd order elements (a) and one possible untangled version of the patch (b).

in physical space.

Given this mapping, a common measure of element distortion is the scaled Jacobian [3]:

$$\text{scaled Jacobian} = \frac{\min J(\xi)}{\max J(\xi)},$$

where (ξ, η) are coordinates in the reference element, and J is the determinant of the Jacobian of $\phi(\xi, \eta)$. In contrast with the other methods in this area, the two methods that we developed do not depend on the mapping. In particular, the two angle-based methods that we developed successfully

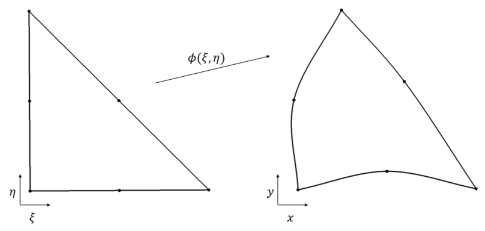


Figure 1.6: The mapping from the second-order unit triangle in reference space to the second-order triangle in physical space.

untangle curvilinear triangular elements. In Chapters 3 and 4, we present our methods based on unsigned and signed angles, respectively. Finally in Chapter 5, we summarize the key results of this work as a whole.

Chapter 2

High-order Mesh Warping Based on Optimal Weighted Combinations of Nodal Positions

Abstract

In this work, we present a framework for high-order triangular and tetrahedral mesh warping based on optimal weighted combinations of nodal positions. Our framework consists of three steps. First, a set of optimal weights relating each node to its neighbors in the initial high-order mesh is calculated. Second, a user-defined boundary deformation is applied. Third, the final positions of the interior nodes are solved for based on the boundary deformation and the optimal weights. We are presenting two methods within our framework for computing optimal weighted combinations. In particular, we consider optimal affine and convex combinations of nodal positions, respectively. We present several numerical examples in both two and three dimensions which demonstrate the capabilities of our framework.

2.1 Introduction

Mesh warping methods are of considerable interest to the scientific computing community because of their ability to deform an initial mesh to a target mesh given only a boundary deformation. This functionality is important for applications that require a single boundary deformation (e.g., curving the boundary in high-order mesh generation), or problems that require multiple deformation steps like moving meshes. The key detail that differentiates these methods is their approach for moving the interior nodes following the boundary deformation. One approach transforms the interior nodes based on the solution of a partial differential equation [6, 18, 22, 37]. More specifically, Xie et al. [37] employed a linear elasticity approach, while Persson and Peraire [22] considered a nonlinear elasticity approach. Moxey et al. [18] used a thermoelastic model, and Fortunato and Persson [6] expressed the problem in terms of the Winslow equations. The second approach optimizes an objective function [3, 4, 5, 7, 8, 13, 24, 25, 26, 30, 31, 32, 33, 34]. Many of the proposed objective functions include a measure of element validity, which allows the methods to address invalid elements. While not all of the methods guarantee valid meshes, many of them are robust [3, 5, 7, 8, 24, 25, 34].

In this chapter, we introduce our general optimization-based framework for high-order mesh warping based on our earlier work in [31, 32]. This framework expresses each interior node as an optimal weighted combination of neighboring nodes. While we provide two geometry-based methods within our framework for weight computation, alternative application-specific methods could be used (e.g., based on problem physics, etc.). The main focus of this work is to present our framework, as well as present additional numerical evidence of its effectiveness. The remainder of this paper is organized as follows. In Section 2, we describe the mesh warping problem. In Section 3, we introduce our framework and two definitions of "optimal" weights. In Section 4, we demonstrate the performance of our method on several examples in two and three dimensions. Finally, in Section 5, we offer concluding remarks and discuss some directions for our future work.

2.2 Problem Statement

Before describing the specifics of our framework, we will discuss the mesh warping problem in more detail. As described in [14], given an initial domain D_{init} and a triangular or tetrahedral mesh M_{init} conforming to the domain, we wish to deform the boundary of the domain to D_{final} and update the mesh to M_{final} as a result. Given a mapping F between D_{init} and D_{final}, we could easily transform M_{init} to M_{final} by evaluating $F(M_{init})$. Unfortunately, such a mapping between domains is generally unknown in practice. One option to get M_{final} is to simply generate a new mesh on the deformed domain, but there is no guarantee that the resulting mesh would have any similarity (e.g., mesh topology) to M_{init}. Furthermore, remeshing can lead to accumulated error from reinterpolation of the PDE solution on the mesh from one domain to the next [27] and is often slower. With these factors in mind, if the mesh topology is fixed, then strategies based on interior node-movement are a good choice.

2.3 Framework

In this section, we present our optimization-based framework for high-order mesh warping based on the Linear Weighted Laplacian Smoothing (LWLS) framework described by Shontz in [28]. The first step in our framework is to solve an optimization problem for each interior node to calculate a set of weights that relates the interior node to its neighbors in the initial high-order mesh. While we focus on two generic geometry-based methods for calculating the weights, application-specific methods could be used instead. To formalize our discussion, we borrow the following notation from [28] to describe the 2D formulation of the problem. The 3D formulation can be described in a similar fashion. Denote the x- and y-coordinates of the i^{th} interior node by (x_i, y_i). Furthermore, denote the x- and y-coordinates of the neighbors of node i as $\{(x_j, y_j) : j \in N_i\}$, where N_i is the set of all elements to which i belongs. There are several possible definitions for the local neighboring set based on use of the low-order nodes, high-order nodes, or some combination of both. We chose to include all nodes for the neighboring set because this definition resulted in elements with less distortion than other definitions in the presence of larger deformations. For smaller deformations, a simpler set could be chosen (e.g., only the high-order nodes, etcetera), but care must be taken in the case of the certain types of weights. For each interior node i, this information can be represented as the following linear system, where w_{ij} are the weights:

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

$$\sum_{j \in N_i} w_{ij} y_j = y_i.$$

Adding the additional constraint that the weights sum to one results in the following problem of finding an affine combination of the x- and y-coordinates of the vertices adjacent to node i:

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

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

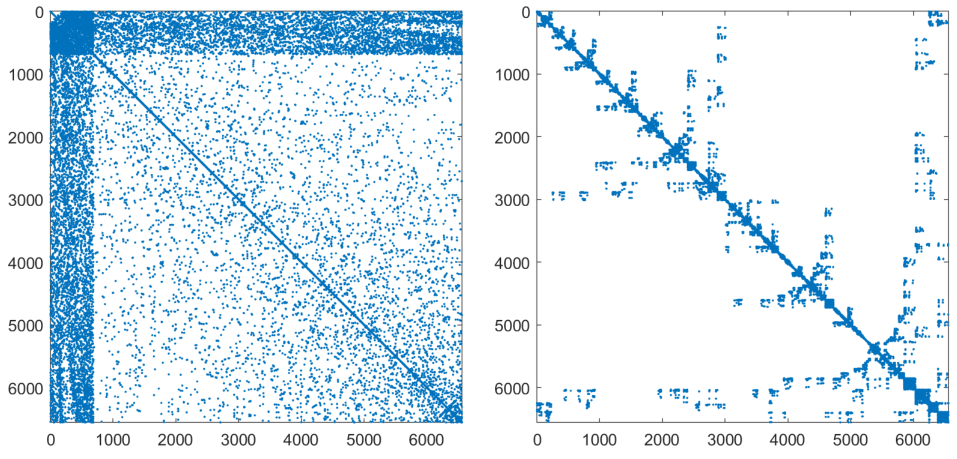
$$\sum_{j \in N_i} w_{ij} = 1. \tag{2.3}$$

In Sections 2.3.1 and 2.3.2, we discuss two approaches for calculating the weights. Before proceeding further, let us borrow some additional notation from [28]. Let x_I and y_I contain the initial x- and y-coordinates of the interior nodes, and let x_B and y_B contain the initial positions of the x- and y-coordinates of the boundary nodes. Then $[x_I, y_I]$ and $[x_B, y_B]$ contain the initial positions of the interior and boundary nodes, respectively. Denote as D_I the matrix containing all of the weights corresponding to the interior neighbors. Similarly, denote as D_B the matrix containing all of the weights corresponding to boundary neighbors. Using this notation, we can express (2.1)-(2.3) as:

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

After calculating the weights, a user-defined deformation is applied to the boundary nodes. This deformation can result from an experiment, or the application of a mapping (i.e., discrete set of points or from continuous motion). The new positions of the boundary nodes are denoted as $[\hat{x}_B, \hat{y}_B]$. The final step is to solve for the new locations of the interior nodes $[\hat{x}_I, \hat{y}_I]$ (while leaving node connectivity unchanged) by solving a linear system of equations using the weights and the new boundary positions from steps one and two, respectively. In particular, we solve the following global linear system, which is a multiple right-hand side (RHS) problem:

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



(a) Natural Ordering (b) Column AMD Ordering

Figure 2.1: Sparsity plots: (a) the natural ordering, (b) the matrix after reordering with column AMD.

In this global linear system, each row in D_I contains information relating the interior node to its neighbors. This construction results in a matrix that is very sparse with irregular structure. For this reason, we solve (2.5) with a sparse LU factorization. To give the matrix a more favorable structure (i.e., to create less fill-in in the factorization), we apply the column approximate minimum degree (AMD) reordering provided by Eigen [11]. In Fig. 2.1, we show sparsity plots of the weight matrix from the third example in Section 2.4. In Alg. 1, we provide a pseudocode description of our warping framework.

Algorithm 1 Pseudocode for our warping framework

```
1. Compute weights
for each interior node $i$ do
    calculate weights using, e.g., Alg. 2 or Alg. 3
end for
2. Apply boundary deformation
3. Solve Eq. (2.5) for new positions of interior nodes
```

2.3.1 Affine Weights

To simplify our discussion in this section, we will express (2.1)-(2.3) for each interior node i as $Aw_i = b$ in the following way:

$$\begin{bmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & y_n \\ 1 & 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} w_{i1} \\ w_{i2} \\ \vdots \\ w_{in} \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix},$$

where $n = |N_i|$. Based on the set of neighbors, this linear system will be underdetermined (i.e., A = $(d+1) \times n$ with $(d+1) < n$, where d is two or three) in general. If we assume that A has full rank, we can find one solution to our problem by requiring that w_i has the smallest Euclidean norm of any solution. This results in the following optimization problem:

$$\min_{w_{ij}, j \in N_i} \sum_{j \in N_i} w_{ij}^2 \tag{2.6}$$

$$\text{subject to } \sum_{j \in N_i} w_{ij} x_j = x_i \tag{2.7}$$

$$\sum_{j \in N_i} w_{ij} y_j = y_i \tag{2.8}$$

$$\sum_{j \in N_i} w_{ij} = 1. \tag{2.9}$$

In the current form, this problem can be solved by a nonlinear programming method. Alternatively, we can reformulate the problem to solve it using a QR factorization. This removes the need for an interative nonlinear optimization method whose convergence behavior would be dependent on a variety of factors (e.g., starting position, convergence tolerance, etc). The following derivation for the problem reformulation can be found in [32], but we reproduce it here for convenience. From the Karush-Kuhn-Tucker (KKT) theory [20], the following conditions must be satisfied for a local solution (w^*, λ^*) to be optimal:

$$\nabla_w \mathscr{L}(w^*, \lambda^*) = 0 \text{ (stationarity)} \tag{2.10}$$

$$Aw^* - b = 0 \text{ (feasibility)} \tag{2.11}$$

$$\lambda^*(Aw^* - b) = 0 \text{ (complementarity)}, \tag{2.12}$$

where the Lagrangian $\mathscr{L}(w, \lambda) = w^T w - \lambda^T (Aw - b)$, and λ are the Lagrange multipliers.

Using (2.10)-(2.12), the following local solution pair (w^*, λ^*) can be found as follows:

$$\begin{aligned}
\nabla_w \mathscr{L}(w, \lambda) &= 2w - A^T \lambda \\
\nabla_w \mathscr{L}(w^*, \lambda^*) = 0 &\Rightarrow w^* = \frac{1}{2} A^T \lambda^* \\
Aw^* - b = 0 &\Rightarrow A(\frac{1}{2} A^T \lambda^*) - b = 0 \\
&\Rightarrow \lambda^* = 2(AA^T)^{-1} b \\
&\Rightarrow w^* = \frac{1}{2} A^T \lambda^* \\
&= \frac{1}{2} A^T 2(AA^T)^{-1} b \\
&= A^T (AA^T)^{-1} b.
\end{aligned}$$

Although we have shown that (w^*, λ^*) is a stationary point, to claim that it is a minimum, we must investigate $\nabla^2_{ww} \mathscr{L}(w^*, \lambda^*)$:

$$\begin{aligned}
\nabla^2_{ww} \mathscr{L}(w, \lambda) &= 2I_{|N_i| \times |N_i|} \\
\nabla^2_{ww} \mathscr{L}(w^*, \lambda^*) &= 2I_{|N_i| \times |N_i|}.
\end{aligned}$$

From the second-order sufficient conditions, if w^* satisfies (2.10)-(2.12) and the following condition is satisfied:

$$z^T \nabla^2_{ww} \mathscr{L}(w^*, \lambda^*) z > 0, \text{ for all } z \in C(w^*, \lambda^*), z \neq 0, \tag{2.13}$$

where $C(w^*, \lambda^*) = \{z \mid \nabla_w \ c(w^*)^T z = 0\}$ is the critical cone and $c(w) = Aw - b$, then our local solution is a minimum. Since $\nabla^2_{ww} \ \mathscr{L}(w^*, \lambda^*)$ is symmetric positive definite, the inequality in (2.13) is satisfied for any choice of z. Thus we can conclude that our solution w^* is a strict local solution of (2.6)-(2.9).

Now that we have verified that w^* is the solution, we describe how to calculate it via a reduced QR factorization. Suppose that $A^T = QR$, where $Q_{n\times m}$ has orthogonal columns and $R_{m\times m}$ is upper triangular. Substituting in the QR factorization of A^T into w^*, we get the following:

$$\begin{aligned}
w^* &= A^+ b \\
&= A^T(AA^T)^{-1}b \\
&= QR(R^TQ^TQR)^{-1}b \\
&= QR(R^TR)^{-1}b \\
&= QRR^{-1}R^{-T}b \\
&= QR^{-T}b
\end{aligned}$$

Rewriting this in linear system form, we have:

$$R^TQ^Tw^* = b.$$

If we let $t = Q^Tw^*$, then $R^Tt = b$ and $w^* = Qt$. Thus calculating w^* involves computing a QR decomposition of A^T, solving the lower triangular system $R^Tt = b$ using forward substitution, and calculating the matrix-vector product Qt. In Alg. 2, we give the pseudocode for computing affine weights.

Algorithm 2 Pseudocode for computing affine weights

```
1. Compute $A^T = QR$ using a reduced QR factorization.
2. Solve $R^Tt = b$ using forward substitution.
3. Set $w^* = Qt$.
```

2.3.2 Convex Weights

If we wish to enforce that the weights are strictly positive (corresponding to interpolation within the convex hull of the neighboring set), we can formulate the following optimization problem as described in [29]:

$$\min_{w_{ij}, j \in N_i} -\sum_{j \in N_i} \log(w_{ij}) \tag{2.14}$$

$$\text{subject to } w_{ij} > 0 \tag{2.15}$$

$$\sum_{j \in N_i} w_{ij} x_j = x_i \tag{2.16}$$

$$\sum_{j \in N_i} w_{ij} y_j = y_i \tag{2.17}$$

$$\sum_{j \in N_i} w_{ij} = 1. \tag{2.18}$$

Since the convex weights can be interpreted as interpolation within the convex hull defined by the neighbors, the interior node that we are representing must be inside of the convex hull of the neighboring set of nodes for the problem to be well-posed. The log barrier function serves two important purposes in this formulation. In particular, it enforces the inequality constraint since the -log function approaches infinity as its input function approaches 0. For this reason, the -log serves as a barrier which enforces that $w_{ij} > 0$. We are not the first group to use a log barrier in a meshing context. For low-order meshes, Shontz and Vavasis used a log-barrier in LBWARP [29], which served as the basis for our convex weight problem formulation. In a high-order mesh generation context, Toulorge et al. [34] use a log barrier to prevent element Jacobians from becoming too small as part of an objective function for untangling invalid curved elements. As posed in (2.14)-(2.18), this is a strictly convex optimization problem for which there is a unique global minimum. By starting with an initial feasible point, this optimization problem can be solved using the equality-constrained Newton method, which will maintain feasibility at each iterate [2].

$$\begin{bmatrix} \nabla^2 f(w_k) & -A(w_k)^T \\ A(w_k)^T & 0 \end{bmatrix} \begin{bmatrix} p_k \\ p_\lambda \end{bmatrix} = \begin{bmatrix} -\nabla f(w_k) \\ 0 \end{bmatrix} \tag{2.19}$$

To find an initial feasible point, we can use the scheme described in [31]. That is, we can choose three of the node's neighbors and write the node as a convex combination of the these neighbors, where the remaining neighbors are given a weight of ε, a small positive constant. Depending on the number of neighbors, this might require ε values of 10^{-3} or smaller. This would result in some entries of $\nabla^2 f(w_k)$ being 10^6 or smaller, which results in a poorly-conditioned system. While we can address this using a simple preconditioner (e.g., a diagonal preconditioner), we instead explore an alternative formulation of the Newton equations that does not require a feasible starting point. This alternative formulation has the following form:

$$\begin{bmatrix} \nabla^2 f(w_k) & -A(w_k)^T \\ A(w_k)^T & 0 \end{bmatrix} \begin{bmatrix} p_k \\ p_\lambda \end{bmatrix} = - \begin{bmatrix} \nabla f(w_k) \\ Aw_k - b \end{bmatrix}. \tag{2.20}$$

The key difference between (2.19) and (2.20) is the lower block of the right-hand side which gives the residual with respect to the equality constraints (i.e., measures the feasibility of w_k with respect to the equality constraints). This alternative formulation can be solved using the infeasible start Newton method described in chapter 10 of [2]. Recall that we mentioned earlier that -log serves as a barrier, with this in mind our starting position (i.e., the starting weights) must be positive or the optimization cannot cross the barrier. However, the starting position can be infeasible with respect to the equality constraints. In Alg. 3, we give the pseudocode for computing convex weights. One step in the algorithm that warrants further discussion is step 2. In this step, we scale back the descent direction p_k to ensure that each entry in $w_k + \alpha p_k$ is strictly positive. This step is required to avoid division by zero in the evaluation of ∇f. In practice, this safeguard was only necessary when our initial iterate was far from the solution.

Algorithm 3 Pseudocode for computing convex weights

```
residual$(x,y) = [\nabla f(x) + A^T y, Ax - b]$ {the dual and primal residuals, respectively}
while ctr $< 500$ and $||\text{residual}(w_k, \lambda_k)||_2 > 10^{-8}$ do
    1. Solve Eq. 2.20 for $p_k$ and $p_\lambda$.
    $\alpha = 1.0$
    2. Scale $p_k$ by $\alpha$ until all $(w_k + \alpha p_k) > 0$ {to stay in the domain of $\nabla f(x)$}
    while any$(w_k + \alpha p_k) < 0$ do
        $\alpha = 0.8\alpha$
    end while
    3. Backtrack on the norm of the residual
    while $||\text{residual}(w_k + \alpha p_k, \lambda_k + \alpha p_\lambda)||_2 > (1 - 0.25\alpha)||\text{residual}(w_k, \lambda_k)||_2$ do
        $\alpha = 0.75\alpha$
    end while
    $w_{k+1} = w_k + \alpha p_k$, $\lambda_{k+1} = \lambda_k + \alpha p_\lambda$
    ctr = ctr + 1
end while
```

2.4 Numerical Experiments

In this section, we demonstrate the results from applying our framework on a variety of mesh deformations in two and three dimensions. Before discussing our numerical examples, we will start by showing scaling results for the two weighting schemes and the multiple right-hand side linear solve for our framework. In Figs. 2.2 and 2.3, we show scaling plots of the runtime in seconds versus the number of interior nodes (both shown on a log scale) for 2D and 3D, including both second-order and third-order elements. For these tests, we used a simple circle and sphere geometries for two and three dimensions, respectively. In each case, we refined the low-order mesh multiple times, and then enriched each of those refined meshes to become a high-order mesh. For these geometries, we decreased the element sizes with the goal of approximately doubling the number of elements in the previous level. In Tables 2.1 and 2.2, we show the total number of mesh elements, the total number of nodes (including low- and high-order nodes), and the total number of interior nodes (including low- and high-order nodes). As we can see in Fig. 2.2, the affine weighting scheme for a given element order is faster for smaller problem sizes (i.e., fewer interior nodes). However, once the problem size is large enough, the affine and convex weight computations require a similar amount of time. In Fig. 2.3 we can see that the choice of weighting

	Circle				
		P2		P3	
Refinement Level	**Total Num Elems**	**Total Num Nodes**	**Num Int Nodes**	**Total Num Nodes**	**Num Int Nodes**
0	437	920	830	2,035	1,900
1	842	1,749	1,621	3,886	3,694
2	1,747	3,584	3,406	7,996	7,729
3	3,725	7,586	7,316	16,966	16,561
4	7,848	15,891	15,503	35,608	35,026
5	14,914	30,091	29,567	67,507	66,721
6	28,287	56,320	56,084	127,579	126,466
7	58,221	116,966	115,920	262,780	261,211
8	117,001	234,742	233,264	527,614	525,397

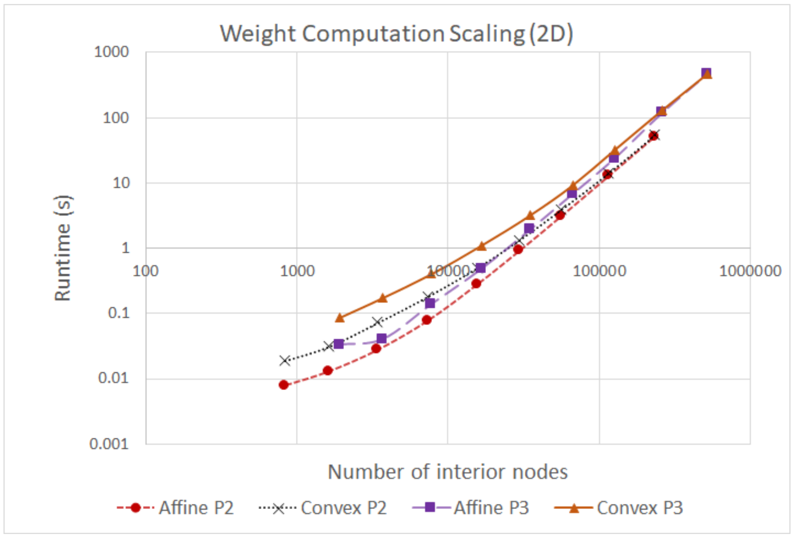
Table 2.1: The total number of elements, total number of nodes (including low- and high-order nodes), and number of interior nodes (including low- and high-order) for the second and third-order meshes at each level of refinement on the circle geometry.

scheme has minimal impact on the multiple RHS linear solve.

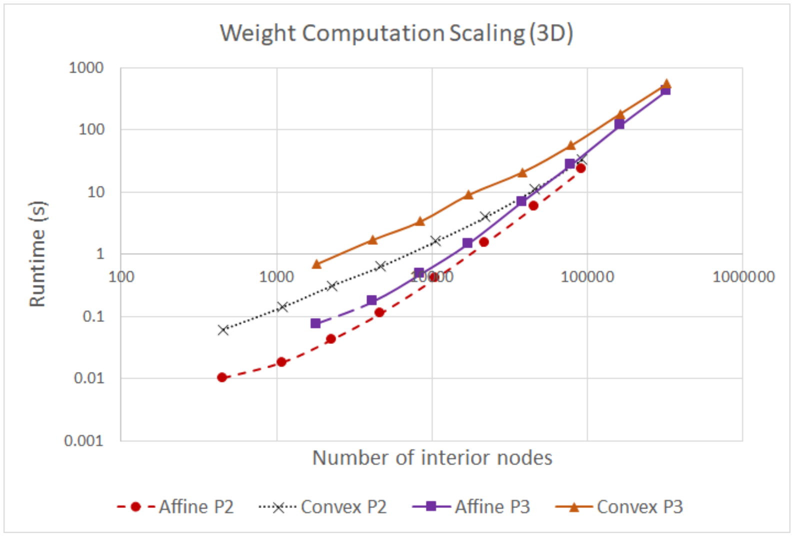
For both 2D and 3D, we explore smaller deformations, like the ones associated with the boundary curving step during the typical high-order mesh generation process, and larger deformations that might be encountered in mesh warping or moving mesh applications. In Tab. 2.3, we list the number of elements, element order, and number of interior nodes for each example. We also list the execution times for our method (excluding I/O) in Table 2.4. The method was implemented in C++, and the wall-clock times were measured on the KU Community Cluster at the University of Kansas. The specific nodes used have 128GB of RAM and Intel Xeon E5-2680 v3 processors. In addition, for each example, we show the initial mesh, the mesh resulting from our methods after applying a deformation and updating the nodal positions, and the element distortion values as measured by the scaled Jacobian ([3]):

$$\text{scaled Jacobian} = \frac{\min J(\xi)}{\max J(\xi)},$$

where ξ are coordinates in the reference element, and J is the determinant of the Jacobian of the mapping from reference to physical space. This measure has a maximum value of 1, indicating

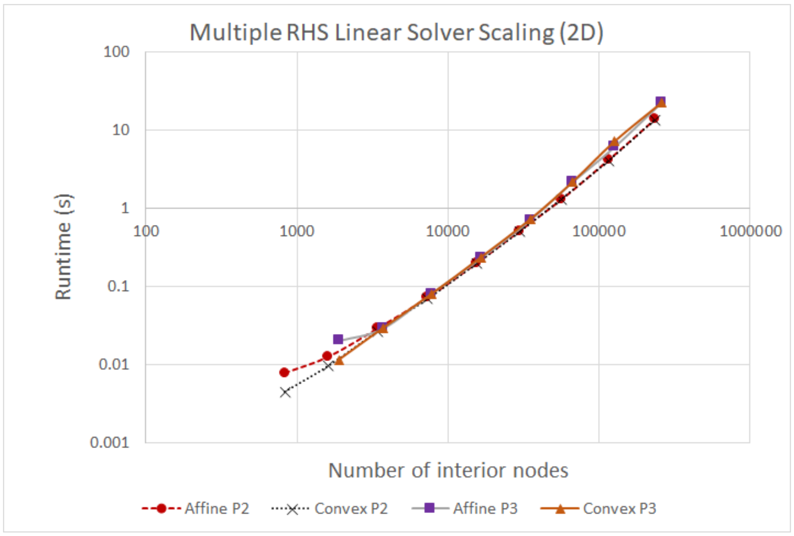


(a)

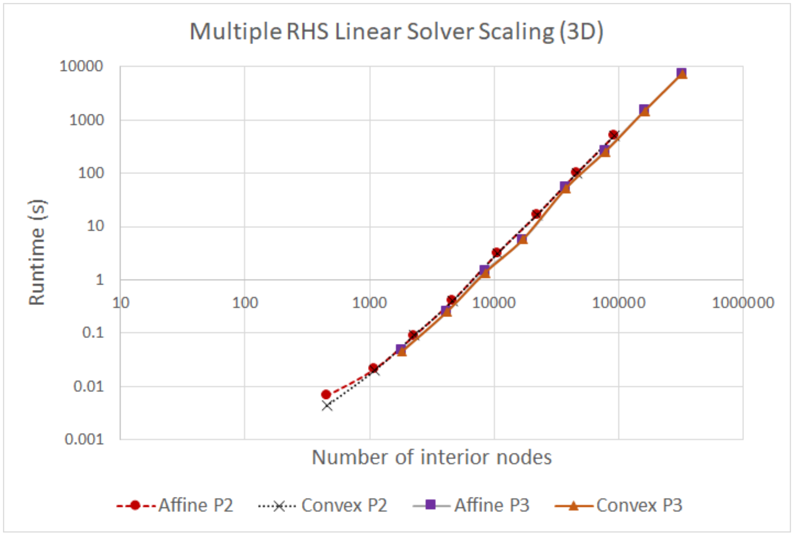


(b)

Figure 2.2: Scaling plots of the affine and convex weight computations on the (a) circle and (b) sphere geometries at different levels of refinement.



(a)



(b)

Figure 2.3: Scaling plots of the multiple right-hand side linear solver on the (a) circle and (b) sphere geometries at different levels of refinement.

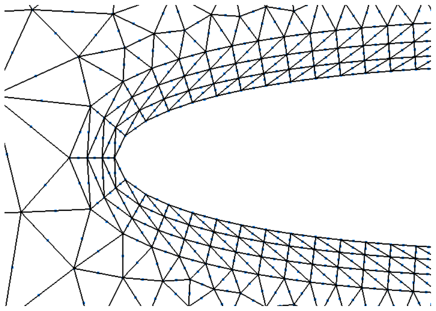
	Sphere				
		P2		**P3**	
Refinement Level	**Total Num Elems**	**Total Num Nodes**	**Num Int Nodes**	**Total Num Nodes**	**Num Int Nodes**
0	536	1,028	454	3,096	1,807
1	1,161	2,056	1,094	6,346	4,184
2	2,231	3,716	2,274	11,691	8,449
3	4,393	7,080	4,662	22,508	17,070
4	9,483	14,668	10,554	47,251	37,997
5	19,208	28,903	22,053	93,942	78,532
6	38,618	56,320	45,890	184,935	161,470
7	76,338	109,333	92,475	361,165	323,237

Table 2.2: The total number of elements, total number of nodes (including low- and high-order nodes), and number of interior nodes (including low- and high-order) for the second and third-order meshes at each level of refinement on the sphere geometry.

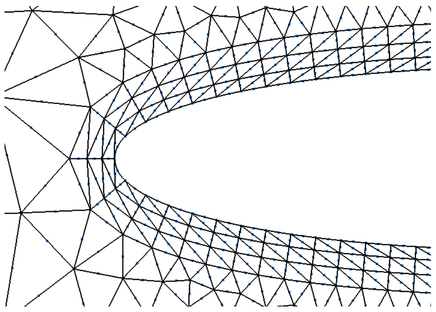
that the element is straight-sided. Negative values indicate that the element is tangled. When reporting the mesh distortion, we list the minimum distortion, average distortion, and maximum distortion values. Distortion values are reported in Tab. 2.5. All mesh distortion evaluations and visualizations were done using Gmsh [10, 12, 23].

Our first example starts with a straight-sided second-order boundary layer mesh around an NACA0012 airfoil. The deformation that is applied moves the second-order nodes onto the curved boundaries. In Fig. 2.4, we show the initial mesh, and the meshes resulting from each type of weighted combination. For this example, the two weighting schemes resulted in meshes with similar distortion values. Their minimum distortion values are 0.163 and 0.169 for the affine and convex combinations, respectively. However, the convex weight computation required roughly twice the amount of time as the affine weight computation.

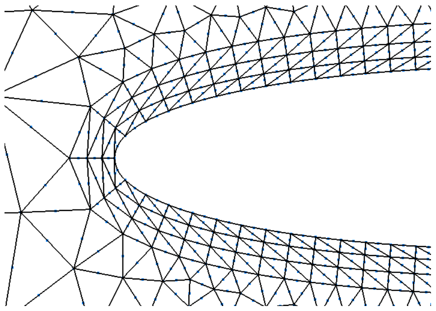
For the remainder of the 2D examples, we focus on larger deformations. We start by moving a circle through a rectangular domain. In this example, the circle was moved along the x-axis by 100% of the radius of the circle. In Fig. 2.5, we show the initial mesh and the meshes resulting from each type of weighted combination. In this example, the two approaches resulted in minimum distortion values of 0.453 and 0.454, for the affine and convex weighting schemes, respectively.



(a)

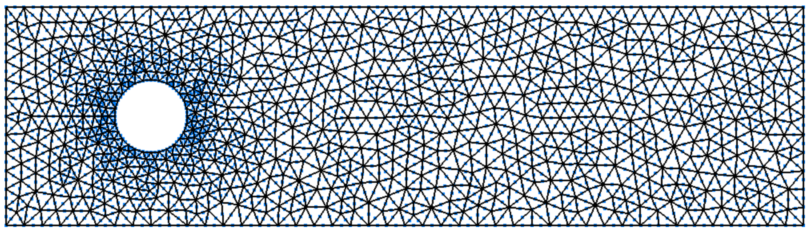


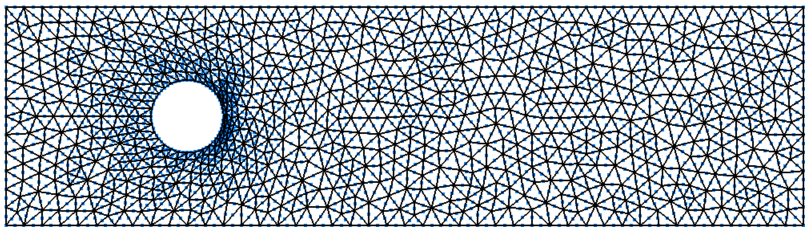
(b)



(c)

Figure 2.4: NACA0012 airfoil example: (a) the initial straight-sided second-order mesh, (b) the mesh resulting from our affine method, and (c) the mesh resulting from our convex method.


(a)


(b)

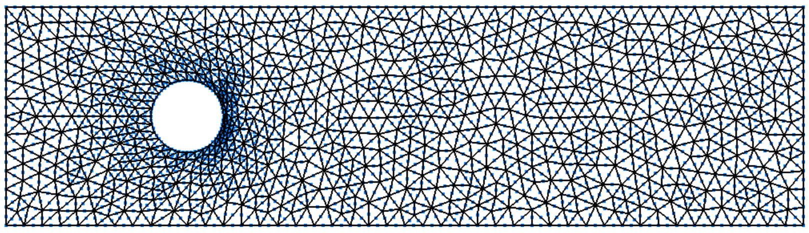

(c)

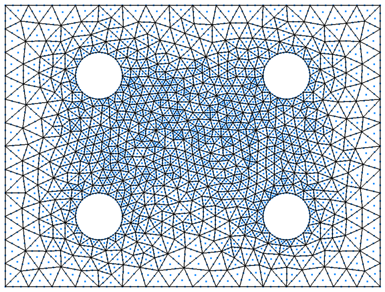
Figure 2.5: Moving circle example: (a) the initial second-order mesh, (b) the deformed mesh using our affine method, and (c) the deformed mesh using our convex method.

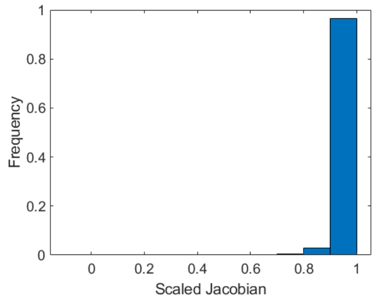
For examples three and four, we use an initial mesh with four circles in the domain. For the third example, we move the X and Y coordinates of the circles toward the center of the domain by 37.5% of the radius r. For the fourth example, we move the X and Y coordinates of each circle by a random amount between -0.65*r and 0.65*r. In contrast with our previous examples, the convex weighting scheme performed noticeably better than the affine weighting scheme with respect to element distortion. In particular, the minimum distortions for each example using the affine weighting scheme were 0.109 and 0.124, respectively, while the convex weighting scheme resulted in a minimum of 0.137 for both examples. In Fig. 2.6 and Fig. 2.7, we show the initial meshes and the meshes resulting from each type of weighted combination. In contrast with the first two examples, for these larger deformations, the convex weighting scheme resulted in meshes with lower distortion than the affine combination scheme.

Our fifth example starts with a straight-sided second-order mesh of a screw. The deformation that is applied moves the second-order nodes onto the curved boundaries. In Fig. 2.8, we show the initial mesh and the meshes resulting from each type of weighted combination. For this example, the two weighting schemes resulted in meshes with a minimum distortion of 0.008. The convex scheme resulted in a better average and maximum distortion of 0.901 and 0.998, respectively, compared to an average of 0.894 and maximum of 0.996 for the affine scheme.

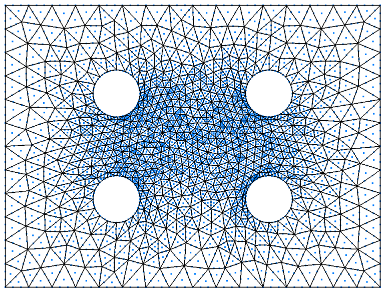
For the remainder of the 3D examples, we focus on larger deformations. We start by moving a sphere through a cube domain. In this example, the sphere moved along the x-axis by 30% of the radius of the sphere. In Fig. 2.9, we show the initial mesh, and the meshes resulting from each type of weighted combination. Each mesh is shown with a cutting plane through the middle to allow visualization of the interior. We also omitted the mesh nodes to simplify the visualization. In this example, the affine weighting scheme resulted in a mesh with a minimum distortion value of 0.173, while the convex scheme resulted in a minimum of 0.543. Although the convex weight computation took roughly twice as long, it resulted in much better elements.

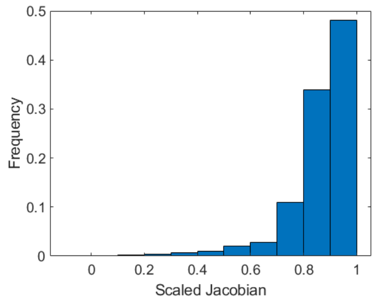
For example seven, we use an initial mesh with four spheres in the domain. In this example, we move the x-, y-, and z- coordinates of each sphere by a random amount between -0.4*r and


(a)

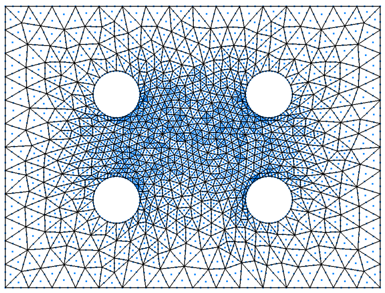


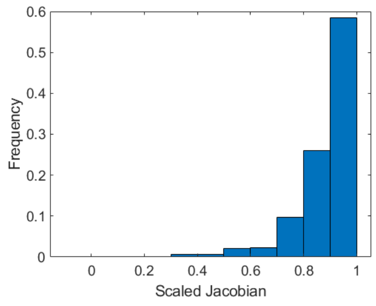
(b)


(c)



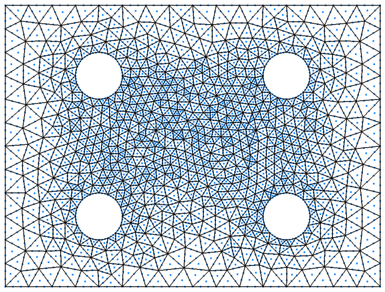
(d)

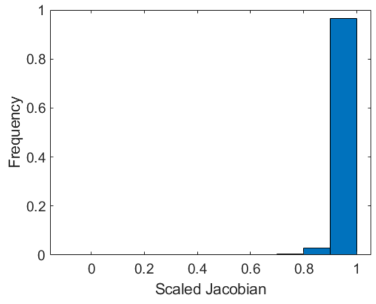

(e)



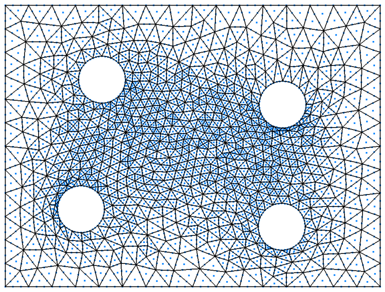
(f)

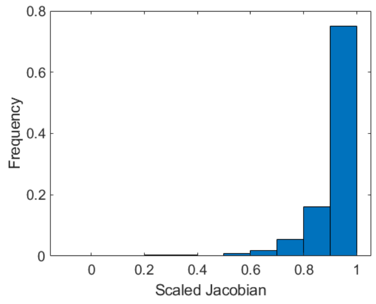
Figure 2.6: Moving circles 1 example: (a) the initial third-order mesh, (c) the deformed mesh using affine combinations after moving the circles closer together, (e) the deformed mesh using convex combinations, and (b,d,f) histograms of the distortion values for each mesh.


(a)

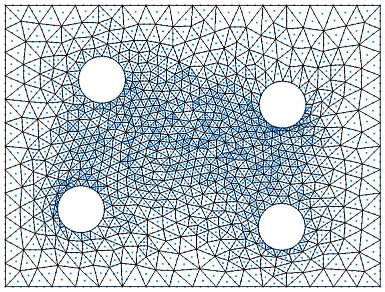


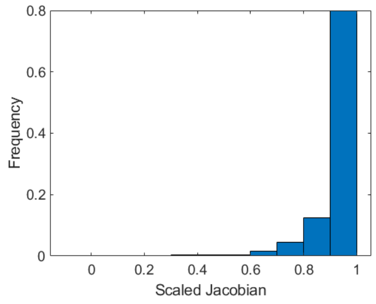
(b)


(c)



(d)


(e)



(f)

Figure 2.7: Moving circles 2 example: (a) the initial third-order mesh, (c) the deformed mesh using affine combinations after moving the circles by a random amount, (e) the deformed mesh using convex combinations, and (b,d,f) histograms of the distortion values for each mesh.

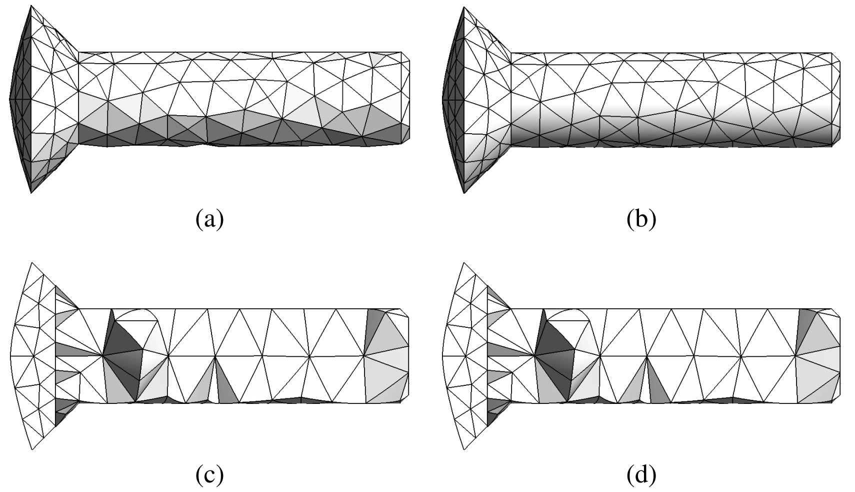
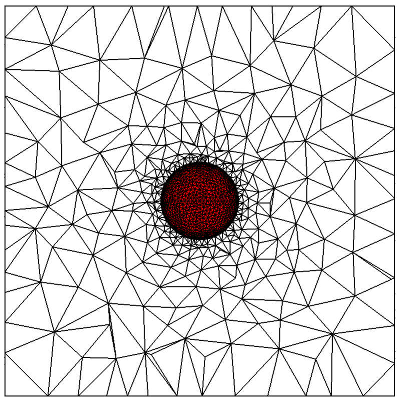


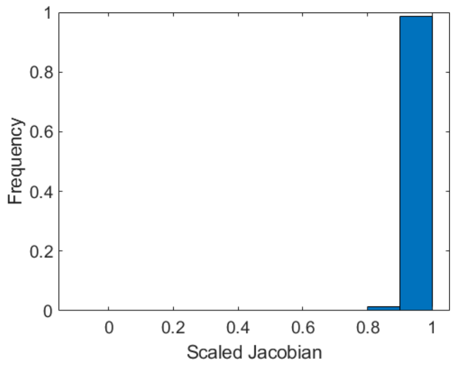
Figure 2.8: Screw example: (a) the initial second-order surface mesh, (b) the deformed surface, and (c,d) a cutting plane through the volume of the deformed meshes using affine and convex combinations, respectively.

0.4*r. In Fig. 2.10, we show the initial mesh, and the meshes resulting from each type of weighted combination (with high-order nodes omitted). In each case, we show a cross section of the domain to allow visualization of the interior. Once again, the convex weighting scheme resulted in a mesh with a better minimum distortion value of 0.153, compared to 0.120 with the affine scheme.

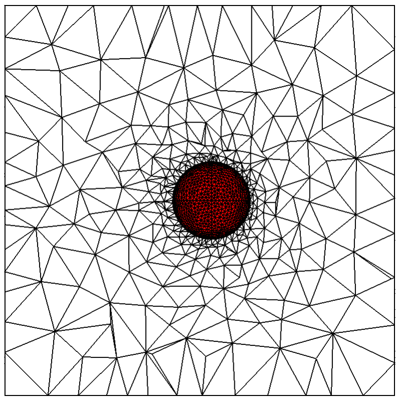
In our final two examples, we apply deformations to cylindrical shell geometries. For the first shell, we increase the diameter of each cylinder, increase the thickness of the shell wall, and reduce the length of the shell. For this example, the two schemes resulted in similar minimum distortion values of 0.315 and 0.318 for the affine and convex schemes, respectively. However, the convex weight computation required roughly three times as long. For the second shell example, we deform the shell into the shape of a sin wave. In this example, the affine scheme resulted in a mesh with two tangled elements and a minimum distortion value of -0.175, while the convex scheme maintained element validity with a minimum distortion value of 0.085. Although computing the affine weights was significantly faster, this savings would be mitigated by the need for an untangling pass to recover mesh element validity.

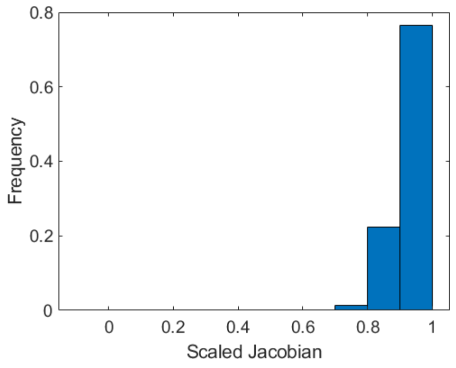
Taking a closer look at the information in Tables 2.3 to 2.5, some patterns start to emerge. First, for smaller deformations, the two weighting schemes result in warped meshes with similar levels


(a)

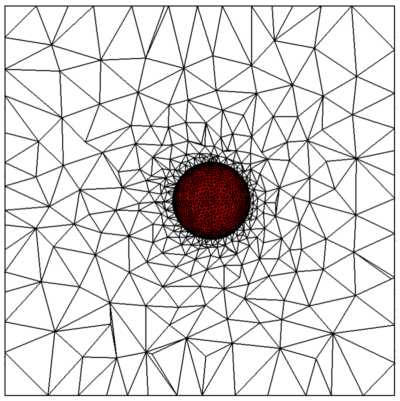


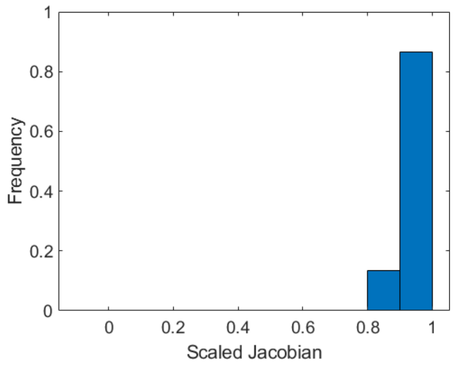
(b)


(c)



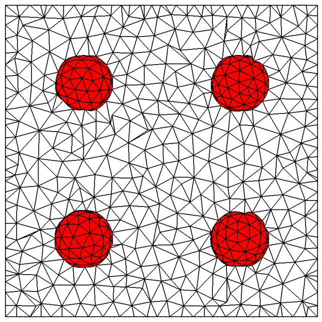
(d)

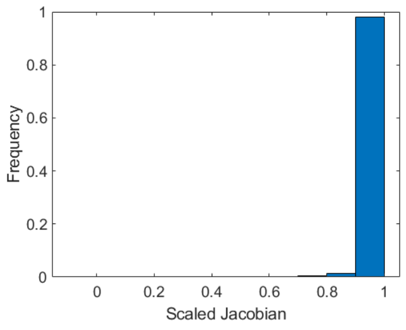

(e)



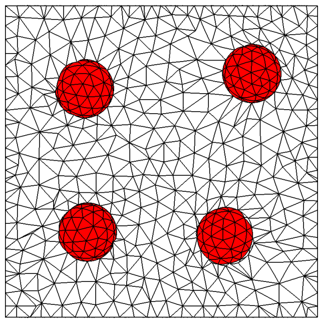
(f)

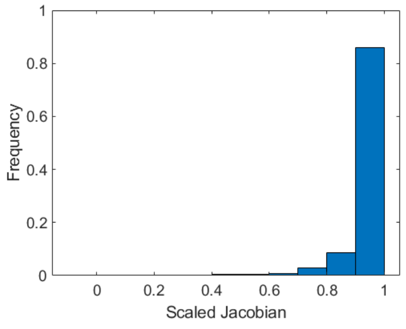
Figure 2.9: Moving sphere example: (a) the initial third-order mesh, (c) the deformed mesh using affine combinations after moving the sphere, (e) the deformed mesh using convex combinations, and (b,d,f) histograms of the distortion values for each mesh.


(a)

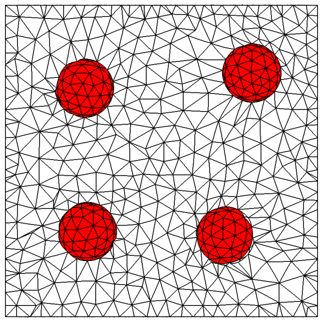


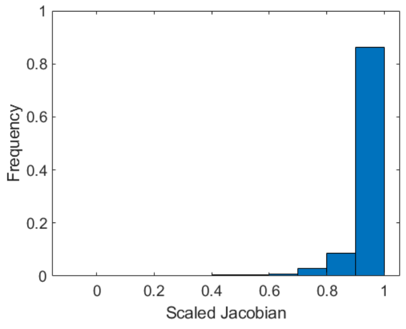
(b)


(c)



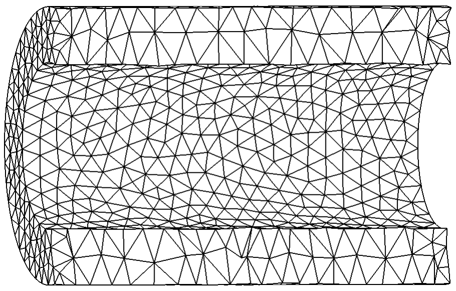
(d)


(e)

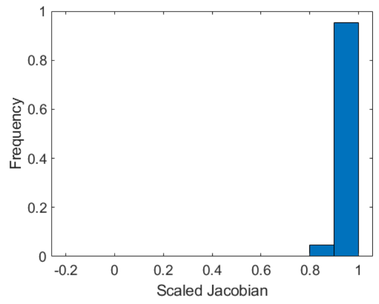


(f)

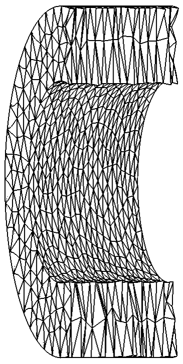
Figure 2.10: Moving spheres example: (a) the initial second-order mesh, (c) the deformed mesh using affine combinations after moving the spheres by a random amount, (e) the deformed mesh using convex combinations, and (b,d,f) histograms of the distortion values for each mesh.



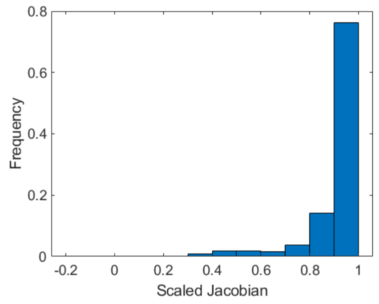
(a)



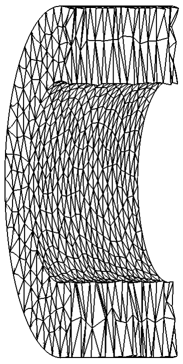
(b)



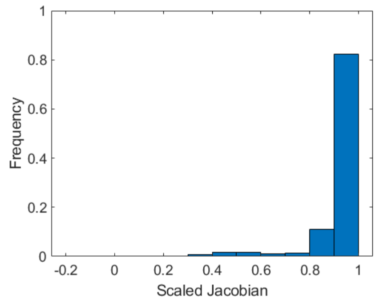
(c)



(d)

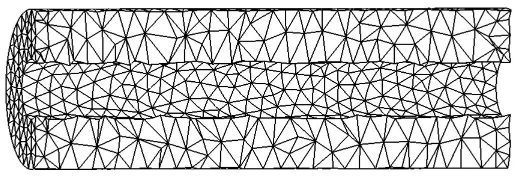


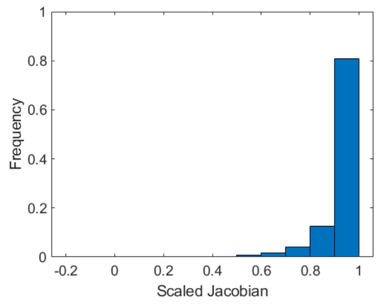
(e)



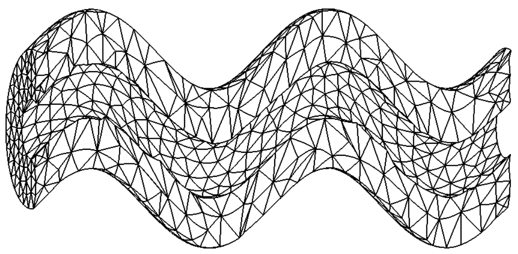
(f)

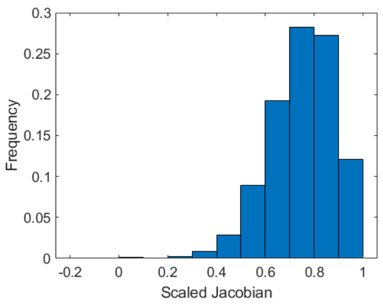
Figure 2.11: Shell 1 example: (a) the initial second-order mesh, (c) the deformed mesh using affine combinations, (e) the deformed mesh using convex combinations, and (b,d,f) histograms of the distortion values for each mesh.


(a)

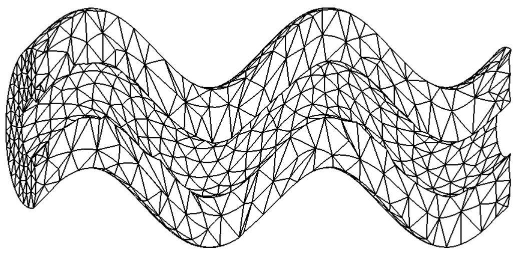


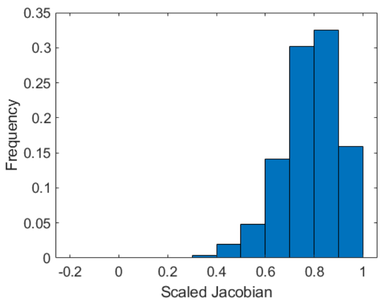
(b)


(c)



(d)


(e)



(f)

Figure 2.12: Shell 2 example: (a) the initial second-order mesh, (c) the deformed mesh using affine combinations, (e) the deformed mesh using convex combinations, and (b,d,f) histograms of the distortion values for each mesh.

Example	**Num of Elems**	**Mesh Order**	**Num of Int Nodes**
NACA0012 Airfoil	2,384	2	4,610
Moving Circle	1,686	2	3,232
Moving Circles 1	1,498	3	6,558
Moving Circles 2	1,498	3	6,558
Screw	615	2	402
Moving Sphere	19,207	3	76,175
Moving Spheres	51,533	2	59,104
Shell 1	8,914	3	31,411
Shell 2	7,904	3	28,277

Table 2.3: The number of elements, mesh order, and number of interior nodes for each example.

	Runtime (s)				
Example	**Neigh Set Comp**	**Affine Weight Comp**	**Multiple RHS Solve**	**Convex Weight Comp**	**Multiple RHS Solve**
Airfoil	0.0052	0.0412	0.0631	0.1060	0.0628
Moving Circle	0.0034	0.0224	0.0346	0.0700	0.0342
Moving Circles 1	0.0085	0.0970	0.1226	0.3142	0.1220
Moving Circles 2	0.0085	0.0970	0.1220	0.3156	0.1222
Screw	0.0022	0.0078	0.0043	0.0418	0.0030
Moving Sphere	0.3917	23.9505	310.7020	50.2970	310.6000
Moving Spheres	0.2435	8.3268	168.9980	13.0349	169.2470
Shell 1	0.1506	4.1448	6.2183	12.9734	6.2156
Shell 2	0.1357	3.3963	4.6712	11.4073	4.6667

Table 2.4: The wall clock times for the neighboring set computation, weight generation, and the multiple right-hand side linear solve phases.

	Distortion					
	Affine			**Convex**		
Example	**Min**	**Avg**	**Max**	**Min**	**Avg**	**Max**
Airfoil	0.163	0.998	1.000	0.169	0.998	1.000
Moving Circle	0.453	0.952	1.000	0.454	0.953	1.000
Moving Circles 1	0.109	0.867	0.997	0.137	0.881	0.997
Moving Circles 2	0.124	0.915	0.997	0.137	0.923	0.997
Screw	0.008	0.894	0.996	0.008	0.901	0.998
Moving Sphere	0.173	0.934	1.000	0.543	0.946	1.000
Moving Spheres	0.120	0.966	1.000	0.153	0.967	1.000
Cyl Shell1	0.315	0.908	0.995	0.318	0.918	0.995
Cyl Shell2	-0.175	0.749	0.993	0.085	0.783	0.993

Table 2.5: The minimum, average, and maximum distortion for the affine and convex weighting schemes for each example.

of distortion, but the affine scheme is faster on coarser meshes. As the deformations get larger, the affine scheme tends to create elements with greater amounts of distortion than the convex scheme, and also results in a comparable weight computation time. With these observations in mind, using the affine scheme is a good choice for small deformations on coarse meshes, but for larger deformations, the convex scheme will likely result in a better mesh (as measured by element distortion) without a significant increase in additional runtime.

2.5 Concluding Remarks and Future Work

We have presented an optimization-based framework for high-order mesh warping based on optimal weighted combinations of nodal positions. While we presented two geometry-based methods within our framework for weight computation, our framework is flexible and allows the inclusion of alternative weight computation schemes that could be problem-specific. We have also explored the use of our framework on larger deformations that might be encountered in mesh warping or moving mesh applications. While our runtimes are reasonable, our scaling tests have demonstrated the need for a parallel implementation similar to the low-order parallel LBWARP described in [21] for large mesh sizes. In particular, there is great potential for computing each set of weights in parallel because they are independent. In addition, leveraging a parallel linear solver for the multiple

RHS linear solve would further improve the scaling capabilities for large problems.

Our future work will include the exploration of iterative linear solvers, as well as extending the method to additional orders beyond third-order. In addition, we might explore alternative definitions of neighboring sets, in order to improve sparsity for example, by carefully reducing the number of neighbors to discourage highly distorted elements.

Acknowledgments

The work of the first author was funded in part by the Madison and Lila Self Graduate Fellowship and NSF CCF grant 1717894. The work of the second author was supported in part by NSF grants CCF 1717894 and OAC 1808553. The work of both authors was supported through instrumentation funded by the Army Research Office under contract W911NF-15-1-0377.

Chapter 3

An Angular Approach to Untangling High-order Curvilinear Triangular Meshes*

Abstract

To achieve the full potential of high-order numerical methods for solving partial differential equations, the generation of a high-order mesh is required. One particular challenge in the generation of high-order meshes is avoiding invalid (tangled) elements that can occur as a result of moving the nodes from the low-order mesh that lie along the boundary to conform to the true curved boundary. In this paper, we propose a heuristic for correcting tangled second- and third-order meshes. For each interior edge, our method minimizes an objective function based on the unsigned angles of the pair of triangles that share the edge. We present several numerical examples in two dimensions with second- and third-order elements that demonstrate the capabilities of our method for untangling invalid meshes.

3.1 Introduction

The appeal of high-order methods for solving partial differential equations lies in their ability to achieve higher accuracy at a lower cost than low-order methods. One challenge in the adoption of these high-order methods for problems with curved geometries is the lack of robust high-order mesh generation software [36]. More specifically, to fully leverage the accuracy of high-order

*Reprinted by permission from Springer Nature: Springer, Cham. Stees M., Shontz S.M. (2019) An Angular Approach to Untangling High-Order Curvilinear Triangular Meshes. In: Roca X., Loseille A. (eds) 27th International Meshing Roundtable. IMR 2018. Lecture Notes in Computational Science and Engineering, vol 127. Copyright 2019.

methods in the presence of curved geometries, such methods need to be paired with a high-order mesh that correctly reflects the curvature of the geometry, as demonstrated in [1, 17].

The most common approach for high-order mesh generation methods is to transform a coarse linear mesh [3], [6, 7, 8], [18, 19], [9, 22, 24, 25],and [31, 32, 33, 34, 35, 37]. The main challenge of the transformation is obtaining a valid high-order mesh. In general, these methods involve three steps: (1) adding additional nodes to the linear mesh; (2) moving the newly added boundary nodes to conform with the curved geometry, and (3) moving the interior nodes. There are two categories of methods which are especially popular for transforming the initial mesh. The first category involves transforming the mesh based on optimization of an objective function [3], [7, 8],[24, 25], and [26, 31, 32, 33, 34]. Several of the objective functions proposed in this category include a measure of element validity, which allows them to untangle invalid elements [3, 7, 8, 24, 25, 34]. While they do not guarantee successful untangling, many of them are robust. The second category of methods transform the mesh based on the solution to a partial differential equation [6, 18, 22, 37].

In this paper, we describe an optimization-based approach for untangling invalid second- and third-order meshes. The primary goal of this work is to untangle invalid meshes that result from deforming the newly added boundary nodes to conform with the true boundary. Toward that end, we demonstrate our method on several meshes composed of second- and third-order elements that became invalid following the projection of the boundary nodes onto the true boundary. We also explore the untangling of meshes that became invalid as a result of small deformations. The remainder of this paper is organized as follows. In Section 2, we present our new method for high-order mesh untangling. In Section 3, we illustrate the performance of our method on several examples. Finally, in Section 4, we offer concluding remarks and discuss some possibilities for future work.

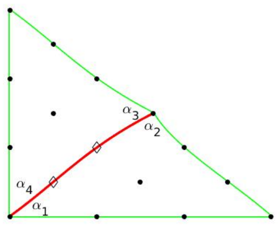


Figure 3.1: A pair of triangles showing the interior edge (red), the free nodes (black diamonds), the fixed nodes (black dots), and the four angles α_i

3.2 Untangling High-order Curvilinear Meshes

In this section, we propose a local edge-based optimization method for untangling high-order curvilinear meshes based on the unsigned angles of curvilinear triangles. For each interior mesh edge, we identify the two triangles that share the edge and compute the distortion of each of the two triangles. For each pair of triangles with a minimum distortion measure less than 0, we solve the following unconstrained optimization problem:

$$x^* = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{i=1}^{4} \alpha_{\mathbf{i}}(\mathbf{x}), \tag{3.1}$$

where

$\alpha_{\mathbf{i}}$	$=$ the i^{th} entry of the vector of the four unsigned angles,
$\mathbf{x}$	$=$ the nodal positions of the high-order nodes that lie on the edge

In Fig. 3.1 we give an example which shows an interior edge in red and the pair of triangles that share that edge in green. We also label the four unsigned angles that are calculated, the nodes that are allowed to move during the optimization (black diamonds), and the nodes that are fixed (black dots).

To better understand the behavior of the objective function, consider the five examples shown in Fig. 3.2. In Fig. 3.2(a) for $\alpha_1 + \alpha_4$ and $\alpha_2 + \alpha_3$, moving the free node (green diamond) will shift the proportion of each term, while leaving the overall sum fixed. In other words, increasing α_1

will cause a corresponding decrease in α_4 while the quantity $\alpha_1 + \alpha_4$ remains the same. Similarly, increasing α_2 will cause a corresponding decrease in α_3 while the quantity $\alpha_2 + \alpha_3$ remains the same. This behavior means that the sum of all four angles cannot be further decreased by moving the free node. Furthermore, this behavior is desirable because patches with no distortion will not be modified since the optimization will not move the free node (as there is no step that will lead to a decrease in the objective function). Fortunately, this behavior holds true as we add minor distortion as well. In Fig. 3.2(b), we moved the bottom node (denoted by a blue square) to increase the distortion of the bottom element. In Fig. 3.2(c), we moved the node slightly further to increase distortion. In both cases, we can see that the overall sum cannot be further decreased by moving the free node. Finally, in Fig. 3.2(d), we move the node to the point that it causes tangling. Now that α_1 is an angle between tangled edges, this angle can be decreased by moving the free node. By decreasing the value of α_1, we decrease the value of $\alpha_1 + \alpha_4$, and thus decrease the overall sum of the four angles. In other words, minimizing our objective function attempts to decrease the value of angles that occur between tangled edges by moving the free node away. In Fig. 3.2(e), we show results of moving the free node to minimize our objective function.

To measure distortion, we use the scaled Jacobian [3]. To solve our unconstrained optimization problem, we use the Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton method decribed in Chapter 6 of [20]. In place of the analytical gradient, we use a 6^{th} order centered finite difference with a step size of 10^{-6}. As our initial Hessian approximation, we use a scaled version of the identity matrix. In Alg. 4 and Alg. 5, we give pseudocode descriptions of our untangling method and optimization method, respectively. Our implementation of the BFGS quasi-Newton method uses a backtracking line search. This backtracking approach based on the Wolfe conditions ensures that the step results in a sufficient decrease in our objective function. In the next section, we discuss how the angles $\alpha_i(x)$ of the curved elements are calculated.

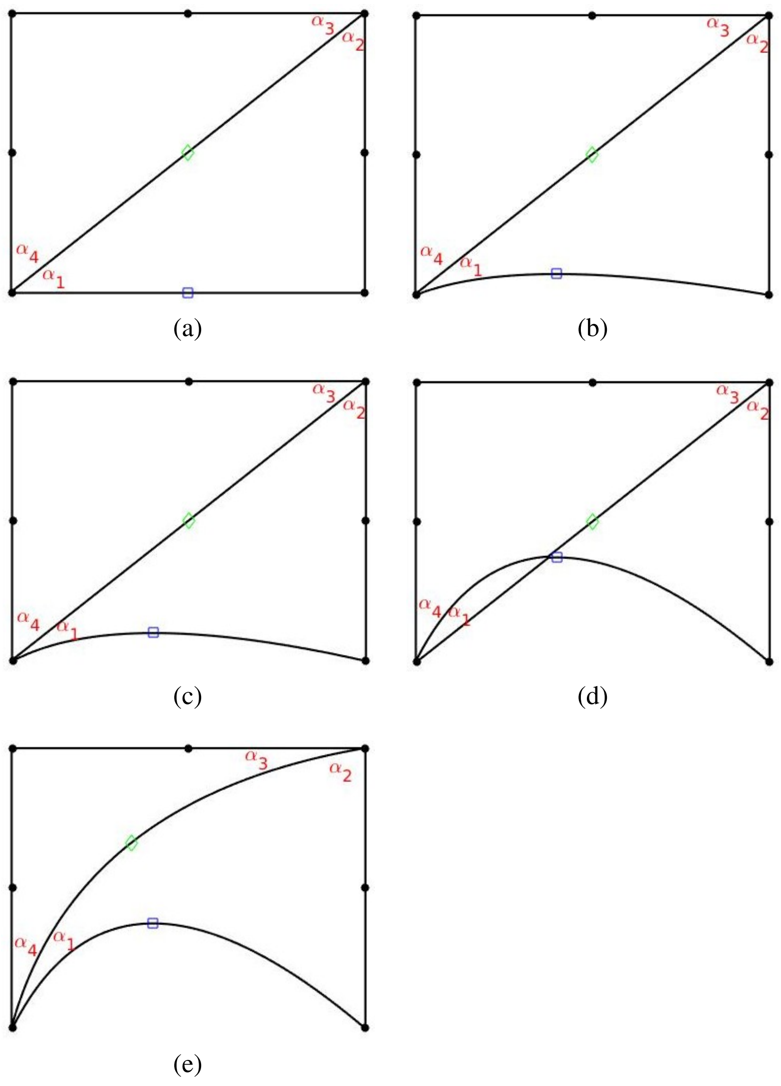


Figure 3.2: A simple patch showing the angles (α_i), the free node (green diamond), and the node that is moved to increase distortion (blue square). In (a), the patch with no distortion is shown. In (b-d), the amount of distortion is gradually increased. In (e), the mesh after applying our method to minimize the sum of the angles is shown.

Algorithm 4 Pseudocode for our edge-based mesh untangling method

```
while there are tangled elements or passes < count do
    for each interior edge $e$ do
        Find the two triangles $t_1$ and $t_2$ with $e$ as a common edge
        Compute the element distortions $ed_1$ and $ed_2$ for $t_1$ and $t_2$, respectively
        if $\min(ed_1, ed_2) < 0$ then
            Solve Eq. 3.1 for $x^*$ using Alg. 5
            Update nodal positions of the free nodes on $e$ to $x^*$
        end if
    end for
    passes = passes + 1
end while
```

Algorithm 5 Pseudocode for our BFGS quasi-Newton method

```
Given an initial value $x_0$, an initial value for the Hessian $B_0$, and a tolerance $tol$;
while $\|\nabla f(x_k)\| > tol$ do
    Compute Cholesky factorization $B_k = LL^T$
    Compute the direction vector $d_k$ by solving $LL^T d_k = -\nabla f(x_k)$.
    $\rho_k = 1.0$
    while $f(x_k + \rho_k d_k) > f(x_k) + 10^{-4}\rho_k \nabla f(x_k)^T d_k$ do
        $\rho_k = 0.5\rho_k$
    end while
    $x_{k+1} = x_k + \rho_k d_k$
    $s_k = x_{k+1} - x_k$
    $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$
    $B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}$
    $k = k + 1$
end while
```

3.2.1 Measuring the Angles of Curvilinear Triangles

In order to compute the angle between two curves at a given point, we compute the derivatives of the curves, evaluate the derivatives at the given point, and then compute the angle between the resulting tangent vectors. Following this approach, we will compute the angles between each pair of edges of curvilinear triangles. For our derivation, we use the third-order Lagrange elements. Derivation for the second-order Lagrange elements is similar.

Consider the third-order Lagrange triangle shown in Fig. 3.3 with shape functions defined as follows:

$$\begin{aligned}
s_1 &= \frac{9}{2}(1-\xi-\eta)\left(\frac{1}{3}-\xi-\eta\right)\left(\frac{2}{3}-\xi-\eta\right) \\
s_2 &= \frac{9}{2}\xi\left(\xi-\frac{1}{3}\right)\left(\xi-\frac{2}{3}\right) \\
s_3 &= \frac{9}{2}\eta\left(\eta-\frac{1}{3}\right)\left(\eta-\frac{2}{3}\right) \\
s_4 &= \frac{27}{2}(1-\xi-\eta)\xi\left(\frac{2}{3}-\xi-\eta\right) \\
s_5 &= \frac{27}{2}(1-\xi-\eta)\xi\left(\xi-\frac{1}{3}\right) \\
s_6 &= \frac{27}{2}\xi\eta\left(\xi-\frac{1}{3}\right) \\
s_7 &= \frac{27}{2}\xi\eta\left(\eta-\frac{1}{3}\right) \\
s_8 &= \frac{27}{2}(1-\xi-\eta)\eta\left(\eta-\frac{1}{3}\right) \\
s_9 &= \frac{27}{2}(1-\xi-\eta)\eta\left(\frac{2}{3}-\xi-\eta\right) \\
s_{10} &= 27\xi\eta(1-\xi-\eta).
\end{aligned}$$

The mapping $\phi(\xi,\eta)$ from the reference unit element in Fig. 3.3 onto the physical element is then given by:

$$\phi(\xi,\eta)=\sum_{i=1}^{10}\mathbf{x_i}\, s_i(\xi,\eta), \tag{3.2}$$

where $\mathbf{x_i}$ are the nodal positions, and (ξ,η) is a point in the reference element. Since we are con-

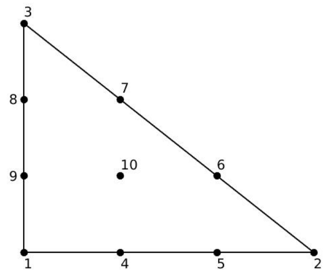


Figure 3.3: Third-order Lagrange reference unit triangle

cerned with the angles between each pair of edges, we need to define mappings from each point on the edges of the reference element to the corresponding point on the edges of the physical element. The edges correspond to third-order Lagrange elements in 1D. The shape functions associated with these elements are defined as:

$$\begin{aligned} n_1(t) &= \frac{9}{2}(1-t)\left(\frac{2}{3}-t\right)\left(\frac{1}{3}-t\right) \\ n_2(t) &= \frac{27}{2}(1-t)\left(\frac{2}{3}-t\right)(t) \\ n_3(t) &= \frac{27}{2}(1-t)\left(\frac{1}{3}-t\right)(-t) \\ n_4(t) &= \frac{9}{2}\left(\frac{2}{3}-t\right)\left(\frac{1}{3}-t\right)(t). \end{aligned}$$

The derivatives of these shape functions with respect to t are given by:

$$\begin{aligned} {n_1}'(t) &= \frac{1}{2}\left(-11+36t-27t^2\right) \\ {n_2}'(t) &= \frac{1}{2}\left(18-90t+81t^2\right) \\ {n_3}'(t) &= \frac{1}{2}\left(-9+72t-81e^2\right) \\ {n_4}'(t) &= \frac{1}{2}\left(2-18t+27t^2\right). \end{aligned}$$

Using these shape functions, we define the mappings from each edge in the reference element to each edge in the physical element as:

$$\begin{aligned}
\mathbf{f_{12}}(\mathbf{t}) &= \mathbf{x_1}n_1(t)+\mathbf{x_4}n_2(t)+\mathbf{x_5}n_3(t)+\mathbf{x_2}n_4(t) \\
\mathbf{f_{23}}(\mathbf{t}) &= \mathbf{x_2}n_1(t)+\mathbf{x_6}n_2(t)+\mathbf{x_7}n_3(t)+\mathbf{x_3}n_4(t) \\
\mathbf{f_{31}}(\mathbf{t}) &= \mathbf{x_3}n_1(t)+\mathbf{x_8}n_2(t)+\mathbf{x_9}n_3(t)+\mathbf{x_1}n_4(t).
\end{aligned}$$

The notation f_{ij} denotes the edge between nodes i and j in Fig. 3.3. In their expanded forms, each $\mathbf{f_{ij}}(\mathbf{t})$ is a cubic polynomial in the variable t. Next, we need to compute the derivatives of our functions. Straightforward differentiation with respect to t results in the following:

$$\begin{aligned}
\mathbf{f_{12}}'(\mathbf{t}) &= \mathbf{x_1}n_1'(t)+\mathbf{x_4}n_2'(t)+\mathbf{x_5}n_3'(t)+\mathbf{x_2}n_4'(t) \\
\mathbf{f_{23}}'(\mathbf{t}) &= \mathbf{x_2}n_1'(t)+\mathbf{x_6}n_2'(t)+\mathbf{x_7}n_3'(t)+\mathbf{x_3}n_4'(t) \\
\mathbf{f_{31}}'(\mathbf{t}) &= \mathbf{x_3}n_1'(t)+\mathbf{x_8}n_2'(t)+\mathbf{x_9}n_3'(t)+\mathbf{x_1}n_4'(t).
\end{aligned}$$

Given these derivatives, we can return to the problem of calculating the angles between edges. As an example, suppose that we want to calculate the angle between edge e_{12} and edge e_{31} in Fig. 3.3. To calculate the angle in radians, we use the following formula:

$$\theta = \pi - \arccos\left(\frac{f_{12}'(0)f_{31}'(1)}{||f_{12}'(0)||\ ||f_{31}'(1)||}\right) = \frac{\pi}{2}.$$

Returning to the calculation of $\alpha_{\mathbf{i}}(\mathbf{x})$ in Eq. 3.1, we loop over each triangle in the patch and calculate the two angles of each triangle formed by the edges incident to the shared edge between the triangles as described above.

3.3 Numerical Experiments

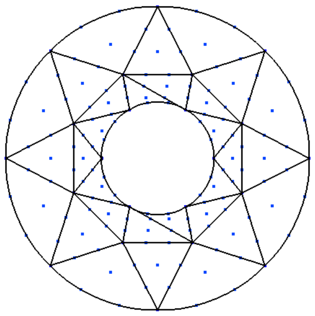
In this section, we show the results from performing several numerical experiments to untangle invalid second- and third-order meshes. For each example, we show the initial meshes; the meshes which result after untangling them with our method; the minimum distortion, maximum distortion, average distortion computed over all elements (referred to as Avg1 in figures), and average distortion computed over curved elements (referred to as Avg2 in figures), and the run time needed for our method to untangle the mesh. For each mesh, we show the nodes associated with the element of the given order. We do not show the location of the quadrature points. The code was run using Matlab R2017b, and the execution times were measured on a machine with 8GB of RAM and an Intel Xeon(R) W3520 CPU. All mesh visualizations and distortion calculations were done using Gmsh [10].

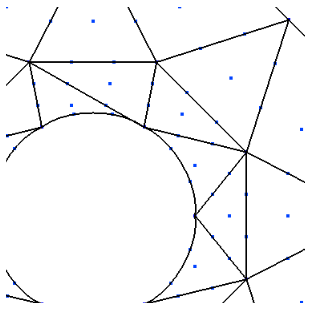
Our first example is a third-order annulus composed of 30 elements. During the process of curving the boundary, tangled elements were created near the top and bottom of the inner ring. Figure 3.4(a,c) show the initial invalid mesh and the final mesh resulting from our method, respectively. Figure 3.4(b,d), show detailed views of the inner ring from Fig. 3.4(a,c), respectively. In Fig. 3.4(e) we give the mesh element distortion.

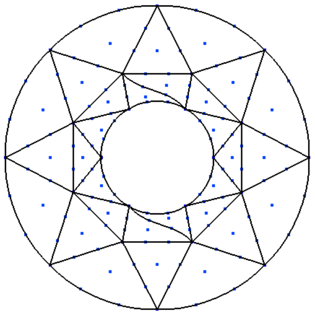
Our second example is the leading edge of a third-order NACA0012 airfoil. In Fig. 3.5(a), we can see that curving the inner boundary resulted in two tangled elements near the leading edge of the airfoil. In Fig. 3.5(b), we show the final mesh resulting from our method. Finally, Fig. 3.5(c) gives the mesh element distortion.

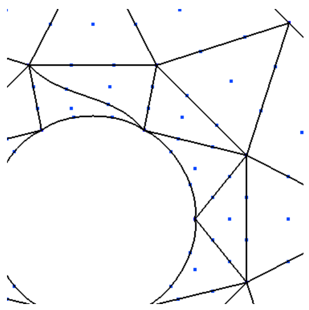
Our third example is a second-order mesh of a mechanical part with several holes. Figure 3.6(a-c) shows the initial invalid mesh, the final mesh resulting from our method, and the mesh quality as measured by the distortion metric. In Fig. 3.6(a), we can see that curving the boundaries resulted in tangled elements near the top and bottom holes.

Our fourth and fifth examples are valid meshes of a square plate with a circular hole. To induce mesh tangling in the fourth example, we applied a rotation of 10 degrees counterclockwise to the inner ring followed by a horizontal shear with a shear factor of 0.5. In Fig. 3.7(a,b,d), we show the initial valid mesh, the tangled mesh resulting from rotation and shearing, and the final untangled mesh resulting from our method. In Fig. 3.7(c,e), we show detailed views of the inner ring. In Fig. 3.7(f), we give the element distortion for the initial, tangled, and final meshes, respectively. In the


(a)


(b)

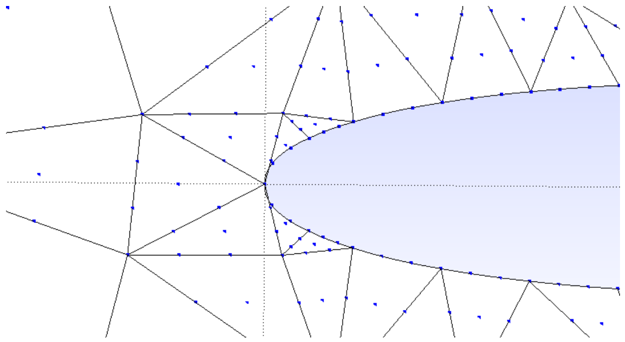

(c)


(d)

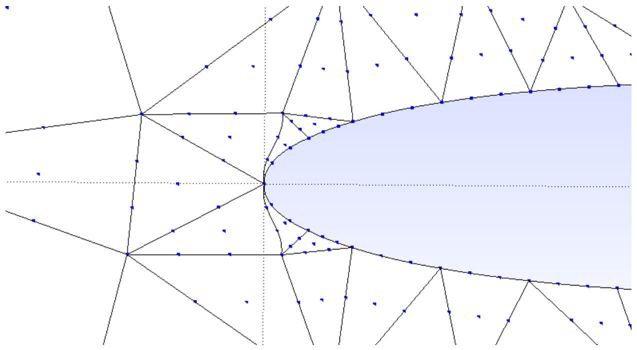
	Distortion			
Example	**Min**	**Max**	**Avg1**	**Avg2**
original mesh	-0.223	1.0000	0.615	0.279
resulting mesh	0.039	1.0000	0.595	0.325

(e)

Figure 3.4: Annulus example: (a) the tangled third-order mesh; (b) a detailed view of one tangled element along the top of the inner boundary; (c) the mesh resulting from our method; (d) a detailed view of the untangled element from (c), and (e) the mesh quality as measured by the element distortion metric.



(a)



(b)

	Distortion			
Example	**Min**	**Max**	**Avg1**	**Avg2**
original mesh	-0.566	1.000	0.968	0.967
resulting mesh	0.050	1.000	0.970	0.969

(c)

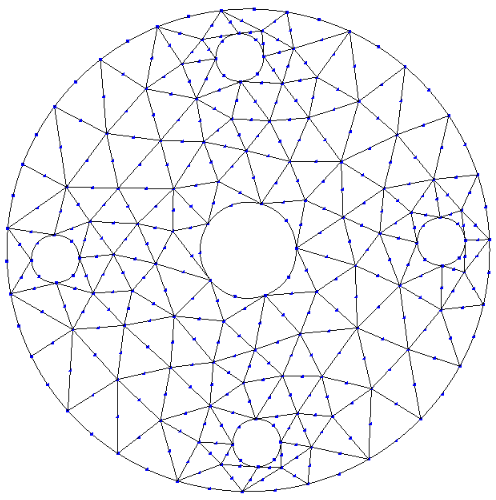
Figure 3.5: Airfoil example: (a) the tangled third-order mesh; (b) the mesh resulting from our method, and (c) the mesh quality as measured by the element distortion metric.

Example	Number of Elements	Wall Clock Time(s)
annulus	30	2.56
airfoil	282	16.63
mechanical part	182	2.44
plate 1	597	7.40
plate 2	597	7.48
beam	542	6.73

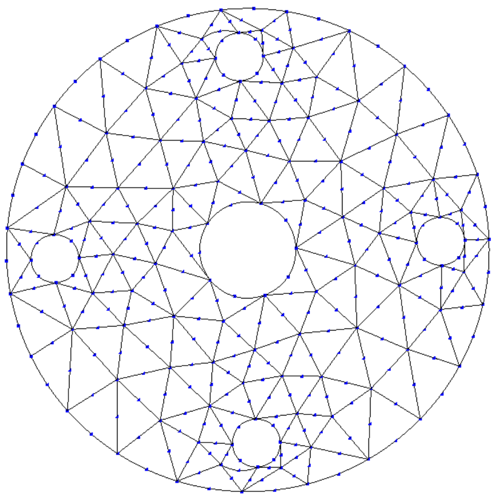
Table 3.1: The number of elements and the wall clock time for each mesh

fifth example, we applied a rotation of 10 degrees counterclockwise to the inner ring followed by a stretching of the bottom half of the plate. In Fig. 3.8(a,b,c), we show the initial valid mesh, the tangled mesh resulting from rotation and stretching, and the final untangled mesh resulting from our method. In Fig. 3.8(d), we give the element distortion for each mesh. In our final example, we show a valid mesh of a two-dimensional beam. To create mesh tangling, we treated the beam as a simply supported beam and applied a center load. After applying the load, we translated the left and right sides of the beam. In Fig. 3.9(a,b,c), we show the initial valid mesh, the tangled mesh resulting from our transformations, and the final untangled mesh resulting from our method. In Fig. 3.9(d,e), we show detailed views of the left side of the beam from Fig. 3.9(b,c). Finally in Fig. 3.9(f), we give the mesh element distortion.

While the test cases are relatively straightforward, our goal was to explore the types of tangling that occur as a result of moving the new boundary nodes onto the curved boundary during the typical high-order mesh generation process. We were also interested in tangling that might result from small deformations to a valid mesh. With these points in mind, the examples demonstrate that our method is able to handle the small deformations that might result in tangling for second- and third-order meshes. Additionally, our method only required a single pass for each of the test cases. We demonstrate the runtime performance of our method in Tab. 3.1. We list the number of elements and wall clock time for each of our numerical examples in Tab. 3.1. While these times are reasonable, for large meshes, faster run times will be required. Fortunately, there is high potential for improved performance using parallel computing, as our method can be applied to non-adjacent patches simultaneously.



(a)

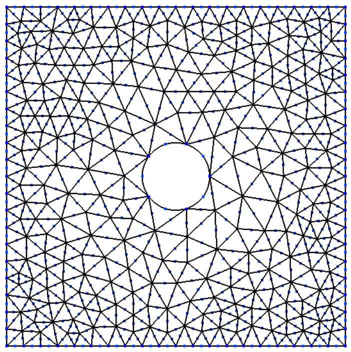


(b)

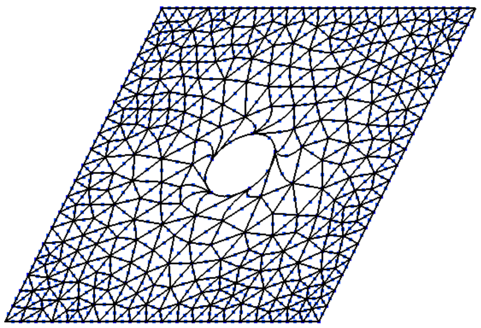
	Distortion			
Example	**Min**	**Max**	**Avg1**	**Avg2**
original mesh	-0.049	1.000	0.904	0.601
resulting mesh	0.008	1.000	0.905	0.625

(c)

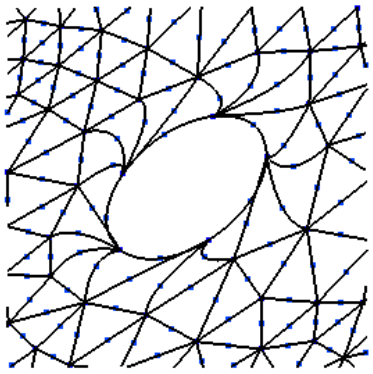
Figure 3.6: Mechanical part example: (a) the tangled second-order mesh; (b) the mesh resulting from our method, and (c) the mesh quality as measured by the element distortion metric.



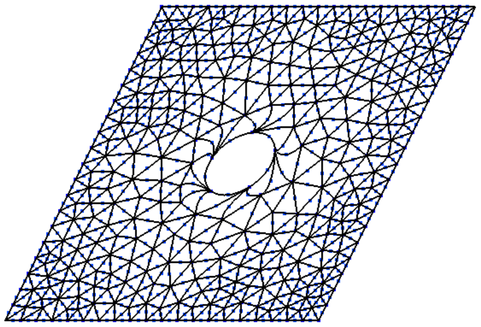
(a)



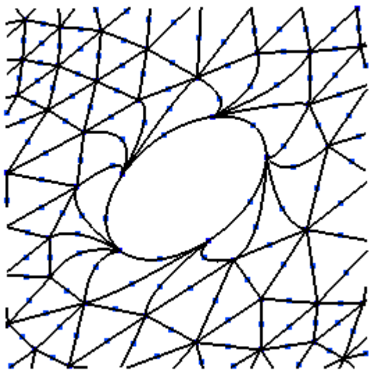
(b)



(c)



(d)

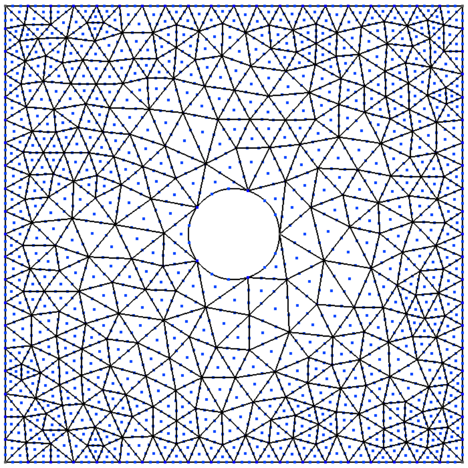


(e)

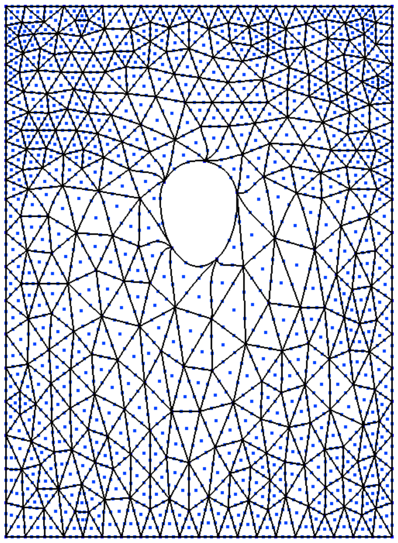
	Distortion			
Example	**Min**	**Max**	**Avg1**	**Avg2**
original mesh	0.287	1.000	0.995	0.412
tangled mesh	-0.716	1.000	0.974	0.375
resulting mesh	0.027	1.000	0.979	0.500

(f)

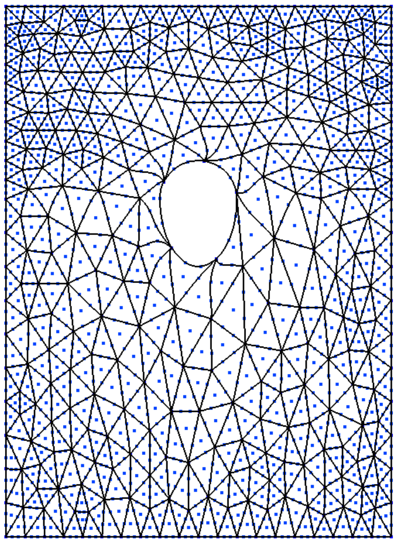
Figure 3.7: Square plate example: (a) the initial second-order mesh; (b) the mesh resulting from rotating the inner ring 10 degrees counterclockwise and applying a horizontal shear with a shear factor of 0.5; (c) a detailed view of the elements along the inner ring; (d) the mesh resulting from applying our method, (e) a detailed view of the elements along the inner ring; and (f) the mesh quality as measured by the element distortion metric.



(a)



(b)

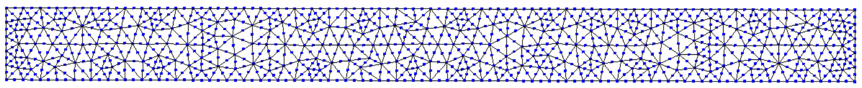


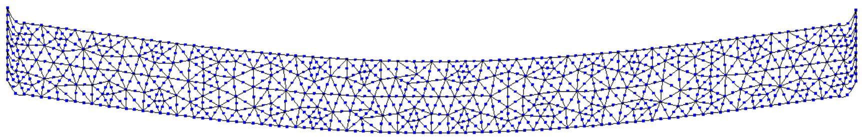
(c)

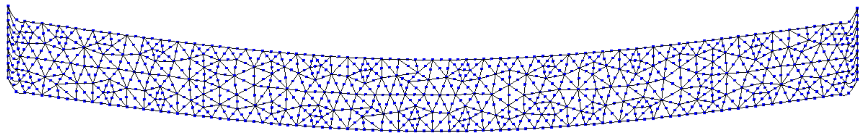
	Distortion			
Example	**Min**	**Max**	**Avg1**	**Avg2**
original mesh	0.287	1.000	0.995	0.412
tangled mesh	-0.120	1.000	0.966	0.934
resulting mesh	0.042	1.000	0.964	0.930

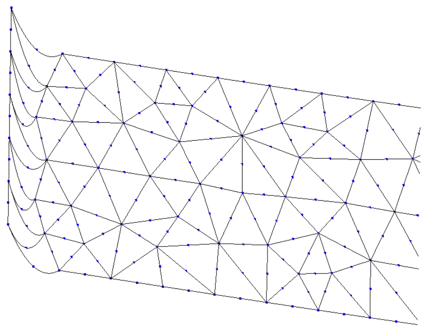
(d)

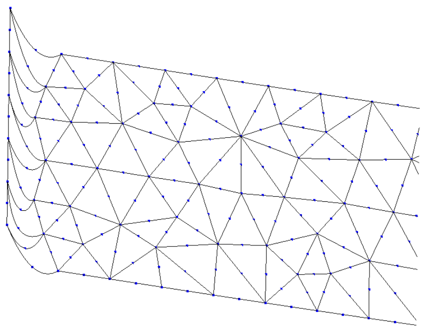
Figure 3.8: Square plate example: (a) the initial third-order mesh; (b) the mesh resulting from rotating the inner ring 10 degrees counterclockwise and stretching the bottom half of the plate; (c) the mesh resulting from applying our method, (d) the mesh quality as measured by the element distortion metric.


(a)


(b)


(c)


(d)


(e)

	Distortion			
Example	**Min**	**Max**	**Avg1**	**Avg2**
original mesh	1.000	1.000	1.000	1.000
tangled mesh	-0.044	1.000	0.982	0.975
resulting mesh	0.032	1.000	0.980	0.973

(f)

Figure 3.9: Beam example: (a) the initial second-order mesh; (b) the mesh after treating it as a simply supported beam with a center load and translating the left and right ends; (c) the mesh resulting from applying our method; (d) a detailed view of the left edge of (b); (e) a detailed view of the left edge of (c), and (f) the mesh quality as measured by the element distortion metric.

3.4 Conclusions

We have presented a new optimization-based method for untangling the edges of second- and third-order meshes. The two-dimensional examples have shown that our proposed method based on the unsigned angles of curvilinear triangles is able to successfully untangle several invalid second- and third-order meshes.

We note that presently our method has a few limitations. The first limitation is that it only allows movement of the high-order nodes that lie on the interior edge (e.g. the free nodes show in Fig. 3.1). That is, it does not allow movement of the endpoints. The second limitation is that non-edge nodes (e.g. like node 10 in Fig. 3.3) are not moved at all. The final limitation is that our objective function does not measure element validity. Due to these limitations, our method does not guarantee that it will successfully untangle a given tangled patch. With these limitations in mind, our future work will include extending the capabilities of our method to include moving non-edge nodes, as well as allowing the endpoints of edges to move. We will also explore the use of signed angles, where a negative angle indicates that tangling is present. This would allow us to directly check element validity, but would likely require modification of the objective function to achieve the desired untangling behavior. Other future improvements include extending our approach to three dimensions by using the dihedral angles between curved faces of high-order tetrahedral elements, and extending our implementation to allow for elements with $p > 3$.

3.5 Acknowledgments

The work of the first author was funded in part by the Madison and Lila Self Graduate Fellowship and NSF CCF grant 1717894. The work of the second author was funded in part by NSF CCF grant 1717894.

Chapter 4

Untangling High-order Meshes Based on Signed Angles[*]

Abstract

One challenge in the generation of high-order meshes is that mesh tangling can occur as a consequence of moving the new boundary nodes to the true curved boundary. In this paper, we propose a new optimization-based method that uses signed angles to untangle invalid second- and third-order triangular meshes. Our proposed method consists of two passes. In the first pass, we loop over each high-order interior edge node and minimize an objective function based on the signed angles of the pair of triangles that share the node. In the second pass, we loop over face nodes and move them to the mean of the high-order nodes of the triangle to which the face node belongs. We present several numerical examples in two dimensions with second- and third-order elements that demonstrate the capabilities of our method for untangling invalid meshes.

4.1 Introduction

One appealing aspect of high-order methods for solving partial differential equations is their ability to obtain more accurate solutions with a lower computational overhead than the corresponding low-order methods. One barrier to the adoption of these methods in the presence of curved domains is the lack of software capable of robustly generating high-order meshes [36]. In particular, to achieve the full potential of high-order methods in the presence of curved domains, these methods

[*]Stees, Mike, Dotzel, Myra, & Shontz, Suzanne M. (2020). Untangling High-Order Meshes Based on Signed Angles. Zenodo.

need to be paired with a high-order mesh that conforms to the curved domain [1, 17].

The typical approach used by high-order mesh generation methods is to apply a transformation to a coarse low-order mesh [3, 6, 7, 8, 9, 13, 18, 19, 22, 24, 25, 31, 33, 34, 35, 37]. The main difficulty in applying the transformation is obtaining a valid high-order mesh as the result. In general, these methods consist of the following three steps: (1) additional nodes are added to the low-order mesh; (2) the newly-added boundary nodes are projected onto the curved domain, and (3) the interior nodes are moved as a result of the boundary deformation. There are generally two approaches which are especially popular for transforming the low-order mesh. The first approach transforms the mesh based on optimization of an objective function [3, 4, 7, 8, 13, 24, 25, 26, 31, 33, 34]. Many of the proposed objective functions include a measure of element validity, which allows the methods to address invalid elements. While not all of the methods guarantee successful untangling, many of them are robust [3, 7, 8, 24, 25, 34]. The second approach transforms the mesh based on the solution of a partial differential equation [6, 18, 22, 37]. More specifically, Xie et al. [37] employed a linear elasticity approach, while Persson and Peraire [22] considered a nonlinear elasticity approach. Moxey et al. [18] used a thermoelastic model, and Fortunato and Persson [6] expressed the problem in terms of the Winslow equations.

In this paper, we describe a new two-pass method for untangling invalid second- and third-order triangular meshes. The first pass is an optimization-based approach that minimizes an objective function based on signed angles for each high-order interior edge node. The second pass is a smoothing step for the face nodes. The main focus of this work is to untangle invalid meshes that result from the boundary curving step of a typical high-order mesh generation method. Toward that end, we apply our method to several second- and third-order meshes that have invalid elements following the boundary curving process. The remainder of this paper is organized as follows. In Section 2, we present our new two-pass method for high-order mesh untangling. In Section 3, we demonstrate the performance of our method on several two dimensional examples. Finally, in Section 4, we offer concluding remarks and discuss some directions for our future work.

4.2 Untangling High-order Curvilinear Meshes

In this section, we propose a two-pass local node-based method for untangling high-order curvilinear triangular meshes. The first pass is based on the signed angles of curvilinear triangles, where a negative angle indicates tangling. For each iteration of the problem, we consider a high-order interior edge node. Then, we identify the two triangles that share the node and examine the four angles made by the tangent vectors adjacent to that edge. Our algorithm then moves the high-order edge node with the goal of making these angles positive. In our first pass, we solve the following unconstrained optimization problem:

$$\begin{aligned} f(x) &= (1-\beta)||\mathbf{x}-\mathbf{x_I}||_2+\beta\sum_{i=1}^{4}\mathbf{e}^{-\mathbf{10}*\alpha_{\mathbf{i}}(\mathbf{x})}, \\ x^* &= \underset{\mathbf{x}}{\operatorname{argmin}} f(x). \end{aligned} \tag{4.1}$$

where $\alpha_{\mathbf{i}}$ is the i^{th} entry of the vector of the four signed angles adjacent to a given interior edge; $\mathbf{x}$ is the nodal position of the high-order edge node to be moved; $\mathbf{x_I}$ is the initial position of the node at the start of the optimization, and β is a user-defined weighting parameter. By changing the value of β, more emphasis can be applied to the angles or the displacement of the node from its initial position. Note, if too much emphasis is placed on the displacement of the node, then the norm will dominate the objective function values, and the mesh will not be untangled.

To better understand the behavior of the objective function, consider the examples shown in Fig. 4.1 and the corresponding values shown in Fig. 4.2(a). The β value in this example was 0.35. In Fig. 4.1(a), we show the initial tangled mesh. At this point, the first term of $f(x)$ is zero because the interior node has not been moved. The second term will thus dominate the value of $f(x)$. In Fig. 4.1(b), we show the mesh after applying two iterations of the optimization method. As we see in the first two rows of Fig. 4.2(a), in both examples, the exponential term is the primary contributor to $f(x)$ because of α_3, the negative angle. In Fig. 4.1(c,d), we see that the impact of the exponential term decreases as the values of the angles increase (e.g., from negative to positive) after four and nine iterations, respectively. In other words, the second term in $f(x)$ acts as a penalty function to enforce positive angles (i.e., an untangled mesh). Once the angles become sufficiently positive, then the first term in $f(x)$ becomes a larger contributor to the overall value of $f(x)$. The goal of this term is to reduce the amount of displacement for a given node by minimizing the node's distance from its initial location.

To find a local minimum of our unconstrained optimization problem, we use a derivative-free method. We do so because of the complexity of evaluating $f(x)$, specifically, the signed angle calculations. In particular, to solve our optimization problem, we use the Nelder-Mead simplex method [15]. For the motivational example in Fig. 4.1, a relaxed convergence tolerance of 0.01 was used for the Nelder-Mead simplex method. For all of our examples in the next section, the tolerance and maximum number of iterations for Nelder-Mead were 0.0001 and 400, respectively. Convergence is reached when the change in function values and the step size both satisfy the tolerance.

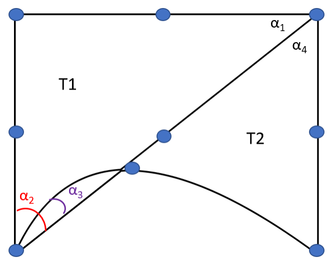
As described in [15], the Nelder-Mead simplex method is a direct search method that maintains a simplex at each step of the method. This simplex is defined by $n+1$ vertices and the corresponding function values, where n is the dimension of the problem space. Before moving forward, let us introduce the following notation for the description of the 2D method. Let the vertices of the current simplex be represented as v_1, v_2, and v_3. In addition, denote their corresponding function values $f(v_1)$, $f(v_2)$, and $f(v_3)$. Given these definitions, each iteration of a typical Nelder-Mead method consists of the following steps. First, the vertices are ordered from the lowest function value, say $f(v_1)$, to their highest function value, say $f(v_3)$. Second, the midpoint m of the best side of the simplex is computed, i.e., the side opposite v_3. Third, a new simplex is computed from the current one using reflection, expansion, or contraction steps. In Fig. 4.3, we show examples of the reflection, expansion, and contraction steps, denoted by r, e, and c_o/c_i, respectively. We also illustrate the current simplex with a solid black line and the simplices computed via the operations in dashed black lines. To compute the new simplex, an attempt is made to replace v_3 by reflecting the vertex about the best side. If the reflected vertex r leads to a decrease in the objective function, then an attempt is made at further reduction by computing an expansion vertex e. If $f(e) < f(r)$, then v_3 is replaced with e. Otherwise, v_3 is replaced with r. If the reflected vertex does not lead to a decrease in the objective function, then r is contracted back to c_o, and the function values are compared again. If this step fails to decrease the function, then c_o is reflected about the best edge to get c_i. If all of these steps are unsuccessful, then the simplex is shrunk toward vertex v_1, and a new simplex is formed with v_1, m, and the midpoint between v_1 and v_3

After minimizing the objective function for every high-order interior edge node (i.e., completing the first pass of our untangling algorithm), we perform a second pass to move the non-edge nodes. In this pass, for each non-edge node, we move the node to the mean of the high-order nodes of the triangle to which it belongs. To better motivate the need for two passes, we have included an example in Fig. 4.4. In Fig. 4.4(a), we show the initial tangled mesh. In Fig. 4.4(b), we show the mesh after completing the first pass of our method. Since the objective function applied in the first pass is formulated in terms of angles between edges, which do not apply to face nodes, the first pass neglects to improve the quality of these elements with respect to their face nodes. Thus, a face node's close proximity to the edge of its element could result in an invalid element. To address this kind of situation, we have included the second pass as shown in Fig. 4.4(c). This pass moves the face nodes toward the interior of the elements to which they belong. These two passes are performed until a tolerance is satisfied. In Alg. 6, we provide a pseudocode description of our untangling method. In the next section, we discuss how the signed angles $\alpha_i(x)$ of the curved elements are calculated.

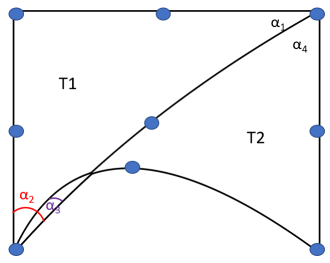
4.2.1 Computing the Signed Angles of Curvilinear Triangles

To compute the angle between two curves at a given node, we compute the derivatives of the curves, evaluate the derivatives at the given node, and then compute the angle between the resulting tangent vectors. Using this approach, we compute the angles between each pair of edges of curvilinear triangles. For the following derivation, we use the third-order Lagrange element. The derivation for other orders is similar.

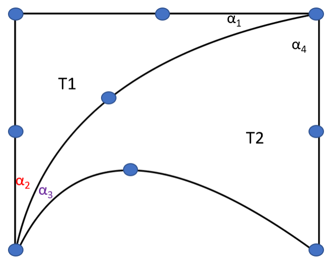
Consider the third-order Lagrange triangle shown in Fig. 4.5. To compute the angles between



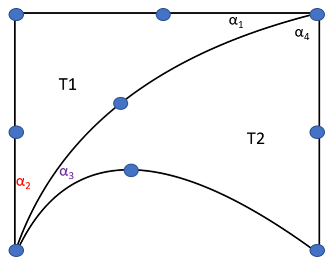
(a)



(b)



(c)

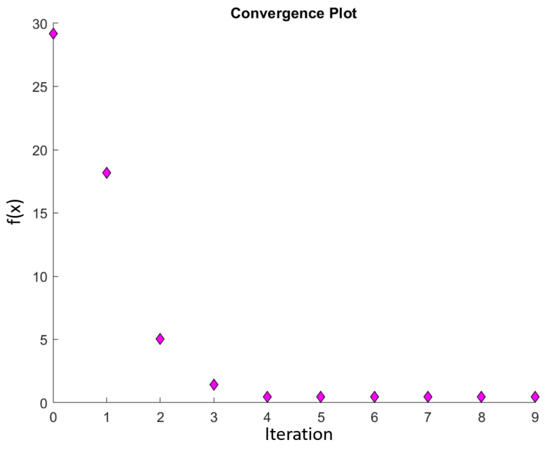


(d)

Figure 4.1: Motivating example: (a) the initial tangled mesh, (b) the mesh resulting from our method after two iterations, (c) the mesh resulting from our method after four iterations, and (d) the final mesh resulting from our converged method.

Iteration Number	Term 1	Term 2	$f(x)$
0	0.0000	29.1931	29.1931
2	0.0812	4.9445	5.0257
4	0.3096	0.1620	0.4716
9	0.2538	0.1963	0.4501

(a)



(b)

Figure 4.2: Figure showing (a) the contributions of each term in $f(x)$ during different iterations of the optimization method, and (b) a convergence plot of our method applied to the example in Fig. 4.1.

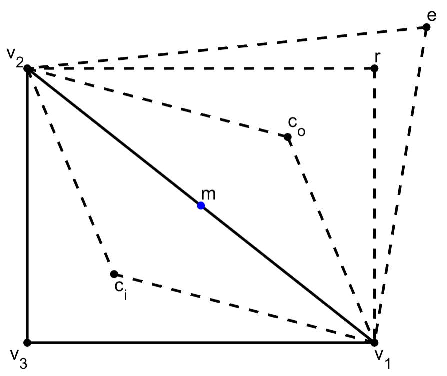
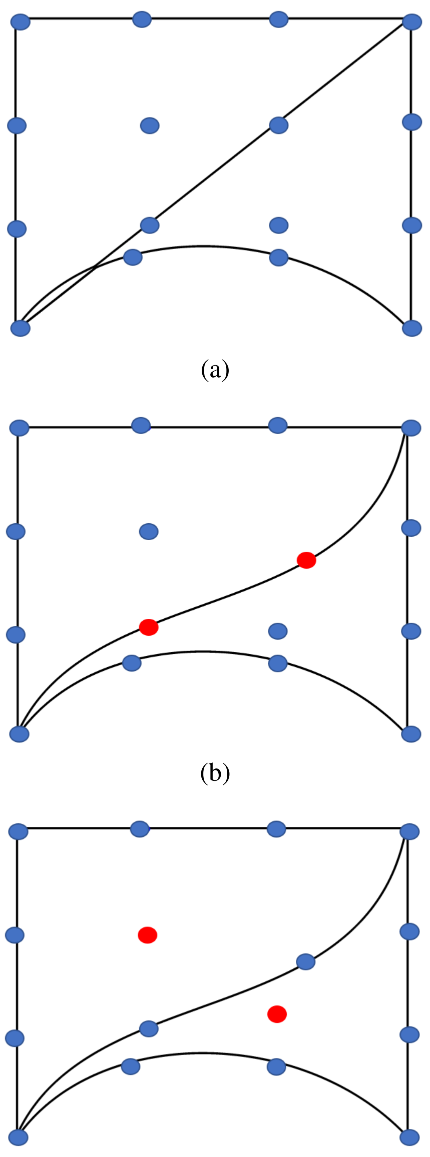


Figure 4.3: The current simplex marked by a solid line, and the simplices computed using the reflection, expansion, and contraction inside/outside operations during a single iteration of a typical Nelder-Mead method.



(a)

(b)

(c)

Figure 4.4: Motivating example for two pass approach: (a) the initial tangled mesh, (b) the mesh after completing the first pass with impacted nodes shown in red, and (c) the mesh after completing the second pass with influenced nodes shown in red.

Algorithm 6 Pseudocode for our node-based mesh untangling method

```
$X^0$ = the zero matrix
$X^1$ = the matrix of node positions at iteration 1
while $\frac{||X^k - X^{k-1}||_F}{||X^k||_F} > 10^{-4}$ do
    First Pass:
    for each high-order interior edge node $i$ do
        1. Find the two triangles $t_1$ and $t_2$ which share node $i$
        2. Solve (4.1) for $x^*$ using Nelder-Mead simplex method [15]
        3. Update nodal position $i$ to $x^*$
    end for
    Second Pass:
    for each high-order interior face node $i$ do
        1. Find the triangle $t_1$ which contains node $i$
        2. Update nodal position $i$ to the mean of $t_1$'s high-order edge nodes
    end for
    $X^{k+1}$ = the matrix with updated node positions
end while
```

each pair of edges, we need to define mappings from each node on the edges of the reference element to the corresponding node on the edges of the physical element. Each edge corresponds to a third-order Lagrange element in one dimension. The shape functions associated with these elements are defined as:

$$\begin{aligned}
n_1(t) &= \frac{9}{2}(1-t)\left(\frac{2}{3}-t\right)\left(\frac{1}{3}-t\right), \\
n_2(t) &= \frac{27}{2}(1-t)\left(\frac{2}{3}-t\right)(t), \\
n_3(t) &= \frac{27}{2}(1-t)\left(\frac{1}{3}-t\right)(-t), \\
n_4(t) &= \frac{9}{2}\left(\frac{2}{3}-t\right)\left(\frac{1}{3}-t\right)(t).
\end{aligned}$$

The derivatives of these shape functions with respect to t are given by:

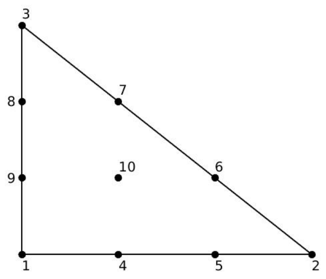


Figure 4.5: Third-order Lagrange reference unit triangle

$$\begin{aligned}
{n_1}'(t) &= \frac{1}{2}\left(-11+36t-27t^2\right), \\
{n_2}'(t) &= \frac{1}{2}\left(18-90t+81t^2\right), \\
{n_3}'(t) &= \frac{1}{2}\left(-9+72t-81t^2\right), \\
{n_4}'(t) &= \frac{1}{2}\left(2-18t+27t^2\right).
\end{aligned}$$

Using these shape functions, we can define the mappings from each edge in the reference element to each edge in the physical element as:

$$\begin{aligned}
\mathbf{f_{12}}(\mathbf{t}) &= \mathbf{x_1}n_1(t)+\mathbf{x_4}n_2(t)+\mathbf{x_5}n_3(t)+\mathbf{x_2}n_4(t), \\
\mathbf{f_{23}}(\mathbf{t}) &= \mathbf{x_2}n_1(t)+\mathbf{x_6}n_2(t)+\mathbf{x_7}n_3(t)+\mathbf{x_3}n_4(t), \\
\mathbf{f_{31}}(\mathbf{t}) &= \mathbf{x_3}n_1(t)+\mathbf{x_8}n_2(t)+\mathbf{x_9}n_3(t)+\mathbf{x_1}n_4(t).
\end{aligned}$$

The notation f_{ij} denotes the edge between nodes i and j in Fig. 4.5. Now that we have the mappings, we need to compute the derivatives of our functions. Taking the derivative with respect to t results in the following:

$$\begin{aligned}
\mathbf{f_{12}}'(\mathbf{t}) &= \mathbf{x_1}{n_1}'(t)+\mathbf{x_4}{n_2}'(t)+\mathbf{x_5}{n_3}'(t)+\mathbf{x_2}{n_4}'(t), \\
\mathbf{f_{23}}'(\mathbf{t}) &= \mathbf{x_2}{n_1}'(t)+\mathbf{x_6}{n_2}'(t)+\mathbf{x_7}{n_3}'(t)+\mathbf{x_3}{n_4}'(t), \\
\mathbf{f_{31}}'(\mathbf{t}) &= \mathbf{x_3}{n_1}'(t)+\mathbf{x_8}{n_2}'(t)+\mathbf{x_9}{n_3}'(t)+\mathbf{x_1}{n_4}'(t).
\end{aligned}$$

Given these derivatives, we can return to the problem of calculating the angles between edges. As an example, suppose that we want to calculate the angle between edge e_{12} and edge e_{31} in Fig. 4.5. To calculate the unsigned angle in radians, we could use the following formula:

$$\theta = \arccos\left(\frac{-f_{12}'(0)\cdot f_{31}'(1)}{||f_{12}'(0)||\;||-f_{31}'(1)||}\right) = \frac{\pi}{2}.$$

In order to calculate the signed angle in radians, we need to modify our calculations. First, we need to include an orientation unit vector $\mathbf{n}$. Then we need to modify our tangent vectors by adding a third component with a value of zero so that the cross product is defined, as well as normalize them. With these modifications, we can compute the signed angle using the following formula:

$$\text{signed angle} = \operatorname{sgn}(\mathbf{n}\cdot(v1\times v2))\cdot\arccos(v1\cdot v2)$$

where

$$\begin{aligned}\mathbf{v1} &= \frac{[f_{12}'(0),0]}{||\,[f_{12}'(0),0]\,||_2},\\ \mathbf{v2} &= \frac{[-f_{31}'(1),0]}{||[-f_{31}'(1),0]||_2},\\ \mathbf{n} &= [0,0,1].\end{aligned}$$

4.3 Numerical Results

In this section, we demonstrate the results from applying our method to untangle several high-order meshes. In each example, the nodes are processed in the order in which they occur in the original mesh. While we have explored other node orderings and found that the order does impact the number of outer iterations required for convergence, we note that this ordering does not influence the final resulting mesh. For each example, we provide a description of the mesh, the initial mesh (with tangled elements shown in red), the mesh which results from applying our untangling

					Runtime (s)	
Example	**Number of Elements**	**Mesh Order**	β	**Number of Iterations**	**First Pass**	**Second Pass**
annulus	30	3	0.500	2	0.005	0.000
mechanical part	295	2	0.500	2	0.041	—
bike gear	672	2	0.035	5	0.321	—
pressure plate	529	2	0.350	5	0.249	—
gear	1340	3	0.950	8	11.323	0.005
brake rotor	7015	2	0.850	2	19.643	—
airfoil	5328	3	0.001	2	24.039	0.005

Table 4.1: The number of elements, mesh order, beta value, number of outer iterations, and the wall clock times for each pass our untangling method (excluding I/O) for each example. Since the second-order meshes do not utilize the second pass, the columns are marked with a dash.

method, and the mesh element distortion as measured by the scaled Jacobian ([3]):

$$\text{scaled Jacobian} = \frac{\min J(\xi)}{\max J(\xi)},$$

where $J(\xi)$ is the Jacobian determinant. When reporting the mesh distortion, we list the minimum distortion and maximum distortion values. We also list the execution times for our untangling method (excluding I/O) in Table 4.1. The method was implemented in C++, and the wall-clock execution times were measured on a machine with 16GB of RAM and an AMD Ryzen 7 1700 CPU. All mesh visualizations and distortion evaluations were done using Gmsh [10, 12, 23].

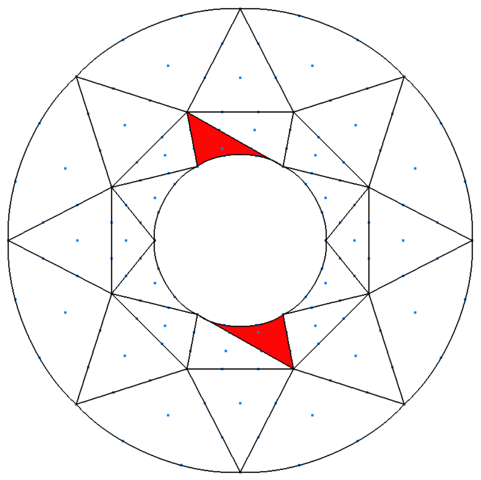
In our first example, we use a simple annulus geometry consisting of 30 elements to show the impact of different values of β on the result. In Fig. 4.6(a), we show the initial mesh with two tangled elements. In Fig. 4.6(b-d), we show the meshes resulting from β values of 0.1, 0.5, and 0.9, respectively. In Fig. 4.6(e), we show the min and max element distortions and execution times for each of the three values of β. As expected, higher values of β place more emphasis on the angular component of the objective function which tends to result in larger displacements of the edge nodes. Initially, increasing the value of β from 0.1 to 0.5 led to better elements with respect to distortion. Beyond 0.5, additional emphasis on the angles resulted in increased element distortion. For the remaining examples in this section, we report the value of β that resulted in the mesh with the least distortion. We also plot histograms of the mesh element distortion in addition to reporting the maximum and minimum values.

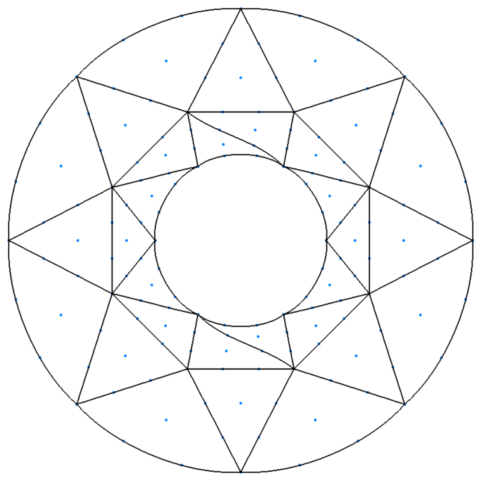
In the second example, we applied our method to a simple 2D mechanical part consisting of 295 second-order elements. Curving the boundaries resulted in two tangled elements near the innermost boundary. The initial tangled mesh and resulting untangled mesh are shown in Fig. 4.7(a,b). The minimum and maximum distortion values for these meshes are shown in Fig. 4.7(c). Finally, we plot histograms for these distortion values in Fig. 4.7(d,e). In this case, our solution raised the minimum distortion value of the mesh from -0.178 to 0.228.

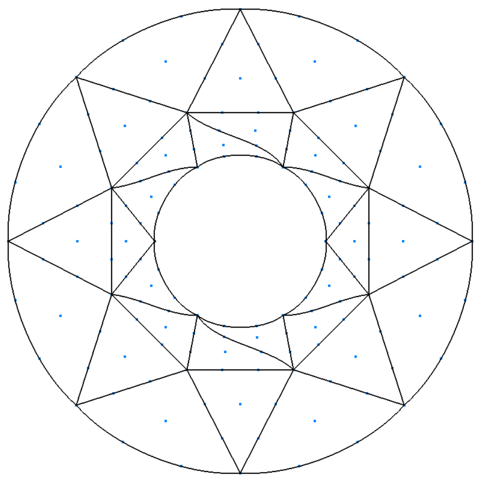
In our third example, we use a simplified bike gear with 672 second-order elements. In contrast with our previous examples, this mesh has several stretched elements near the boundaries which increase the potential for tangled elements after curving the boundaries. The initial tangled mesh and untangled mesh are shown in Fig. 4.8(a,b). Close-up views of the top third of the mesh are depicted in Fig. 4.8(c,d). The minimum and maximum distortion values for this mesh are recorded in Fig. 4.8(e). Lastly, histograms of the distortion values are plotted in Fig. 4.8(f,g). In this case, our method increased the minimum distortion value from -1.730 to 0.211, thus untangling the initial mesh.

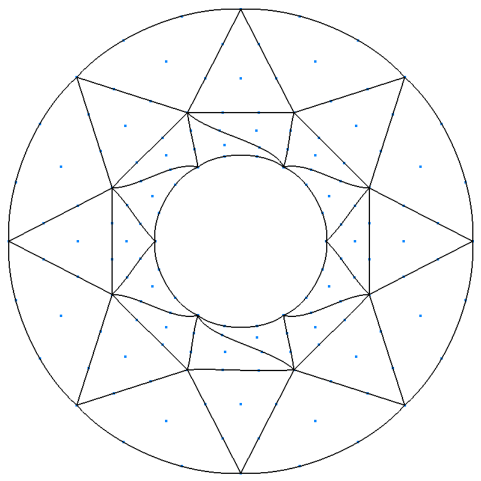
As our last simplified example, we use a pressure plate consisting of 529 second-order elements. After curving the boundaries, six tangled elements were created along the holes in the top and bottom of the geometry. Fig. 4.9(a,b) show the original tangled mesh and the untangled mesh resulting from our method. We show detailed views of the center of (a,b) in Fig. 4.9(c,d), respectively. In Fig. 4.9(e) we give the minimum and maximum mesh element distortion values. Finally in Fig. 4.9(f,g) we plot histograms of the distortion values. For this example, our method increased the minimum distortion value from -0.178 to 0.345.

For our remaining examples, we progress to more realistic meshes with a larger number of elements. The first example is a third-order gear composed of 1340 elements, eight of which are tangled. In Fig. 4.10(a,b) we show the initial tangled mesh and the final mesh produced by our untangling algorithm. In Fig. 4.10(c,d) we show detailed views of the center holes in (a,b),


(a)


(b)

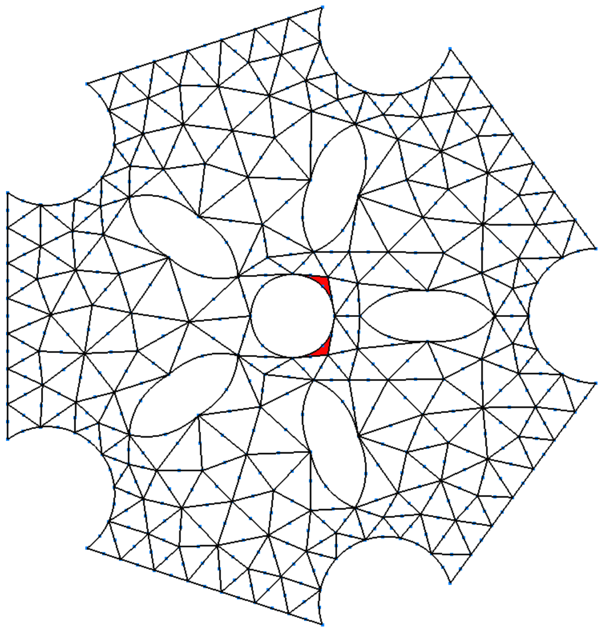

(c)

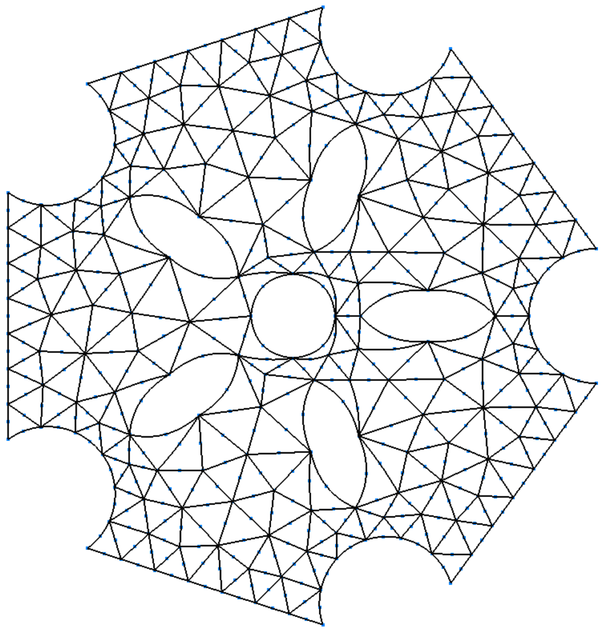

(d)

	Distortion		
Beta	**Min**	**Max**	**Runtime (s)**
0.1	0.208	1.000	0.004
0.5	0.472	1.000	0.005
0.9	0.348	1.000	0.014

(e)

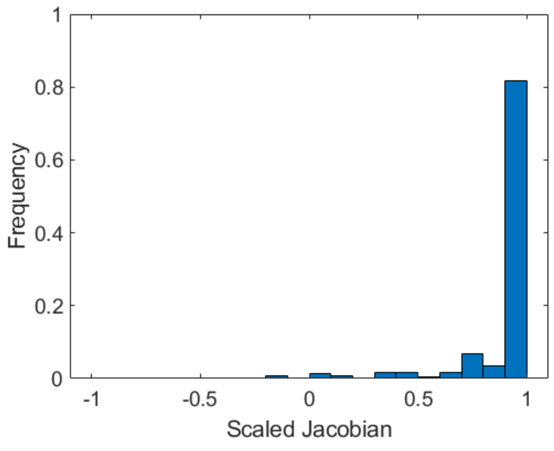
Figure 4.6: Annulus example with three different β values: (a) the initial mesh with two tangled elements; (b) to (d) untangled meshes for β values of 0.1, 0.5, and 0.9, respectively, and (e), shows the minimum and maximum element distortions and runtimes for each value of β.


(a)

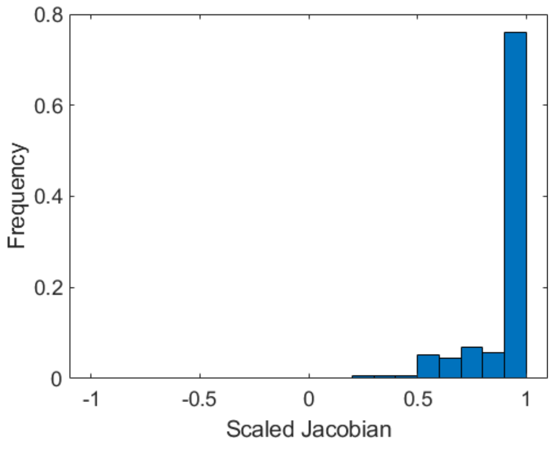

(b)

	Distortion	
Example	**Min**	**Max**
original mesh	-0.178	1.000
resulting mesh	0.228	1.000

(c)

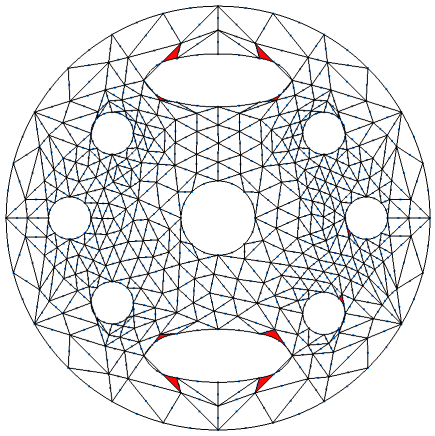


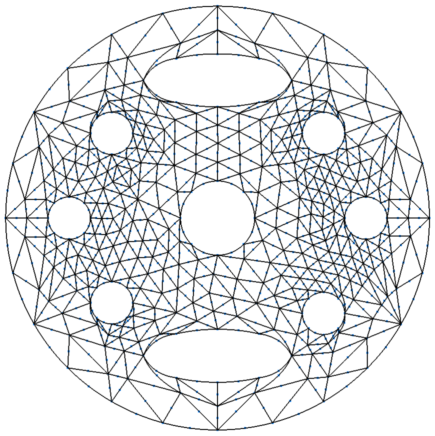
(d)

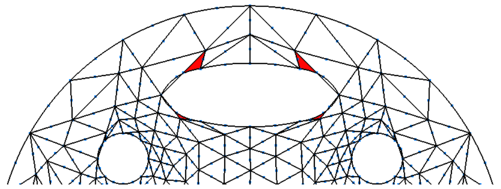


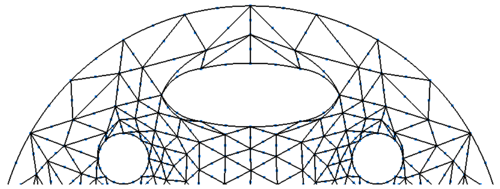
(e)

Figure 4.7: Mechanical part example: (a) the initial second-order mesh with two tangled elements; (b) the untangled mesh resulting from our method; (c) the minimum and maximum element distortion, and (d,e) histogram plots of the distortion metric for each mesh.


(a)

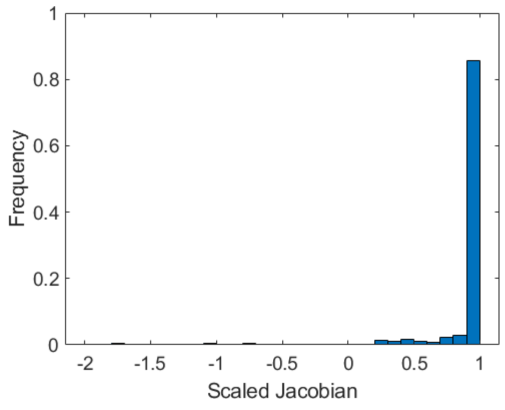

(b)


(c)

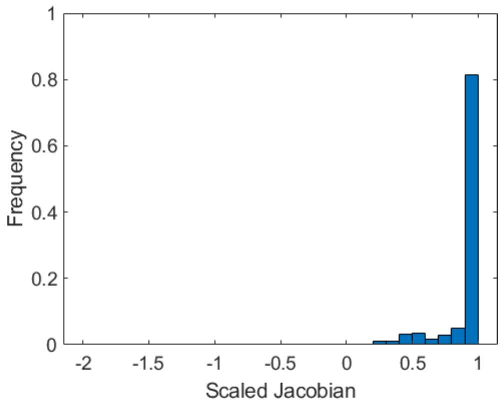

(d)

	Distortion	
Example	**Min**	**Max**
original mesh	-1.730	1.000
resulting mesh	0.211	1.000

(e)

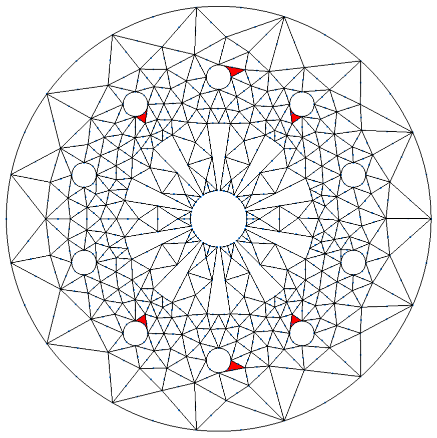


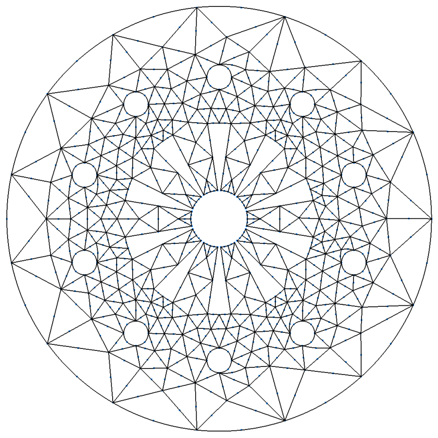
(f)

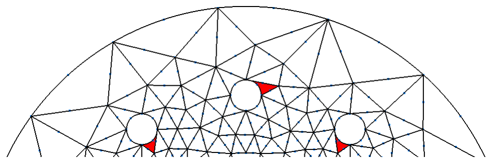


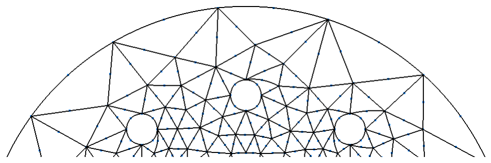
(g)

Figure 4.8: Bike gear example: (a) the tangled second-order mesh with fourteen tangled elements; (b) the mesh resulting from our method; (c,d) detailed views of (a,b), respectively; (e) the minimum and maximum element distortion, and (f,g) histogram plots of the distortion metric for (a,b), respectively.


(a)

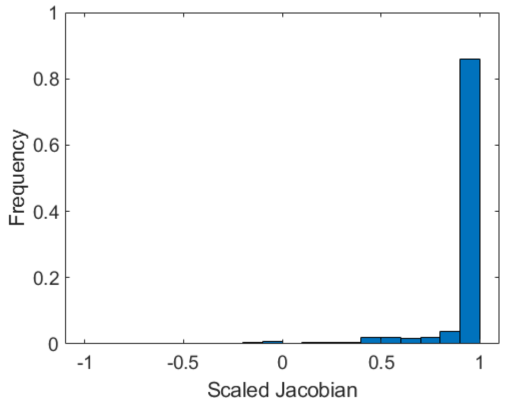

(b)


(c)

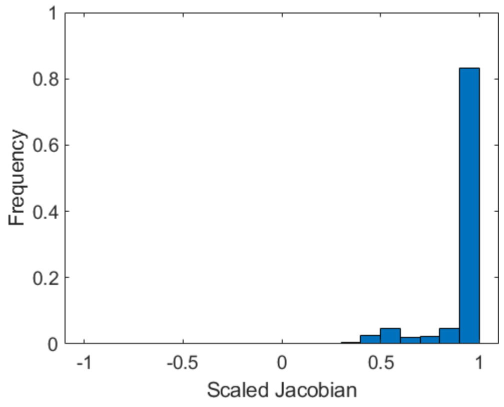

(d)

	Distortion	
Example	**Min**	**Max**
original mesh	-0.178	1.000
resulting mesh	0.345	1.000

(e)

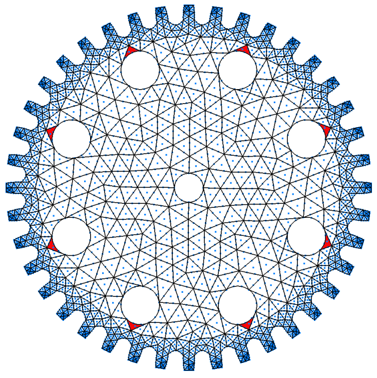


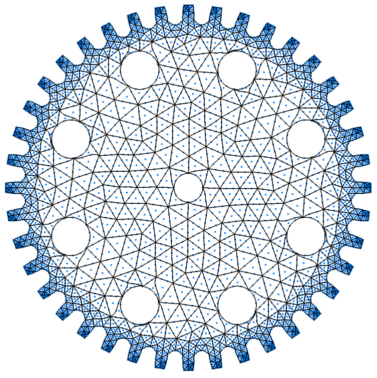
(f)

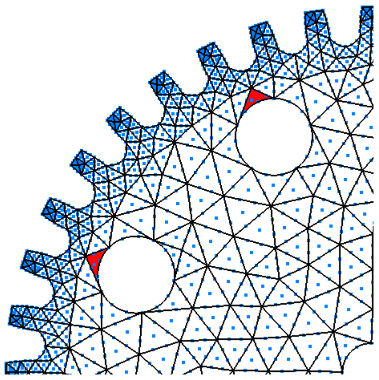


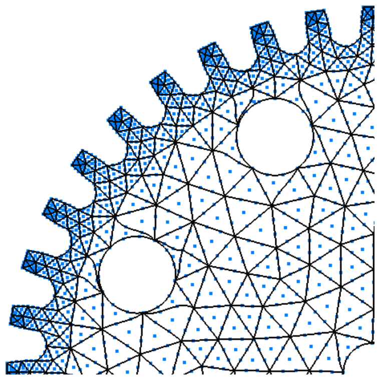
(g)

Figure 4.9: Pressure plate example: (a) the tangled second-order mesh; (b) the mesh resulting from our method; (c,d) detailed views of (a,b), respectively; (e) the minimum and maximum element distortion, and (f,g) histogram plots of the distortion metric for each mesh.


(a)

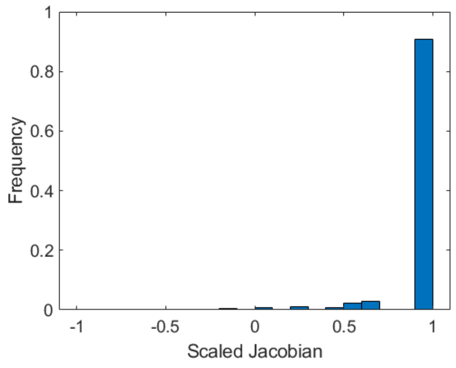

(b)


(c)

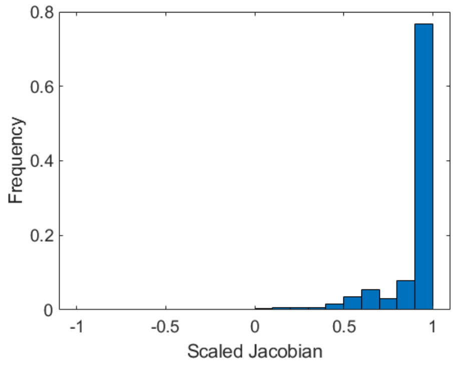

(d)

	Distortion	
Example	**Min**	**Max**
original mesh	-0.122	1.000
resulting mesh	0.092	1.000

(e)

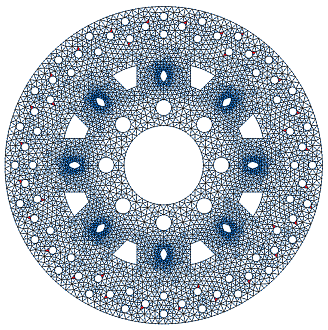


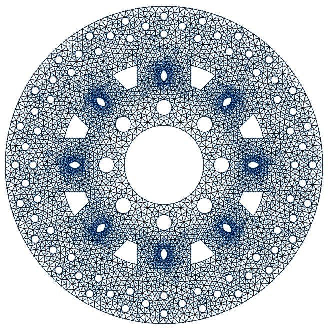
(f)

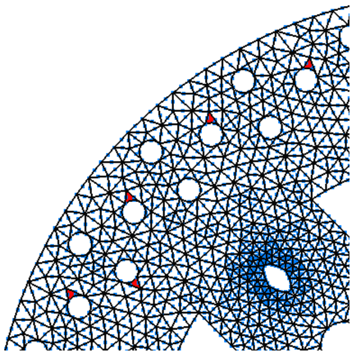


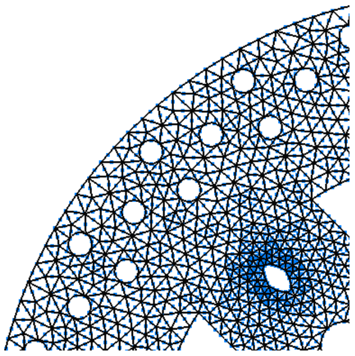
(g)

Figure 4.10: Gear example: (a) the initial third-order mesh with eight tangled elements; (b) the mesh resulting from our method; (c,d) detailed views of (a,b), respectively; (e) the minimum and maximum element distortion, and (f,g) histogram plots of the distortion metric for (a,b), respectively.


(a)

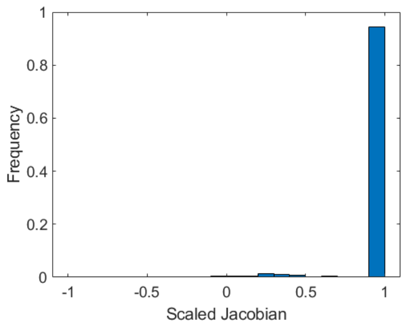

(b)


(c)

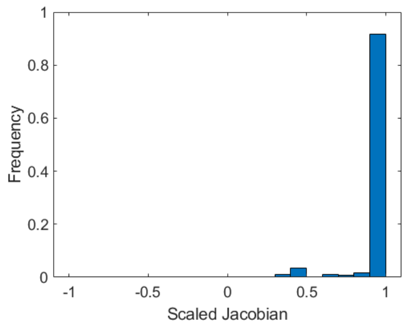

(d)

	Distortion	
Example	**Min**	**Max**
original mesh	-0.156	1.000
resulting mesh	0.346	1.000

(e)

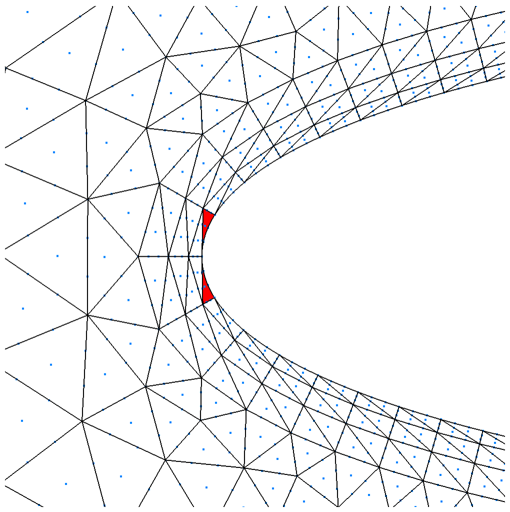


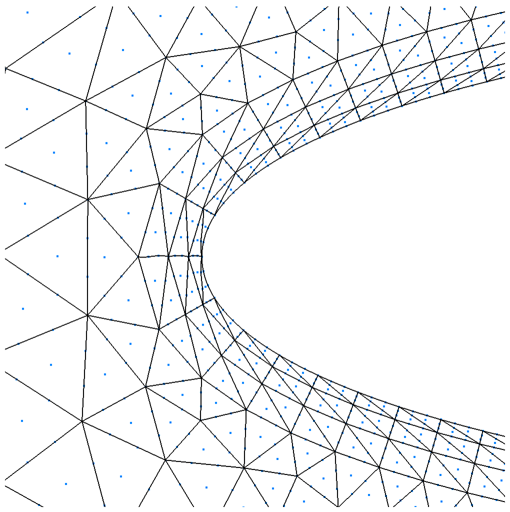
(f)



(g)

Figure 4.11: Brake rotor example: (a) the initial second-order mesh with thirty-four tangled elements; (b) the mesh resulting from our method; (c,d) detailed views of (a,b), respectively; (e) the minimum and maximum element distortion, and (f,g) histogram plots of the element distortion metric for (a,b), respectively.


(a)


(b)

	Distortion	
Example	**Min**	**Max**
original mesh	-0.109	1.000
resulting mesh	0.053	1.000

(c)

Figure 4.12: Airfoil example: (a) the initial third-order mesh with two tangled elements near the leading edge; (b) the mesh resulting from our method, and (c) the minimum and maximum element distortion.

respectively. The minimum and maximum mesh element distortion values are listed in Fig. 4.10(e). Finally in Fig. 4.10(f,g) we plot histograms of the distortion values. After applying our method, the minimum distortion value increased from -0.122 to 0.092.

Our next example is a brake rotor composted of 7015 second-order elements, thirty-four of which are tangled. In Fig. 4.11(a,b) we show the initial tangled mesh and the final mesh produced by our untangling algorithm. In Fig. 4.11(c,d) we show detailed views of the center holes in (a,b), respectively. The minimum and maximum mesh element distortion values are listed in Fig. 4.11(e). Finally in Fig. 4.11(f,g) we plot histograms of the distortion values. After applying our method, the minimum distortion value increased from -0.156 to 0.346.

Finally, our last example is an anisotropic boundary layer mesh of an airfoil. This example is a modified version of an example taken from the 2D benchmarks in the Gmsh repository. The mesh contains 5328 elements, with two tangled elements along the leading edge. In Fig. 4.12(a,b) we show the initial tangled mesh and the final mesh produced by our untangling algorithm. The minimum and maximum mesh element distortion values are listed in Fig. 4.12(c). The histogram plots for this example were omitted because there was minimal distinction between the two plots given the small percentage of tangled elements. After applying our method, the minimum distortion value increased from -0.109 to 0.053.

As we illustrated in Fig. 4.6, there are usually several choices for the parameter β that will result in an untangled mesh. As shown in Table 4.1, the trend we have observed thus far is that smaller values of β perform better for meshes with stretched elements near the curved features like our examples in Figs. 4.8 and 4.12. In particular, a smaller value of β was critical to maintaining the boundary layers in Fig. 4.12. Further experiments are necessary to determine what other factors influence the optimal value for β. The goal of these test cases was to explore the types of tangling that occur as a result of small deformations (e.g., moving the new boundary nodes onto the curved boundary during the typical high-order mesh generation process). With that in mind, our examples demonstrate that our method successfully addresses the typical types of tangling seen in this scenario. In addition to untangling the invalid patches, our method tends to reduce the amount of element distortion in all of our examples. In addition, there is potential to improve the performance of our method using parallel computing, as our local method can be applied to non-adjacent patches simultaneously.

4.4 Concluding Remarks and Future Work

We have presented a new optimization-based method for untangling second- and third-order triangular meshes. The two-dimensional examples have shown that our proposed method based on signed angles is able to successfully untangle a variety of invalid second- and third-order meshes. Furthermore, our method tends to dramatically decrease the amount of element distortion present in the mesh. As our final example in Section 3 showed, the addition of the weighting parameter gives the user increased flexibility in defining the behavior of the objective function. One limitation of our method is that it does not move the low-order nodes. To address this, we plan to combine our untangling algorithm with a weight-based scheme like the one proposed in [31]. By combining these two approaches, we could use the weight-based scheme to move the low-order nodes, and the method we have proposed in this paper to move the high-order nodes.

Our future work will include exploring techniques for determining the ideal value of the weighting parameter β. We will also extend our implementation to untangle meshes composed of elements with $p > 3$. In addition, we plan to extend the capabilities of our method to three dimensions by using signed solid angles between curved faces of high-order tetrahedral elements. We also plan to add support for additional element types (e.g., quadrilaterals, etc). Finally, we plan to explore examples with larger deformations that result in more complicated mesh tangling.

4.5 Acknowledgments

The work of the first author was funded by NSF CCF grant 1717894. The work of the second author was funded by an REU supplement to NSF CCF grant 1717894. The work of the third author was supported in part by NSF grants CCF 1717894 and OAC 1808553.

Chapter 5

Conclusion

The interest in high-order computational methods for PDEs is increasing, and, as a result, the need for high-order mesh generation methods is growing. While there are several existing methods for generating high-order meshes, many of them do not guarantee that the generated mesh will be valid. The overall aim of this work was to develop methods for robustly generating high-order meshes. Our initial work followed the approach of many other researchers in this area. That is, we considered high-order mesh generation from a mesh warping perspective. In contrast with other high-order mesh warping methods that are based on solutions to a PDE or minimizing a measure of distortion, we explored optimal weighted combinations of nodal positions. As our results in Chapter 2 indicate, for smaller deformations, the two methods within our high-order optimization-based mesh warping framework result in meshes with similar distortion. For small deformations on coarse meshes, the method based on optimal affine combinations tends to be faster with respect to runtime. On denser meshes with larger deformations, the two methods within our framework tend to have similar runtimes, but the optimal convex combinations result in meshes with fewer distorted elements. Overall, our resulting high-order warping framework based on optimal weighted combinations of nodal positions tends to work well for smaller mesh deformations, like those found in the typical high-order generation context. One limitation of the methods within our framework is that they do not incorporate a measure of element validity. As the final example in Chapter 2 indicates, this might result in a tangled mesh. Although the tangling only occurred for our method based on affine combinations, it is possible that the other method within our framework could result in a tangled mesh, as well. With this in mind, one goal for future work would be the inclusion of element validity in the objective function with the goal of generating weights that incorporate validity.

Our results from Chapter 2 informed the rest of the work contained here. In particular, our goal was to develop high-order mesh untangling methods which could be applied to our high-order meshes in the event that the mesh warping resulted in invalid elements. In Chapter 3 we presented our unconstrained optimization problem formulation for mesh untangling based on the unsigned angles of curvilinear triangles. As our results in Chapter 3 demonstrate, our edge-based method was successfully able to fix the types of tangling found in the typical high-order mesh generation process. While the method performed relatively well, it had several limitations. In particular, it only moved the high-order nodes on a given edge (i.e., the endpoints of the edge are fixed). Furthermore, non-edge nodes that are present in elements beyond second-order are not moved. To address these issues, we developed the signed angle formulation presented in Chapter 4.

The focus of Chapter 4 was on an untangling method based on signed angles. This method improved on our previous untangling method in several ways. First, the signed angle formulation included an approximation of element validity where negative angles indicated certain types of tangling. Second, this method included two passes, the first one that processed edge nodes, and the second one that processed face nodes. This approach allowed us to untangle certain invalid patches that we were previously unable to successfully untangle. In addition, the objective function in this method included a weighting parameter that allowed the user to place more emphasis on the angle or the displacement of the node. This addition allowed us to untangle boundary layer meshes while preserving the layers, something our unsigned untangling method was unable to do. Finally, this method tends to reduce the amount of element distortion present in the mesh, thus allowing the improvement of elements that were valid, but highly distorted. While both of our untangling methods presented a unique approach to high-order mesh untangling, they are limited to two-dimensional meshes. We explored several potential extensions of our untangling methods to 3D in the form of dihedral angles, solid angles, or some combination of both, but the resulting methods are unable to untangle high-order tetrahedra. While our angular approach did not extend to 3D, future work on related ideas might prove successful. For example, since the angle between two curves or two surfaces at a point is related to the distance between them (i.e., larger angles correspond to a greater distance between the two curves in the neighborhood of the point), it might be worthwhile to calculate the distance directly. One possible approach in 2D could be to sample each of the edges of the triangle and try to enforce that the distance between any two edges was strictly increasing as you move away from the the shared endpoint (i.e., the vertex where those two edges meet). In 3D, the two faces that share an edge could be sampled with a goal of increasing the distance between sample points as you move away from the shared edge.

Bibliography

[1] Bassi, F. & Rebay, S. (1997). High-order accurate discontinuous finite element solution of the 2D euler equations. *Journal of Computational Physics*, 138(2), 251 – 285.

[2] Boyd, S. & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

[3] Dey, S., O'bara, R. M., & Shephard, M. S. (1999). Curvilinear mesh generation in 3D. In *Proceedings of the 8th International Meshing Roundtable*.

[4] Dobrev, V., Knupp, P., Kolev, T., Mittal, K., & Tomov, V. (2019). The target-matrix optimization paradigm for high-order meshes. *SIAM Journal on Scientific Computing*, 41(1), B50–B68.

[5] Dobrev, V., Knupp, P., Kolev, T., & Tomov, V. (2018). Towards simulation-driven optimization of high-order meshes by the target-matrix optimization paradigm. In *Proceedings of the 27th International Meshing Roundtable* (pp. 285–302).: Springer.

[6] Fortunato, M. & Persson, P.-O. (2016). High-order unstructured curved mesh generation using the Winslow equations. *Journal of Computational Physics*, 307(2016), 1–14.

[7] Gargallo-Peiró, A., Roca, X., Peraire, J., & Sarrate, J. (2015a). Distortion and quality measures for validating and generating high-order tetrahedral meshes. *Engineering with Computers*, 31(3), 423–437.

[8] Gargallo-Peiró, A., Roca, X., Peraire, J., & Sarrate, J. (2015b). Optimization of a regularized distortion measure to generate curved high-order unstructured tetrahedral meshes. *International Journal for Numerical Methods in Engineering*, 103(5), 342–363.

[9] George, P. L. & Borouchaki, H. (2012). Construction of tetrahedral meshes of degree two. *International Journal for Numerical Methods in Engineering*, 90(9), 1156–1182.

[10] Geuzaine, C. & Remacle, J.-F. (2009). Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11), 1309–1331.

[11] Guennebaud, G. & Jacob, B. (2010). Eigen. *URL: http://eigen. tuxfamily. org*.

[12] Johnen, A., Remacle, J.-F., & Geuzaine, C. (2014). Geometrical validity of high-order triangular finite elements. *Engineering with Computers*, 30(3), 375–382.

[13] Karman, S. L., Erwin, J. T., Glasby, R. S., & Stefanski, D. (2016). High-order mesh curving using WCN mesh optimization. In *Proceedings of the 46th AIAA Fluid Dynamics Conference* (pp. 3178).

[14] Knupp, P. (2008). Updating meshes on deforming domains: An application of the target-matrix paradigm. *Communications in Numerical Methods in Engineering*, 24(6), 467–476.

[15] Lagarias, J. C., Reeds, J. A., Wright, M. H., & Wright, P. E. (1998). Convergence properties of the nelder–mead simplex method in low dimensions. *SIAM Journal on Optimization*, 9(1), 112–147.

[16] Li, J. & Chen, Y.-T. (2008). *Computational partial differential equations using MATLAB*. Crc Press.

[17] Luo, X., Shephard, M. S., & Remacle, J.-F. (2001). The influence of geometric approximation on the accuracy of high order methods. *Rensselaer SCOREC report*, 1.

[18] Moxey, D., Ekelschot, D., Keskin, Ü., Sherwin, S. J., & Peiró, J. (2016). High-order curvilinear meshing using a thermo-elastic analogy. *Computer-Aided Design*, 72, 130–139.

[19] Moxey, D., Green, M., Sherwin, S., & Peiró, J. (2015). An isoparametric approach to high-order curvilinear boundary-layer meshing. *Computer Methods in Applied Mechanics and Engineering*, 283, 636 – 650.

[20] Nocedal, J. & Wright, S. (2006). *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2nd edition.

[21] Panitanarak, T. & Shontz, S. M. (2017). A parallel log barrier-based mesh warping algorithm for distributed memory machines. *Engineering with Computers*, 34, 59–76.

[22] Persson, P.-O. & Peraire, J. (2009). Curved mesh generation and mesh refinement using Lagrangian solid mechanics. In *Proceedings of the 47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition* (pp. 949).

[23] Remacle, J.-F., Chevaugeon, N., Marchandise, E., & Geuzaine, C. (2007). Efficient visualization of high-order finite elements. *International Journal for Numerical Methods in Engineering*, 69(4), 750–771.

[24] Roca, X., Gargallo-Peiró, A., & Sarrate, J. (2012). Defining quality measures for high-order planar triangles and curved mesh generation. In *Proceedings of the 20th International Meshing Roundtable* (pp. 365–383).: Springer Berlin Heidelberg.

[25] Ruiz-Gironés, E., Sarrate, J., & Roca, X. (2016). Generation of curved high-order meshes with optimal quality and geometric accuracy. In *Proceedings of the 25th International Meshing Roundtable*, volume 163 (pp. 315–327). Procedia Engineering.

[26] Sherwin, S. J. & Peiró, J. (2001). Mesh generation in curvilinear domains using high-order elements. *International Journal for Numerical Methods in Engineering*, 53(1), 207–223.

[27] Shewchuk, J. (2002). What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures (preprint).

[28] Shontz, S. M. (2005). *Numerical methods for problems with moving meshes*. PhD thesis, Cornell University.

[29] Shontz, S. M. & Vavasis, S. A. (2003). A mesh warping algorithm based on weighted Laplacian smoothing. In *Proceedings of the 12th International Meshing Roundtable* (pp. 147–158).

[30] Stees, M., Dotzel, M., & Shontz, S. M. (2020). Untangling high-order meshes based on signed angles. In *Proceedings of the 28th International Meshing Roundtable*: Zenodo.

[31] Stees, M. & Shontz, S. M. (2017). A high-order log barrier-based mesh generation and warping method. In *Procedings of the 26th International Meshing Roundtable*, volume 203 (pp. 180 – 192).: Procedia Engineering.

[32] Stees, M. & Shontz, S. M. (2018). High-order mesh generation based on optimal affine combinations of nodal positions. In *To Appear in Proceedings of the International Conference on Spectral and High Order Methods*.

[33] Stees, M. & Shontz, S. M. (2019). An angular approach to untangling high-order curvilinear triangular meshes. In *Proceedings of the 27th International Meshing Roundtable* (pp. 327–342).: Springer.

[34] Toulorge, T., Geuzaine, C., Remacle, J.-F., & Lambrechts, J. (2013). Robust untangling of curvilinear meshes. *Journal of Computational Physics*, 254, 8 – 26.

[35] Turner, M., Moxey, D., Peiró, J., Gammon, M., Pollard, C. R., & Bucklow, H. (2017). A framework for the generation of high-order curvilinear hybrid meshes for CFD simulations. In *Proceedings of the 26th International Meshing Roundtable*, volume 203 (pp. 206 – 218).: Procedia Engineering.

[36] Wang, Z. J., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H. T., et al. (2013). High-order CFD methods: Current status and perspective. *International Journal for Numerical Methods in Fluids*, 72(8), 811–845.

[37] Xie, Z. Q., Sevilla, R., Hassan, O., & Morgan, K. (2013). The generation of arbitrary order curved meshes for 3D finite element analysis. *Computational Mechanics*, 51(3), 361–374.

Appendix A

Accepted Papers

- **Stees, M**, Dotzel, M, Shontz, SM (2019) Untangling High-order Meshes Based on Signed Angles. Proceedings of the 28th International Meshing Roundtable.
- **Stees, M**, Shontz, SM (2018) An Angular Approach to Untangling High-Order Curvilinear Triangular Meshes. Proceedings of the 27th International Meshing Roundtable.
- **Stees, M**, Shontz, SM (2018) High-Order Mesh Generation Based on Optimal Affine Combinations of Nodal Positions. Proceedings of the International Conference on Spectral and High Order Methods (ICOSAHOM).
- **Stees, M**, Shontz, SM (2017) A high-order log barrier-based mesh generation and warping method. Proceedings of the 26th International Meshing Roundtable.
- Otani, N, Dang, D, Dangi, S, **Stees, M**, Shontz, SM, Linte, C (2017) Assessing Cardiac Tissue Function via Action Potential Wave Imaging Using Cardiac Displacement Data. Proceedings of the VI ECCOMAS Thematic Conference on Computational Vision and Medical Image Processing (VipIMAGE).