Finite Element Analysis over Tangled Meshes:

Theory and Applications

By

Josh R. Danczyk

A dissertation submitted in partial fulfillment of

the requirement of the degree of

Doctor of Philosophy

(Mechanical Engineering)

at the

UNIVERSITY OF WISCONSIN – MADISON

2012

Date of final oral examination: 29 May 2012

The dissertation is approved by the following members of the Final Oral Committee:

Krishnan Suresh, Professor, Mechanical Engineering

Vadim Shapiro, Professor, Mechanical Engineering

Christopher Rutland, Professor, Mechanical Engineering

Tim Tautges, Adjunct Professor, Engineering Physics

James Rossmanith, Assistant Professor, Mathematics

Finite Element Analysis over Tangled Meshes: Theory and Applications

Abstract

Josh R. Danczyk

Under the Supervision of Professor Krishnan Suresh

At the University of Wisconsin – Madison

The finite element method (FEM) is recognized as a versatile simulation tool today. It fundamentally relies on a finite element mesh/discretization of the geometry that must conform to the geometry, be of 'good' quality, and not contain overlapping elements. In particular, a mesh containing overlapping elements is considered to be 'tangled', and can lead to erroneous results.

Tangled meshes can occur in modern FEA during mesh generation, mesh optimization, large-scale deformation, moving domains, and mesh morphing to name a few. If the mesh is tangled, the currently accepted solution is to untangle the mesh through node movement and topological modifications, e.g. edge-flipping or node insertion/deletion. This operation is computationally expensive, and may produce a mesh in violation of other finite element requirements, e.g. poor element quality. Alternatively, if untangling is not feasible, an entirely new mesh is sought.

This thesis shows that such treatment is unnecessary and tangled meshes can be accepted in FEA, provided the mathematical framework underlying finite element is suitably modified. Specifically, a new class of nodal shape functions is defined as an oriented linear combination of the classic element shape functions. These new shape functions resolve the ambiguity of the field, and, moreover, are proven to be in the theoretically-correct function space as required by the continuous Galerkin formulation of FEM.

With these new shape functions, a complete (re-)derivation for finite element analysis over tangled meshes is provided for elliptic boundary value problems. The derivation covers meshes consisting of simplicial elements in 1-, 2-, and 3-D and quadrilateral elements; though I believe the methodology can

be extended to other elements, e.g. hexahedral. Implementation details of the proposed methodology are also provided. In addition, it is shown that in the case of an untangled mesh, the proposed methodology exactly recovers classic finite element.

The new finite element derivation is subsequently validated through a series of tests. These tests are designed to determine whether exact solutions within the finite element space can be recovered to within machine precision. In all cases, a tangled mesh used in conjunction with the proposed finite element framework passes the test, whereas the classic finite element framework leads to erroneous results.

Two applications of tangled meshes are also discussed, the first of which is mesh morphing. Here, the idea is to 'recycle' an existing mesh by morphing it to conform to a new geometry. As previously stated, when the mesh is morphed, a tangled mesh may result. Instead of untangling the mesh, the proposed framework is used directly over the tangled mesh. Multiple examples are provided, each of which is benchmarked against a remesh (untangled) of the new geometry. The results show that direct use of a tangled mesh in combination with the proposed framework is comparable to the remesh.

As the second application, a local tangling of the mesh is used to overcome a poorly conditioned stiffness matrix stemming from elements with poor geometric quality. This technique, referred to here as *element covers*, is demonstrated for 2-D triangle meshes. An algorithm is provided and then analyzed in terms of minimum quality bounds. Examples are provided showing the effectiveness of element covers at solving ill-conditioning due to poor geometric quality.

As tangled meshes can now be considered acceptable, a number of future research directions open up; these are categorized and are briefly discussed in the thesis.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Finite Element Analysis and Mesh Requirements

Most engineering disciplines exploit the finite element method (FEM, also called finite element analysis or FEA) for predicting the physical behavior of virtual designs. While the mathematics behind the finite element method is well known, and computer implementations abound, there are fundamental problems that need to be resolved. For instance, producing quality meshes may be overly time consuming, or even infeasible, for complex geometries [1], [2]. This is due in part to the requirements placed on a mesh for it to be considered 'acceptable' for use in FEA. For example, the mesh must [3–5]:

1. Conform to the boundary of the underlying geometry;
2. Be topologically valid. For example:
    a. Each boundary of an element must be connected to at most one other element.
    b. Elements can only be connected to neighboring elements via their boundaries, e.g. the vertex of one triangle cannot be attached to the edge of another.
    c. When two neighboring elements are connected at a common boundary their respective node numberings for the boundary are opposite.
    d. Physically adjacent elements must be connected and in such a way as to comply with the previous three clauses.
3. Contain elements that are of sufficiently high quality (near equilateral); see [3], [6–10] for a more complete description of element quality and the effects of quality on FEA.
4. Obey all user defined criteria including: size controls, growth rate controls, element type(s), etc.
5. Not contain any inverted (differently oriented) or overlapping (covering the same physical space) elements.

Certainly a lot of burden is placed on the creation of a mesh, though in doing so the assembly procedure of finite element is vastly simplified. For instance, the integration performed during FEA can be done independently over each element, and the only data required during FEA is the list of nodes that

are connected to each element. The focus of this thesis is to revisit requirement number 5 above, showing that this requirement is not necessary. In other words, a mesh containing inverted and overlapping elements, but is otherwise valid (satisfies requirements 1-4 above), is acceptable for use in the finite element method, provided the finite element theory and implementations are modified appropriately.

To be specific, meshes that satisfy the topology requirements but contain inverted and overlapping elements are referred to as 'tangled meshes'; an example of a valid (not tangled) mesh is shown in Figure 1a and by moving a node, the mesh becomes tangled as in Figure 1b. Presently, use of tangled meshes is strongly and unanimously rejected by the FEA community; for example, to quote [11]: "*Because tangled meshes generate physically invalid solutions, it is imperative that such meshes are untangled."* Indeed, as will be confirmed next, commercial FEA systems yield erroneous results on a tangled mesh.

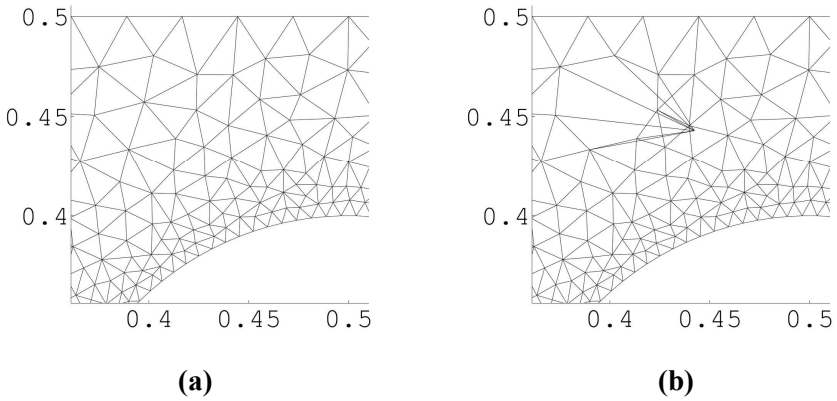

Figure 1: (a) A portion of a valid (not tangled) mesh and (b) a tangled mesh created by moving a node.

1.2 Commercial FEA and Tangled Meshes

In this section direct evidence that tangled meshes are currently not acceptable in FEA is provided. While it is unknown how commercial FEA handles such a mesh, the objective is to demonstrate that even for simple problems involving a tangled mesh commercial FEA yields erroneous results. To this end, consider a thermal conduction problem over a unit square with a thermal conductivity of 1. The left-hand boundary is set to a temperature of 0, a thermal flux of 1 is applied on the right-hand boundary, and the top and bottom edges are insulated; see Figure 2. The exact solution to this problem is $T(x, y) = x$.

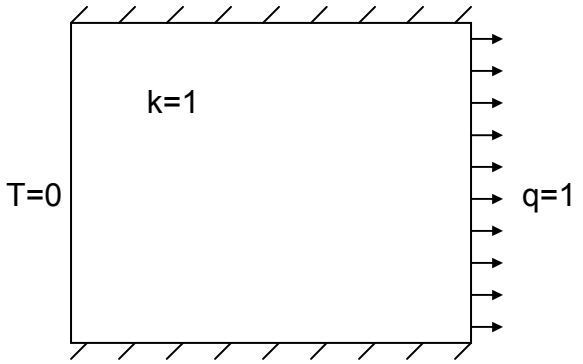

Figure 2: Thermal conduction problem definition.

The problem is solved over two triangle meshes with linear element shape functions: (1) the valid (not tangled) mesh of Figure 3a, and (2) the tangled mesh of Figure 3b that is constructed by flipping the x-location of the internal nodes in Figure 3a about the $x = 0.5$ line.

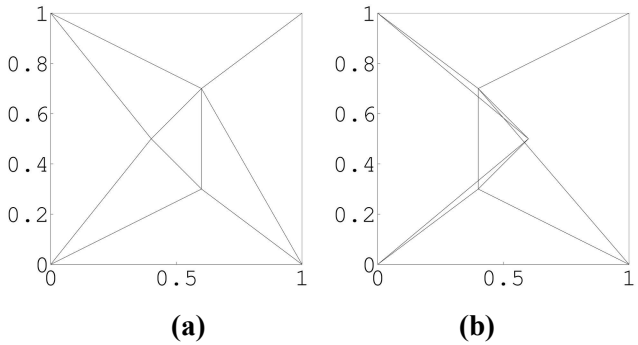

Figure 3: (a) Valid (not tangled) mesh and (b) tangled mesh.

With a valid mesh the exact solution of 1.0 is recovered at the lower right node in the commercial FEA package ANSYS 13 [12] (to within machine precision). However, when the mesh is tangled, ANSYS results in a 1.2% error. Increasing the size of the inverted element by moving the internal nodes away from each other leads to errors as large as 10%. Similar results are obtained with the commercial package Comsol 3.2 [13]; indeed, a warning is produced during the solution stage, see Figure 4.

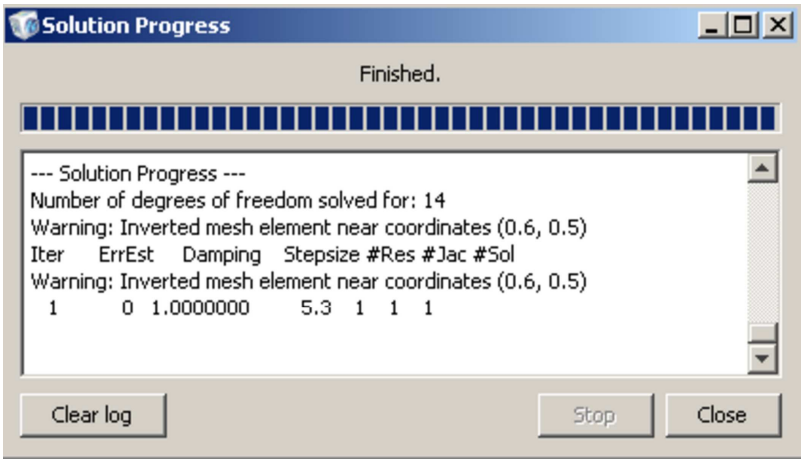

Figure 4: Comsol warning message about inverted element while attempting to solve over a tangled mesh.

1.3 Further Ramifications of Tangled Meshes

While the creation of an initial mesh is certainly limited by the requirement that the mesh cannot be tangled, it is not the only time in which this requirement comes into play. For instance, in order to satisfy element quality requirements, the mesh is typically passed through a quality optimizer. The meshing community has worked diligently on improving quality since the onset of automated meshing, and offers multiple techniques to improve the quality of an element, including node placement optimization [14–17] and topological optimization [18], [19], e.g. edge/face swapping and element insertion/merging/deleting. Each of these techniques is not without limitation, and thus most modern methods typically employ a combination of all such techniques [20–23]. Unfortunately, in all of the techniques there exists the possibility that a modification to the mesh will cause it to become tangled. This has led to the requirement and development of quality optimizers that simultaneously improve quality and untangle the mesh [11], [24], [25].

Another situation where tangled meshes occur is finite element problems involving large-scale deformations. The characteristic property of these problems is that the positions of the nodes in the mesh are continually updated as the simulation is stepped forward. Unfortunately, when there is excessive node movement relative to the local size of the finite elements tangling can occur [26–28]. Even in cases where the mesh is not directly tangled from the node movement, the movement can lead to poor quality elements

which require a pass through a quality optimizer [27–29]; this ultimately leads back to the previously described problem. If the mesh cannot be suitably modified, i.e. untangled and quality improved, a remeshing operation is required. The simulation data must then be interpolated from the original mesh onto the newly generated mesh. This process introduces error and can lead to instabilities in the finite element simulation [29].

Similar to the case of large-scale deformation, simulations involving moving/deformable domains can exhibit mesh tangling during the finite element simulation. Examples of this include contact problems [30] and fluid/solid interaction [31], [32].

One final example where mesh tangling can occur is during mesh morphing. The motivation behind mesh morphing stems from the need to perform FEA over multiple similar geometries; examples of this include design exploration and shape optimization. Instead of generating a new mesh for each geometry, the objective is to 'recycle' an existing mesh through morphing, and solve the partial differential equation over the morphed mesh [33]; see Figure 5 for an example of the mesh morphing process. An additional advantage of morphing is that there is a one-to-one correspondence between mesh nodes before and after morphing; consequently, operations such as finite difference computations are less error-prone [33]. Regardless of the morphing technique employed, mesh tangling can occur depending on the amount of morphing being performed. In addition, as with the case of large-scale deformation, element quality can be adversely affected, and may require a pass through a quality optimizer. This, as previously stated, has its own issues with regards to tangling.

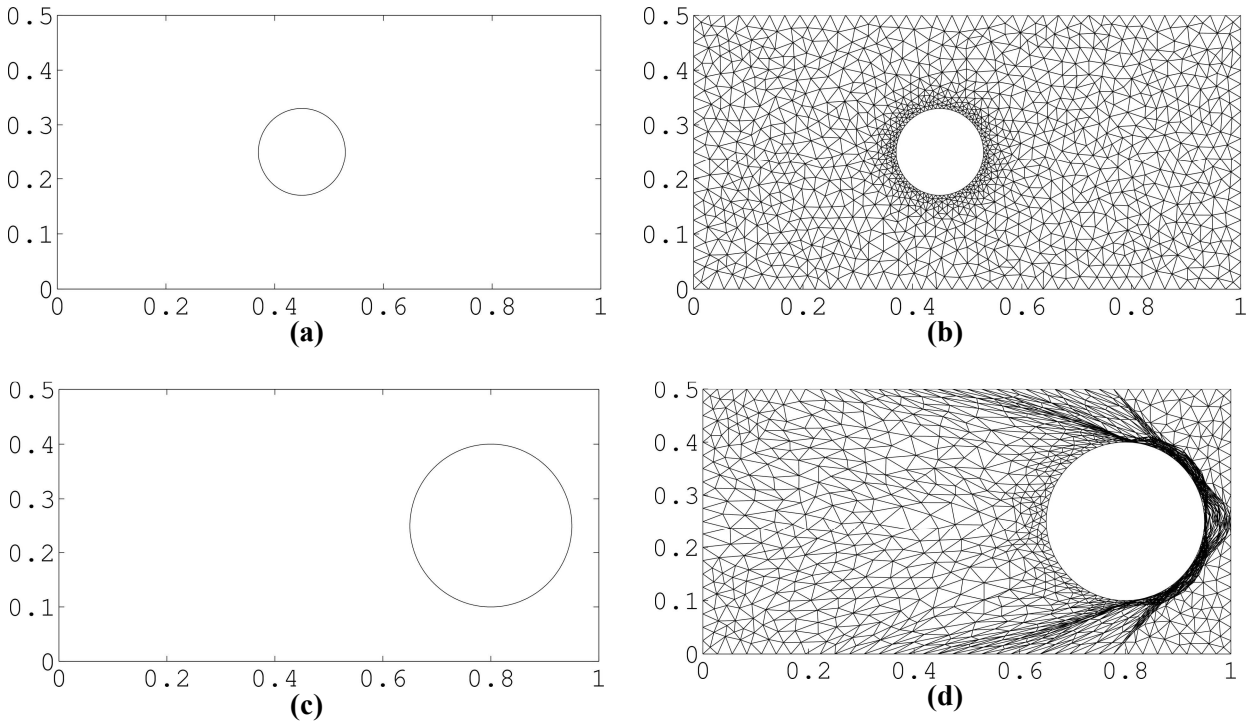

Figure 5: (a) Initial geometry, (b) initial mesh, (c) morphed geometry, and (d) morphed mesh.

In other words, tangling is unavoidable in modern FEA. Current solutions to this problem, as proposed by the research community, are discussed next.

1.4 Current Solutions to Avoid Tangled Meshes

Currently, researchers strongly and unanimously recommend untangling prior to finite element analysis. Unfortunately, untangling is perhaps as difficult as mesh generation and optimization, and has received considerable attention, even recently (since 2008) [11], [34–39], from the finite element and meshing communities [24], [25], [40–42]. Based on this untangling cannot be considered as a complete or robust solution.

While originally intended to account for sharp discontinuities in the field, so-called meshless or mesh-free method have also been promoted as a means to entirely avoid the meshing issue present in the finite element method. The signature of these methods is the reliance on a collection of nodes with no explicit connectivity between them. The name meshless is a bit of a misnomer as most of these methods do rely on a background mesh; however, the background mesh is used for integration purposes only. In other

words, unlike the finite element method, which uses the same mesh for function definition and integration, meshless methods put a schism between function approximation and integration. This splitting allows the background mesh to be a simple regular rectangular gridding or Voronoi decomposition, etc [43].

Examples of the more classic approach to meshless methods include element-free Galerkin (EFG) [44], [45], h-p cloud [46], and Meshless Local Petrov-Galerkin (MLPG) [47]. In an effort to combine the best characteristics of these classic meshless methods and the finite element method, alternative meshless method were developed, including Generalized Finite Element (GFEM) [48], [49], the Meshless Element Method [50], Natural Element Method [51], and Kantorovich's methods , e.g. Scan and Solve [52–54].

While these methods do indeed overcome the meshing issue they are not without disadvantages, the most prominent of which is computational costs and theoretical guarantees – primarily due to the form of basis functions. Additionally, as a separate grid is used for integration, the resolution of the computational domain – specifically the boundary of the domain – is not exact. This is in contrast to the finite element method, and thus meshless methods have a more difficult time enforcing essential boundary conditions. Further, there are no guarantees that back-ground meshes correctly capture the topology of the underlying design.

1.5 Outline

Instead of avoiding the problem, as the currently accepted solutions do, a different approach is taken in this thesis where the finite element framework is modified to directly account for tangling. Specifically, a new class of nodal shape functions is proposed. These shape functions draw on the classic element shape functions but are augmented with the orientation of the element, forming an oriented linear combination. The theoretical aspects of these new shape functions are explored, showing they are indeed appropriate for use in FEA when the mesh is tangled.

To promote these topics, the remainder of this document is organized as follows. In Chapter 2, the precise reason why classic FEA cannot handle tangled meshes is first shown. A new class of nodal shape functions is presented next, along with a proof on the correctness of the function space for these new

shape functions. Lastly, Chapter 2 provides a new derivation of the finite element method over tangled meshes. Chapter 3 covers implementation details pertaining to the finite element method over tangled simplicial meshes and quadrilateral meshes. Next, Chapter 4 provides numerous examples validating the proposed methodology. Two applications of tangles meshes are then provided in Chapters 5 and 6. The first application is in the field of mesh morphing and shows how performing FEA directly over a tangled mesh yields results comparable to a remeshed geometry; in other words, that it is not necessary to untangle a mesh that was tangled during the morphing process. The second application explicitly tangles a mesh to improve the geometric quality of an element. An algorithm known as element covers is additionally provided; this algorithm is shown to guarantee a minimum element quality. A discussion on the effects of this quality improvement on FEA is also provided. Concluding notes are provided in Chapter 7 as well as directions for future work in Chapter 8.

Chapter 2: Theory

2.1 Classic Finite Element

In order to illustrate precisely why classic finite element analysis (FEA), based on the continuous Galerkin formulation, cannot accommodate tangled meshes, a review of the mathematics behind FEA is required; for a complete background of FEA see, for example, [3]. Without loss of generality consider the Poisson equation in variational form [3] over some domain Ω:

$$\begin{aligned} &\text{Find } u \in H_0^1(\Omega) \\ &\int_\Omega (\nabla v)\bullet(\nabla u)\, d\Omega = \int_\Omega v f d\Omega \\ &\forall v \in H_0^1(\Omega) \end{aligned} \tag{2.1}$$

Among various aspects of the problem, the continuity requirement of $u, v \in H_0^1(\Omega)$ in Equation (2.1) is critical, and is discussed next. This continuity requirement, as will be shown later, no longer holds when the classic finite element shape functions are used over a tangled mesh.

2.1.1 Shape Functions

In FEA, the continuity requirement $u \in H_0^1(\Omega)$ is met by first constructing a mesh that discretizes the (polygonal approximation of the) geometry into quasi-disjoint elements [5]; e.g. the domain in Figure 6a is discretized via triangle elements into the mesh of Figure 6b. For simplicity, the focus is on FEA over 2-D simplicial elements; however, the conclusions are applicable in all dimensions and to non-simplicial elements.

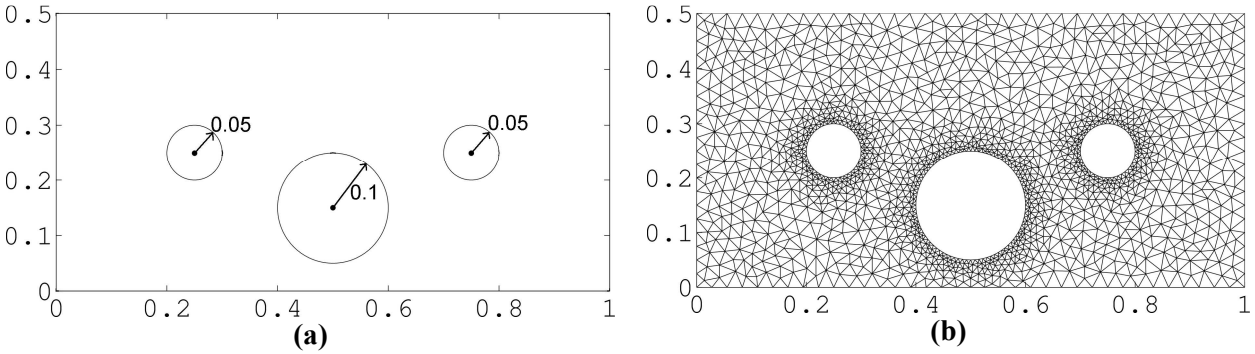

Figure 6: (a) A plate with 3 holes and (b) the quasi-disjoint decomposition (triangle mesh).

The continuity requirement $u, v \in H_0^1(\Omega)$ is satisfied in modern FEA as follows. Consider the finite element mesh in Figure 7a. Specifically, consider an interior node of the mesh, and all elements $E_1, E_2, ...$ attached to it as in Figure 7b.

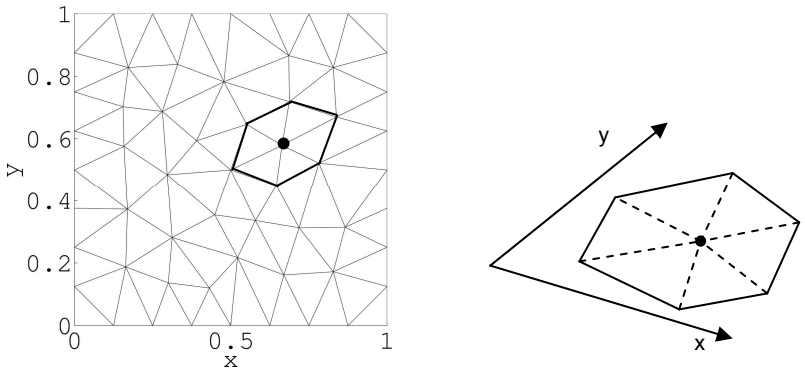

Figure 7: (a) The entire mesh and (b) the portion of the mesh surrounding a given node (circle). The region ω associated with a node is the union of elements attached it.

Let ω be the union of all elements $E_1, E_2, ...$ attached to the node. By construction of the finite element mesh [5]:

$$\omega = \bigcup_i E_i$$
$$E_i \cap E_j = \varnothing, i \neq j \tag{2.2}$$

In other words, the elements $E_1, E_2, ...$ form a quasi-disjoint decomposition of ω. Now, over each element E_i, an element shape function N_i are defined such that each function: (1) takes a value of 1 at the node identified in Figure 7b, (2) goes to zero at all other nodes of that element, and (3) is continuous across element boundaries. A typical element shape function is illustrated in Figure 8.

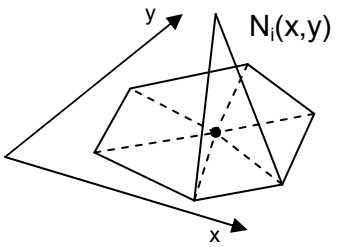

Figure 8: An element shape function N_i.

Due to the quasi-disjoint decomposition and the continuity of element shape functions, all element functions associated with a node can be 'stitched' together into single hat function ϕ over the region ω (see Figure 9). It can then be shown (page 41 of [55]) that:

$$\phi \in H_0^1(\omega) \tag{2.3}$$

In other words, ϕ is sufficiently smooth and goes to zero on the boundary of the region surrounding a node. Indeed, ϕ can be extended trivially to the remainder of the mesh.

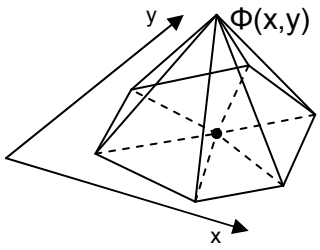

Figure 9: Hat function ϕ defined over the region ω.

This process is repeated for each node i to yield a hat function $\phi^{(i)}$. Finally, if one seeks approximate solutions of the form:

$$u = \sum_i \hat{u}^{(i)} \phi^{(i)} \tag{2.4}$$

where $\hat{u}^{(i)}$ is the degree of freedom associated with node i, then it can be shown that [4], [55]:

$$u \in H_0^1(\Omega) \tag{2.5}$$

2.1.2 Tangled Meshes in Classic FEA

Unfortunately, when a mesh gets tangled Equation (2.2) does not hold true. Consequently, for reasons discussed next, Equations (2.3) and (2.5) also do not hold true. And not surprisingly, when the continuity requirement is not satisfied classic FEA will lead to erroneous results over a tangled mesh (as illustrated earlier). In order to resolve the underlying issues, a few definitions are needed. Specifically, observe that tangling is accompanied by the 'inversion' of one or more elements. Thus the concept of element orientation is critical. The orientation of simplicial elements is defined below through the classic left/right

argument. It can be shown that these definitions are equivalent to the classic Jacobian-based definitions [11].

Definition 1: In 2D, a triangle element defined by three points abc (in that order) is **positively/negatively oriented** if the interior of the element lies to the left/right of the oriented segment ab, or equivalently bc or ca (see Figure 10). ■

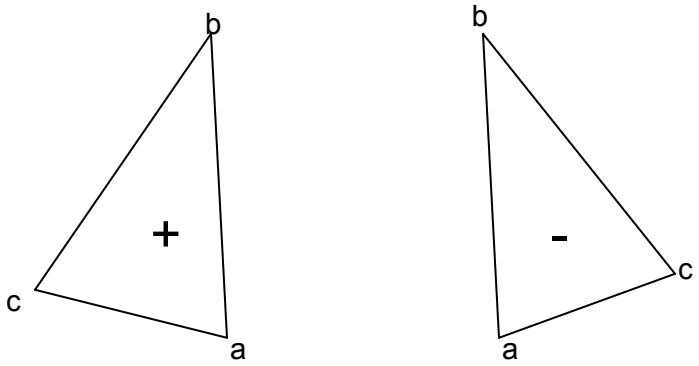

Figure 10: Example of a (left) positively oriented and (right) negatively oriented triangle.

Definition 2: In 3D, a tetrahedral element defined by four points $abcd$ (in that order) is **positively/negatively oriented** if the interior of the element lies to the left/right of the oriented plane abc (see Figure 11). ■

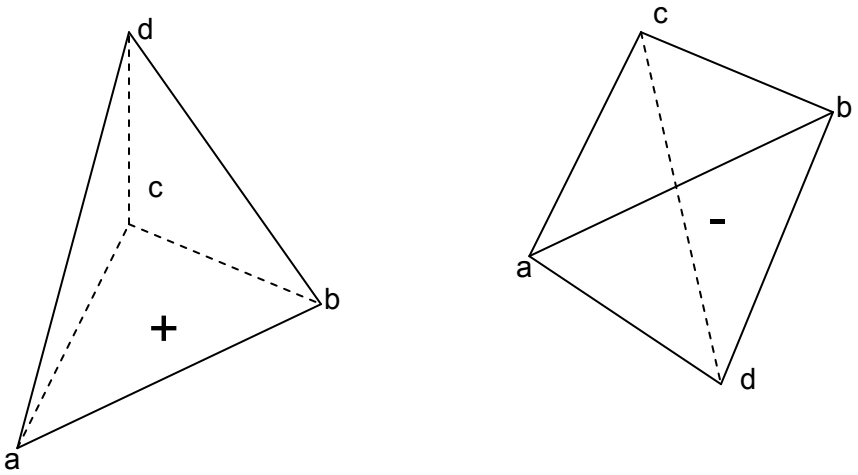

Figure 11: Example of a (left) positively oriented and (right) negatively oriented tetrahedron.

Definition 3: Let the orientation of a simplicial element E_k be denoted by Θ_k, where $\Theta_k = \pm 1$. ■

A tangled mesh is now formally defined.

Definition 4: A connected and topologically valid finite element mesh is *tangled* if it contains elements of opposite orientation. Alternatively (and equally), a mesh is tangled if through some hypothetical node movement the mesh can be transformed to a state that is connected, topologically valid, and no elements are inverted. ■

Therefore, in a tangled mesh (that is necessarily connected and topologically valid) there must exist a pair of neighboring elements with opposite orientation. Further, the two elements must overlap as can be established by the following lemma.

Lemma: In a tangled mesh, if neighboring elements are of opposite orientation, then there are points that belong to both elements, i.e., they must overlap.

Proof: In 2-D, let the two neighboring triangles be defined by abc (the positive element) and bac' (the negative element); see Figure 12. By definition, the interior of abc lies to the left of ab, and the interior of bac' lies to the right of ba, i.e., to the left of ab. Thus there are points that belong to both elements. Similar arguments hold in 3-D.■

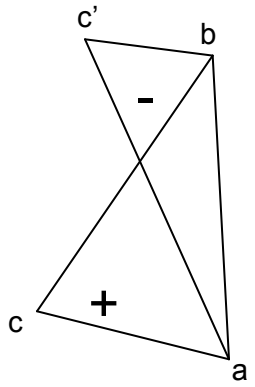

Figure 12: Positively oriented triangle abc and negatively oriented triangle bac' that are necessarily overlapping.

In other words, in a tangled mesh, an obvious conclusion is that:

$$E_i \cap E_j \neq \varnothing, \text{ for some } i \neq j \tag{2.6}$$

Note that this contradicts Equation (2.2); further, there may be elements that are not neighbors, but may still overlap. An immediate consequence of the above lemma is that the hat function ϕ, that is constructed from the element shape functions, is ill-defined/ambiguous since there are points that belong to multiple elements, and therefore multiple element shape function values can be assigned to such points; see Figure 9 and Figure 13.

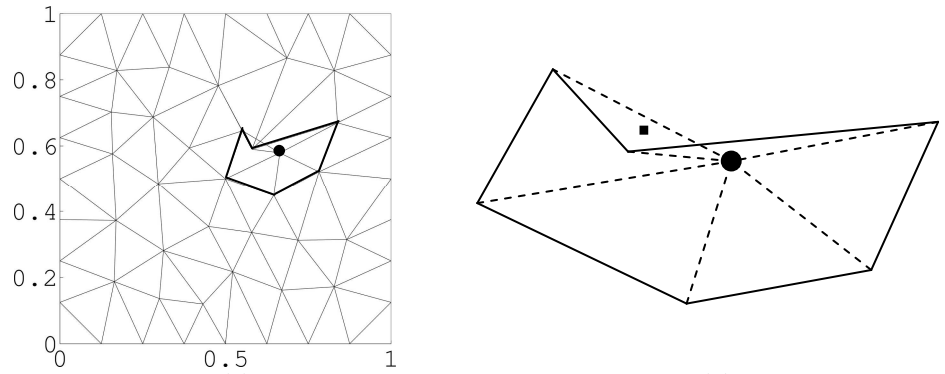

Figure 13: (a) The entire tangled mesh, and (b) the portion of the tangled mesh surrounding a given node (circle). The square represents a point that is in multiple elements, and thus has an ill-defined shape function value.

The ambiguity of ϕ can be resolved in multiple ways. For example, ϕ at points of overlap can be defined as: (1) the maximum of all element shape function values at that point, or (2) the sum/average of these values, and so on. But, observe that, in addition to resolving the ambiguity, Equation (2.3) must be satisfied. Thus, an arbitrary method of resolving the ambiguity will not do.

Indeed, overlay meshes, and therefore overlapping elements, appear in the finite element literature; for example, see the pioneering work on s-FEM [56] for hierarchical analysis. However, all elements in all such instances are positively oriented, and therefore the overlay methods do not apply to tangled meshes that contain elements of opposite orientation. The proposed strategy is discussed in the next Section.

2.2 Tangled Simplicial Meshes

2.2.1 Proposed Definition of Nodal Shape Functions

As in the previous Section, the focus is on a single node within a mesh and the set of elements attached to it. These elements are allowed to tangle in an arbitrary manner. Note that tangling of elements elsewhere in the mesh (and even over the elements of interest) is irrelevant for the current discussion since the focus is only on the hat function ϕ associated with a specific node. Tangling of elements across nodes will be accounted for, as part of the implementation, in a later Section. In addition, at present, the scope is limited to simplicial elements, e.g. lines, triangle, and tetrahedral, but is easily extended to non-simplicial elements (see Section 2.4).

Recall that ω denotes the union of all elements attached to a given node (see Equation (2.2)). Since the elements may overlap, a point within ω can belong to multiple elements and the shape function value is ill-defined. In order to overcome this failure, the region ω is first decomposed as follows. Let the *index* associated with a point within ω be the set of element numbers (integers) where the point in contained within each of the associated elements. Formally the index is defined as:

$$I(p) = \{k \mid p \in E_k\} \tag{2.7}$$

In the above definition note that only elements attached to a given node are considered. In addition, observe that while a point may belong to multiple elements (the original problem), the index at the point is unique. Grouping regions of points with like index naturally leads to a quasi-disjoint decomposition. Formally, these regions are called *cells*.

Definition 5: A *cell* is the set of all points with identical index I. ■

For example, Figure 14 illustrates seven distinct cells within ω. A cell need not be convex, or even connected.

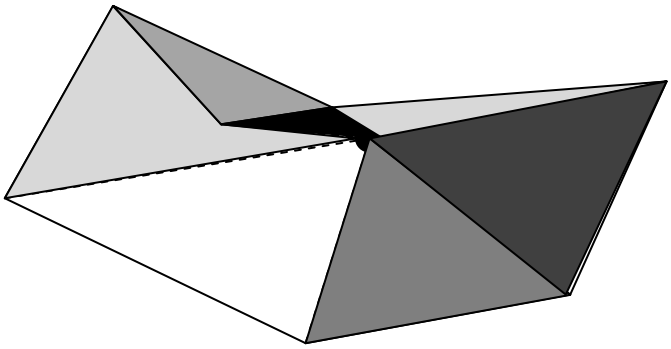

Figure 14: Cells within a tangled mesh.

Figure 15 illustrates the index associated with each cell.

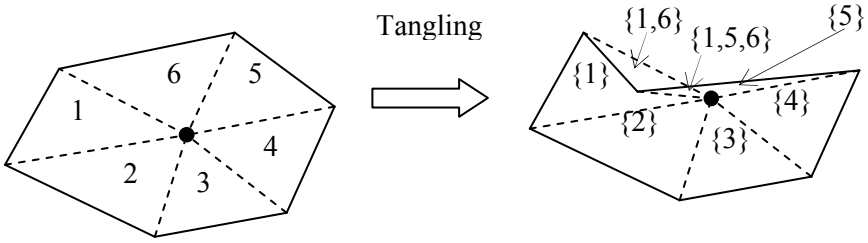

Figure 15: Cells indices.

Henceforth, cells will be denoted by $C_\alpha, C_\beta, \ldots$; observe that, by definition:

$$\begin{aligned} &\omega = \bigcup_\alpha C_\alpha \\ &C_\alpha \cap C_\beta = \varnothing, \alpha \neq \beta \end{aligned} \tag{2.8}$$

In other words, the cells induce a quasi-disjoint decomposition of ω. Next, over each cell define a *cell shape function* S_α as:

$$S_\alpha(\cdot) = \sum_{k \in I_\alpha} \Theta_k N_k(\cdot) \tag{2.9}$$

where Θ_k is the orientation of the element E_k and N_k is the shape function of the k-th element attached to the node (this is carried over from Section 2.1.1).

Theorem: *The cell shape functions defined via Equation (2.9) over the cell decomposition of Equation (2.8) satisfy the following properties: (1) they are continuous across cell-boundaries, and (2) they vanish on the boundary of* ω.

Proof: Consider the first part of the theorem. Let $C_\alpha \,\&\, C_\beta$ be two neighboring cells, with indices $I_\alpha \,\&\, I_\beta$. It must be proven that the cell shape functions $S_\alpha(\cdot)$ and $S_\beta(\cdot)$ are continuous across the common boundary. To this end, consider the difference between the two functions $S_\alpha - S_\beta$; one can group the terms into three categories:

$$S_\alpha - S_\beta = \sum_{k \in I_\alpha \cap I_\beta} [\ ] + \sum_{k \in I_\alpha \setminus I_\beta} [\ ] + \sum_{k \in I_\beta \setminus I_\alpha} [\ ] \tag{2.10}$$

The first term contains the contribution from every element k that belongs to both $I_\alpha \,\&\, I_\beta$. For every such element, N_k is continuous across the common boundary, therefore the first term vanishes on the boundary, independent of the orientation Θ_k. Next consider an element k in the second term, i.e., the element is in I_α but not in I_β. This implies that in crossing from C_α to C_β, we must exit element E_k. Observe that exiting E_k can only occur in two ways: (1) simultaneously exit ω, or (2) enter a neighboring element E_j. In the first case, N_k is necessarily zero at the boundary of point of ω. Therefore, all such contributions vanish from the second term. In the second case of exiting E_k and entering one of its neighbors E_j, there are again two cases: (2a) E_k and E_j are of the same orientation, and (2b) E_k and E_j are of opposite orientation. If the elements are of the same orientation, then E_j must belong to C_β, and $\Theta_k N_k = \Theta_j N_j$. Therefore these contributions vanish from the 2nd term. Finally, if E_k and E_j are of opposite orientation, then E_j must also belong to C_α (neighbors of opposite orientation must be overlapping), therefore $\Theta_k N_k + \Theta_j N_j = \Theta_k N_k - \Theta_k N_j$, which also vanishes since the two element functions are continuous at that point. Through a similar argument, the third term of Equation (2.10) also vanishes. Thus $S_\alpha(\cdot)$ and $S_\beta(\cdot)$ are continuous across the common boundary.

Now consider the second part of the proof where it must shown that if C_α intersects the boundary of ω, then S_α is necessarily zero on the boundary. Observe that the boundary of any C_α must is also the boundary of at least one element E_k. Since we are exiting C_α, we must also be exiting the element E_k. Once again, exiting E_k can only occur in two ways: (1) simultaneously exit ω, or (2) enter a neighboring element E_j. In the first case, N_k is necessarily zero at the boundary of point of ω. In the second case E_k and E_j must necessarily be of opposite orientation (since we are also exiting ω). As before, the contributions from both elements cancel out at such boundary points, and therefore, S_α vanishes.■

Exactly as in classic FEA [55], due to the quasi-disjoint decomposition and continuity, the cell shape functions S_α can be stitched together to define ϕ at a node. Moreover, as a consequence of the above theorem Equation (2.3) holds for this definition of nodal shape functions.

2.2.2 Derivation of FEA with the Proposed Nodal Shape Functions

The previous section introduced the concept of cells and cell shape functions (based on an oriented linear combination of classic finite element shape functions) for establishing certain theoretical properties central to this thesis. However, cells are unnecessary in a practical implementation of FEA over a tangled mesh. In other words, there is no need to explicitly compute the cell decomposition; the underlying reason is shown through the following derivation of finite element over tangled meshes (again, only simplicial elements are considered in this Section).

Upon stitching the cell shape functions together an equivalent definition for the nodal shape functions is given by:

$$\phi(p) = \sum_{k|p\in E_k} \Theta_k N_k(p) \tag{2.11}$$

Observe that in the scenario where the elements are not tangled this definition recovers the classic definition from FEA. Generalizing this shape function about an arbitrary node i yields:

$$\phi^{(i)}(p) = \sum_{k|p\in E_k, E_k\in\kappa^{(i)}} \Theta_k N_k^{(i)}(p) \tag{2.12}$$

where $\kappa^{(i)}$ is the set of all elements attached to node i. This restriction is required as it is possible in a tangled mesh for a point to be in many elements that are not topologically connected to one another. These $\phi^{(i)}$ are now super-imposed to yield approximate solutions of the form:

$$u(\cdot) = \sum_i \hat{u}^{(i)} \phi^{(i)}(\cdot) \tag{2.13}$$

whereupon, terms are regrouped as:

$$u(p) = \sum_{k|p\in E_k} \Theta_k \sum_{i|E_k \in \kappa^{(i)}} \hat{u}^{(i)} N_k^{(i)}(p) \tag{2.14}$$

Observe that the inner summation is the description of the field over one element, and is identical to that which is used in classic finite element. Indeed, for a given element one can define $\mathbb{N}_k(p)$ as the collection of element shape functions over all nodes attached to the element:

$$\mathbb{N}_k(p) = \{N_k^{(i_1)}(p) \quad N_k^{(i_2)}(p) \quad \ldots \quad N_k^{(i_m)}(p)\} \tag{2.15}$$

Further, one can define the unknown degrees of freedom associated with all nodes attached to an element as:

$$\hat{\mathbf{u}}_k = \{\hat{u}^{(i_1)} \quad \hat{u}^{(i_2)} \quad \ldots \quad \hat{u}^{(i_m)}\} \tag{2.16}$$

This leads to:

$$u(p) = \sum_{k|p\in E_k} \Theta_k \mathbb{N}_k(p) \hat{\mathbf{u}}_k \tag{2.17}$$

As with classic finite element, Equation (2.17) is substituted into Equation (2.1) (the variational form of the Poisson equation) and the auxiliary degrees of freedom are eliminated; this results in the following matrix equation:

$$\left[\int_\Omega \left(\nabla \sum_j \Theta_j \mathbb{N}_j \right)^T \bullet \left(\nabla \sum_k \Theta_k \mathbb{N}_k \right) d\Omega \right] \hat{\mathbf{u}} = \int_\Omega \left(\sum_j \Theta_j \mathbb{N}_j \right) f d\Omega \tag{2.18}$$

The left-hand side of Equation (2.18) can be regrouped into two terms, namely:

$$\sum_j \int_\Omega \Theta_j^2 \nabla \mathbb{N}_j^T \bullet \nabla \mathbb{N}_j d\Omega + \sum_j \sum_{k\neq j} \int_\Omega \Theta_j \Theta_k \nabla \mathbb{N}_j^T \bullet \nabla \mathbb{N}_k d\Omega \tag{2.19}$$

Since $\mathbb{N}_k(p)$ is zero outside of element k, the integration in Equation (2.19) can be restricted to the element itself. In addition, as $\Theta_j = \pm 1, \Theta_j^2 = 1$.

$$\sum_j \int_{E_j} \nabla \mathbb{N}_j^T \bullet \nabla \mathbb{N}_j d\Omega + \sum_j \sum_{k \neq j} \int_{E_j \cap E_k} \Theta_j \Theta_k \nabla \mathbb{N}_j^T \bullet \nabla \mathbb{N}_k d\Omega \tag{2.20}$$

The first term of Equation (2.20) is exactly the term that produces the stiffness matrix in classic FEA. The second term, on the other hand, captures the coupling between overlapping elements. Observe that, while there may be many overlapping elements that contain the same point, only two elements need to be considered at a time – far simpler than computing the actual cells, but certainly more costly than classic FEA.

In summary, to solve the Poisson equation via finite element over a tangled mesh the following terms are computed (the summation is interpreted in the usual sense of finite element assembly):

$$\begin{aligned} K_{classic} &= \sum_j \int_{E_j} \nabla \mathbb{N}_j^T \bullet \nabla \mathbb{N}_j d\Omega \\ K_{overlapping} &= \sum_j \sum_{k \neq j} \int_{E_j \cap E_k} \Theta_j \Theta_k \nabla \mathbb{N}_j^T \bullet \nabla \mathbb{N}_k d\Omega \\ f_{oriented} &= \sum_j \int_{E_j} \Theta_j \mathbb{N}_j^T f d\Omega \end{aligned} \tag{2.21}$$

The final linear system of equations is then:

$$\left(K_{classic} + K_{overlapping} \right) \hat{\mathbf{u}} = f_{oriented} \tag{2.22}$$

Observe that if there are no overlaps, then $K_{overlapping}$ is identically zero. In additions, all elements are necessarily positively oriented (see Lemma of previous Section), and therefore $f_{oriented}$ reduces to $f_{classic}$, i.e., we recover classic FEA from Equation (2.21). Further, Equation (2.21) identifies the reason why classic FEA leads to erroneous results in the presence of tangles; terms are not captured with proper orientation (as with $f_{oriented}$) and overlapping elements are simply unaccounted for (as with $K_{overlapping}$).

2.3 Comments on the Convention for Orientation

Equations (2.21),(2.22) detail the mathematical operations required to perform FEA over tangled, simplicial meshes. However, there is a subtlety of performing the integration of Equations (2.21) that

must be addressed; specifically in regards to orientation. Recall that in deriving Equation (2.21) the domain of integration was restricted from the whole domain Ω to the region in which the integrand is non-zero (either an element or an intersection of two elements). In either case, the integration must preserve the original meaning, i.e. it must be performed in the positive sense, regardless of orientation. Another way to state this is that the differential element must be positively oriented and orientation comes into play strictly as Θ_k. The reason for this is illustrated next.

Typically in FEA numeric integration is performed and is facilitated through the use of a standard element and a parametric map from the standard element to the physical element [3]; an example of this is shown in Figure 16. This change of coordinates is easily accounted for via the Jacobian which is defined (in 2D) as:

$$J = \nabla_{st} \begin{bmatrix} x & y \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial s} & \frac{\partial y}{\partial s} \\ \frac{\partial x}{\partial t} & \frac{\partial y}{\partial t} \end{bmatrix} \tag{2.23}$$

Continuing with the change of coordinate, the differential element is rewritten as

$$d\Omega = dxdy = |J| dsdt \tag{2.24}$$

where $|\bullet|$ denotes the determinant operator. The domain of integration thus becomes the standard element.

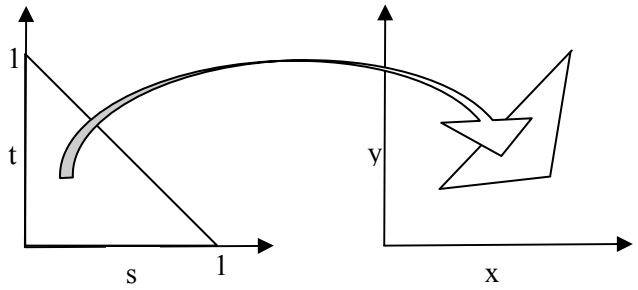

Figure 16: Parametric map from the standard element in an (s,t) coordinate system to the (x,y) coordinate system.

Now, consider the difference between evaluating the area of a positively oriented triangle and it's opposite (formed by reversing the node ordering). For example, let the three nodes be

$$(x,y)=\left\{(0,0)\quad(1,1/2)\quad(1/2,2/3)\right\} \tag{2.25}$$

with the positive triangle being defined by nodes $1,2,3$ and the negative triangle being defined by nodes $1,3,2$. The Jacobian of the positive element is then

$$J^{+}=\nabla_{st}\left(\begin{bmatrix}1-s-t & s & t\end{bmatrix}\begin{bmatrix}0 & 0\\ 1 & 1/2\\ 1/2 & 2/3\end{bmatrix}\right)=\begin{bmatrix}-1 & 1 & 0\\ -1 & 0 & 1\end{bmatrix}\begin{bmatrix}0 & 0\\ 1 & 1/2\\ 1/2 & 2/3\end{bmatrix}=\begin{bmatrix}1 & 1/2\\ 1/2 & 2/3\end{bmatrix} \tag{2.26}$$

while the Jacobian of the negative element is

$$J^{-}=\nabla_{st}\left(\begin{bmatrix}1-s-t & s & t\end{bmatrix}\begin{bmatrix}0 & 0\\ 1/2 & 2/3\\ 1 & 1/2\end{bmatrix}\right)=\begin{bmatrix}-1 & 1 & 0\\ -1 & 0 & 1\end{bmatrix}\begin{bmatrix}0 & 0\\ 1/2 & 2/3\\ 1 & 1/2\end{bmatrix}=\begin{bmatrix}1/2 & 2/3\\ 1 & 1/2\end{bmatrix} \tag{2.27}$$

Finally, the area of the positively oriented triangle is

$$A^{+}=\int_{E^{+}}d\Omega=\int_{E_{\text{standard}}}\left|J^{+}\right|dsdt=5/24 \tag{2.28}$$

whereas the area of the negative element is

$$A^{-}=\int_{E^{-}}d\Omega=\int_{E_{\text{standard}}}\left|J^{-}\right|dsdt=-5/24 \tag{2.29}$$

or exactly opposite the area of the positively oriented version of this triangle. This leads to the conclusion that changing the node numbering in this manner not only changes the orientation, but also leads to a negated area.

Taking this example further, consider a non-tangled mesh of the domain Ω in which all elements are positively oriented. Certainly the sum of the areas of each element will result in the total area of Ω. Now, reverse the node numbering of every element so that the mesh consists of all negatively oriented elements (the mesh is still not tangled). In this case the sum of the areas of each element does *not* result in the total area of Ω, but rather just the opposite. This cannot make sense as the area of a domain is a positive quantity, regardless of the orientation convention used.

In order to rectify this situation, the orientation that gets embedded into the determinant of the Jacobian must be removed. For simplicial elements this is trivial and can be accounted for by introducing an absolute value around the offending term (for a treatment of non-simplicial see Section 2.4). Observe that the Jacobian is not modified; only the determinant of the Jacobian requires an absolute value.

It is interesting to note the behavior of the gradients with respect to the orientation of the element. Specifically, as a change of coordinates is performed the gradient must be suitably modified to

$$\nabla \mathbb{N}_j = \nabla_{xy} \mathbb{N}_j = J^{-1} \nabla_{st} \mathbb{N}_j \tag{2.30}$$

where the subscript on the gradient operator denotes the coordinates to take the gradient with respect to. With linear shape function the gradient for the positive element from above is

$$\begin{bmatrix} \frac{du}{dx} \\ \frac{du}{dy} \end{bmatrix}^{+} = \frac{2}{5} \begin{bmatrix} -1 & 4 & -3 \\ -3 & -3 & 6 \end{bmatrix} \begin{bmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \hat{u}_3 \end{bmatrix} \tag{2.31}$$

whereas the gradient for the negative element is

$$\begin{bmatrix} \frac{du}{dx} \\ \frac{du}{dy} \end{bmatrix}^{-} = \frac{2}{5} \begin{bmatrix} -1 & -3 & 4 \\ -3 & 6 & -3 \end{bmatrix} \begin{bmatrix} \hat{u}_1 \\ \hat{u}_3 \\ \hat{u}_2 \end{bmatrix} \tag{2.32}$$

Observe that the two equations are in fact identical. In other words, regardless of the orientation of the element Equation (2.30) holds, i.e. the gradient of the field in the physical coordinate system does not depend on the orientation of the element.

Based on these two results, the behavior of the differential element and gradients, it is clear that the orientation of an element is only, and need only be, captured by Θ_k. Orientation is not hidden in any other term.

The last point to make in regards to orientation follows from analyzing the effects of orientation on $K_{overlapping}$ and $f_{oriented}$. Specifically, consider the effect of changing the orientation of every element in a

tangled mesh. Based on the previous discussion, $K_{classic}$ does not change; however, $K_{overlapping}$ and $f_{oriented}$ will be negated. This leads to two different systems with two different solutions:

$$\begin{aligned}\left(K_{classic}+K_{overlapping}\right)\hat{\mathbf{u}}&=f_{oriented}\\ \left(K_{classic}-K_{overlapping}\right)\hat{\mathbf{u}}&=-f_{oriented}\end{aligned} \tag{2.33}$$

In order to rectify this discrepancy it is imperative that the convention for orientation be consistent with the finite element method, or rather how it is used within the context of FEM.

The resolution, again, comes from analyzing how area is computed. Specifically, consider computing the oriented area of an element as:

$$A_k=\int_{E_k}\Theta_k d\Omega \tag{2.34}$$

Recall that $d\Omega$ is always taken in the positive sense. Upon summing over all elements the total oriented area will be either the area of the domain (a positive value) or just the opposite, depending on the overall orientation of the mesh. However, only the mesh that yields a positive total oriented area will result in $K_{overlapping}$ and $f_{oriented}$ that is consistent with $K_{classic}$. To put this more succinctly, the notion of 'positive' and 'negative' orientation is governed by how orientation is used in finite element, and for Equations (2.21),(2.22) to be valid/consistent the overall orientation of a mesh (changed by inverting every element) must result in a positive total oriented area.

Note that these two principles, that $d\Omega$ is always taken in the positive sense and the rule for selecting the overall orientation of a mesh, place only minor limitations on node numbering (beyond what is already required by finite element). As a mesh must be topologically valid the only allowable difference in node numbering schemes is one that inverts all elements. In this case all oriented quantities are off by a minus sign, but this can be detected by computing the sign of the total oriented area. The entire mesh can then be inverted if necessary.

Perhaps a standardization of node numbering, for example [57], can alleviate any confusion about the overall orientation of a mesh.

2.4 Generalization to Tangled Non-Simplicial Meshes

The previous Sections focused on finite element analysis over tangled simplicial meshes. Here, the methodology is generalized and extended to handle non-simplicial elements. Before embarking on a study of finite element analysis for tangled non-simplicial meshes, a few properties of non-simplicial elements are shown. While the focus is on 2D quadrilateral (quad) elements, the concepts and methods for handling these non-simplicial elements should be extendable to all non-simplicial elements, e.g. high-order triangle, hexahedral, or prism elements.

The most critical difference between simplicial and non-simplicial elements (in the current context) is that non-simplicial elements can contain points that lie outside of what is colloquially considered to be the interior of an element. To see this property, consider the parametric mapping from the standard element to the physical quadrilateral in Figure 17; the nodes of the physical element are at $(x,y)=\left\{(0,0)\quad(1,0)\quad(1/4,1/4)\quad(0,1)\right\}$. If the standard bilinear parametric map is used [3], i.e.

$$\begin{aligned} x(s,t) &= \frac{5+5s-3t-3st}{16} \\ y(s,t) &= \frac{5-3s+5t-3st}{16} \end{aligned} \tag{2.35}$$

then the locus of all point in the physical domain generated by all points in the standard domain will contain points that are 'outside' the quad. For example, the point 'b' in Figure 17, i.e. $(s,t)=(1/3,1/3)$, is mapped to the point $(x,y)=(1/3,1/3)$, which is certainly not in the confines of the edges of the quad.

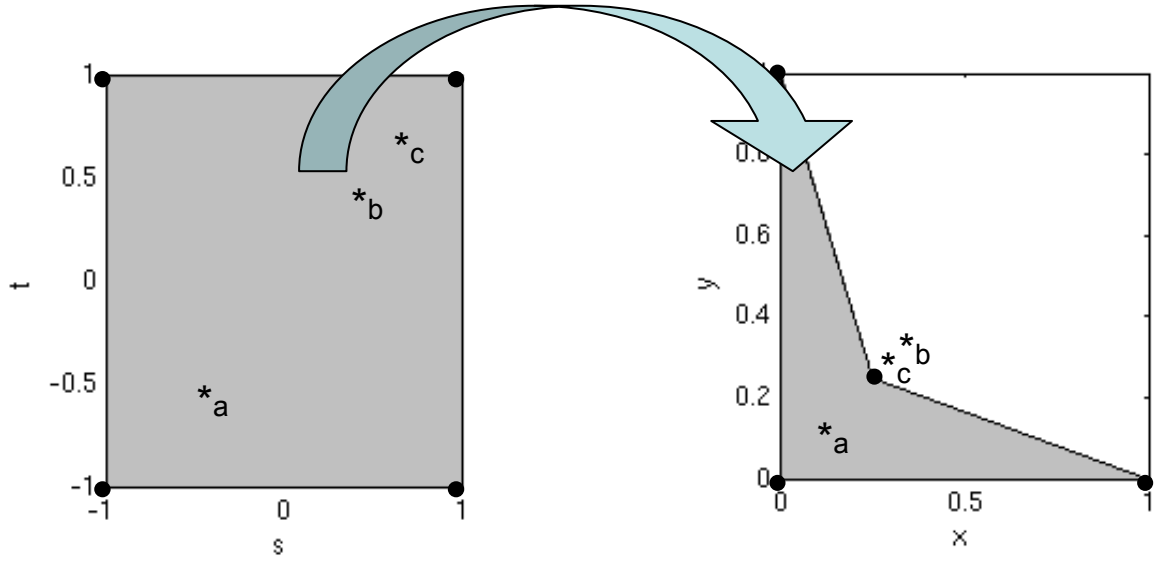

Figure 17: Parametric mapping from (left) standard element to (right) physical element. The original and mapped locations of some points 'a', 'b', 'c' are also shown.

In addition to points being outside the element, some points in the standard element are mapped to the same physical location. For instance, $(s,t)=\left(-\frac{1}{3},-\frac{1}{3}\right)$ *and* $(s,t)=(1,1)$ get mapped to $(x,y)=\left(\frac{1}{4},\frac{1}{4}\right)$.

Another characteristic of non-simplicial element is that, as opposed to simplicial elements, such an element can simultaneously be positively and negatively oriented (based on the sign of the determinant of the Jacobian). For the parametric mapping defined in Equation (2.35) the determinant of the Jacobian is

$$|J|=\frac{2-3s-3t}{32} \tag{2.36}$$

Consequently, at $(s,t)=(0,0)$, $|J|=\frac{1}{16}$ and at $(s,t)=(1,1)$, $|J|=-\frac{1}{8}$; they are of opposite orientation. Figure 18 shows the orientation that every point in the standard element generates. Note that orientation is a property of the physical domain and not the standard domain, but it is induced by the parametric mapping between the two (hence the wording of the previous sentence). In other words, every point in the standard domain is mapped to a point in the physical domain and an orientation is attached to the physical point.

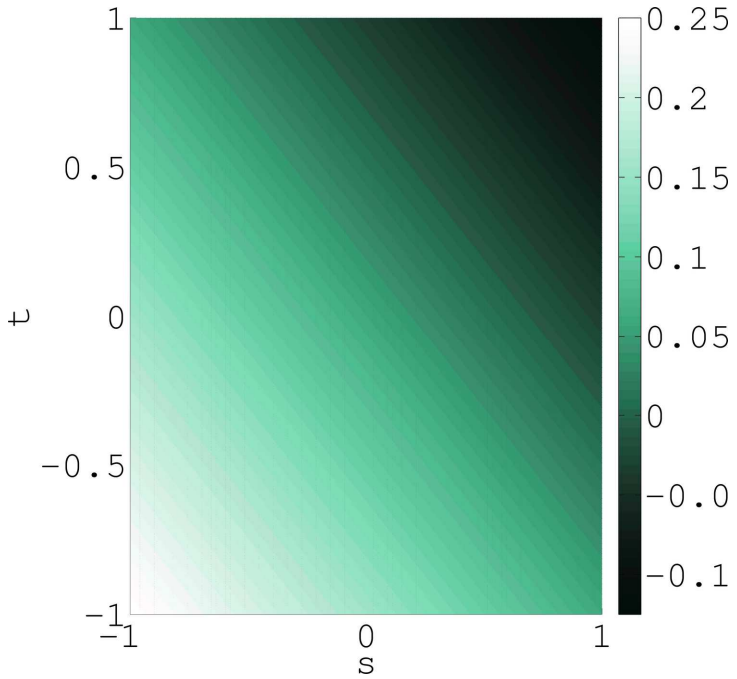

Figure 18: Plot of determinant of Jacobian over the standard element.

Splitting the standard element into regions of the same orientation and then applying the parametric map to those regions results in the two physical regions shown in Figure 19. Note that these two physical regions are connected topologically by a boundary that is the map of points in the standard domain in which the determinant of Jacobian is zero; shown as a thick line in Figure 19. Observe that this is just another form of tangling that is internal to a single element.

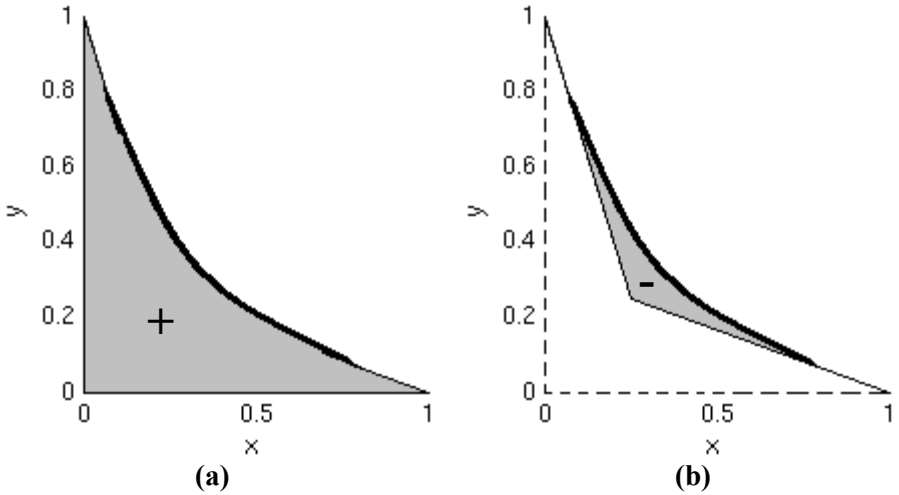

Figure 19: Portions of the quadrilateral element that are (a) positively oriented and (b) negatively oriented. The thick line is a map of the points in the standard element where the determinant of the Jacobian is zero.

2.4.1 Proposed Definition of Nodal Shape Functions

The properties just discussed are now put to use by showing that the oriented linear combination proposed in Section 2.2.1 will guarantee that even for non-simplicial meshes the nodal basis functions are in $H_0^1(\Omega)$. As with the case of simplicial elements, the focus is on one node and the collection of elements connected to it. As non-simplicial elements can have internal tangling the following definitions are essential. For a physical element and an associated map from the standard element to the physical element in which the map is continuous and differentiable:

Definition 6: A closed, connected subset of the standard element in which all points generate physical points of the same orientation is defined as a *sub-element* of the standard element. ■

Certainly this implies that for any physical element the associated standard element can be decomposed into a quasi-disjoint union of sub-elements. The choice of decomposition is arbitrary (provided Definition 6 is upheld), and it is valid to have adjacent sub-elements with the same orientation. This freedom allows finite element implementers to best choose a decomposition scheme. Note that it is also valid for this decomposition to involve only one sub-element, e.g. simplicial elements or convex quads.

Definition 7: The map of a sub-element of the standard element is a *sub-element* of the physical element.

■

The choice of using 'sub-element' in both contexts is intentional as there is a one-to-one correspondence between the two. Ambiguity will not arise as the use of the term sub-element is context dependent, i.e. when talking about a standard element the former definition is obviously in play.

Now, given two sub-elements of the standard element that are connected by a common boundary, the associated sub-elements of the physical element are also topologically connected; the connection is though the map of the aforementioned boundary. Moreover, any continuous function that is defined over the standard element and subsequently mapped into the physical domain is consistent (same valued) along this boundary between sub-elements.

Mirroring the work of Section 2.2.1, cells and cell shape functions are used to define nodal shape functions, though one addendum is made: Instead of defining cells via elements, cells are now defined via element/sub-element pairs. More concretely, the index pairs associated with a point is defined as

$$I(p) = \{k, b \mid p \in E_{k,b}\} \tag{2.37}$$

where $E_{k,b}$ is the b-th sub-element of the k-th element attached to the node. The notion of a cell (definition 5) is retained, albeit with Equation (2.37) defining the index. With the extension to sub-elements, cells now define a quasi-disjoint decomposition of ω, i.e. Equation (2.8) still holds, where ω is the union of all points covered by all sub-elements of all elements attached to the node.

Cell shape functions are then defined as

$$S_\alpha(\cdot) = \sum_{k,b \in I_\alpha} \Theta_{k,b} N_{k,b}(\cdot) \tag{2.38}$$

where $\Theta_{k,b}$ is the orientation of the b-th sub-element of the k-th element attached to the node, and $N_{k,b}$ is the restriction of the classic finite element shape function N_k to the b-th sub-element. Observe that by construction $\Theta_{k,b}$ is constant (as opposed to Θ_k, the orientation of the whole non-simplicial element). An extension to non-simplicial elements for the theorem in Section 2.2.1 can finally be stated.

Theorem: *The cell shape functions defined via Equation* (2.38) *over the cell decomposition of Equation (2.8) satisfy the following properties: (1) they are continuous across cell-boundaries, and (2) they vanish on the boundary of* ω.

Observe that all of the properties required to perform the original proof (from Section 2.2.1) still hold here, albeit for sub-elements instead of elements. In summary these properties amount to connectivity of sub-elements and the continuity of $N_{k,b}$ across connected sub-elements. The proof is therefore trivial and thus omitted.

Exactly as in classic FEA [55] and the previous derivation for tangled simplicial meshes, due to the quasi-disjoint decomposition and continuity, the cell shape functions S_α can be stitched together to define ϕ at a node. Moreover, as a consequence of the above theorem Equation (2.3) holds for this definition of nodal shape functions.

2.4.2 Derivation of FEA with the Proposed Nodal Shape Functions

Similar to the case of simplicial meshes, an alternative form of Equation (2.38), whereby all cell shape functions are stitched together, is given by

$$\phi(p) = \sum_{k,b|p\in E_{k,b}} \Theta_{k,b} N_{k,b}(p) \tag{2.39}$$

This shape function is then generalized to an arbitrary node in the mesh as

$$\phi^{(i)}(p) = \sum_{k,b|p\in E_{k,b}, E_k\in\kappa^{(i)}} \Theta_{k,b} N^{(i)}_{k,b}(p) \tag{2.40}$$

and then substituted into Equation (2.4) to obtain

$$u(p) = \sum_{k,b|p\in E_{k,b}} \Theta_{k,b} \sum_{i|E_k\in\kappa^{(i)}} \hat{u}^{(i)} N^{(i)}_{k,b}(p) \tag{2.41}$$

The inner summation is the description of the field over one sub-element, or alternatively, the restriction of the field over one element to the sub-element. Through a natural extension of the definition of $\mathbb{N}_k$ to $\mathbb{N}_{k,b}$ the following is obtained.

$$u(p) = \sum_{k,b|p\in E_{k,b}} \Theta_{k,b} \mathbb{N}_{k,b}(p) \hat{\mathbf{u}}_k \tag{2.42}$$

Substituting the above Equation into Equation (2.1) (the variational form of the Poisson problem) and eliminating the auxiliary degrees of freedom yields:

$$\left[\int_{\Omega}\left(\nabla\sum_{j,a}\Theta_{j,a}\mathbb{N}_{j,a}\right)^{T}\bullet\left(\nabla\sum_{k,b}\Theta_{k,b}\mathbb{N}_{k,b}\right)d\Omega\right]\hat{\mathbf{u}}=\int_{\Omega}\left(\sum_{k,b}\Theta_{k,b}\mathbb{N}_{k,b}\right)fd\Omega \tag{2.43}$$

Regrouping the terms results in

$$\left[\sum_{j,a}\int_{\Omega}\nabla\mathbb{N}_{j,a}^{T}\bullet\nabla\mathbb{N}_{j,a}d\Omega+\sum_{j,a}\sum_{k,b\neq j,a}\Theta_{j,a}\Theta_{k,b}\int_{\Omega}\nabla\mathbb{N}_{j,a}^{T}\bullet\nabla\mathbb{N}_{k,b}d\Omega\right]\hat{\mathbf{u}}=\sum_{k,b}\Theta_{k,b}\int_{\Omega}\mathbb{N}_{k,b}fd\Omega \tag{2.44}$$

As $\mathbb{N}_{k,b}$ is zero outside the sub-element, the integration can be reduced from the whole domain to sub-element, i.e.

$$\left[\sum_{j,a}\int_{E_{j,a}}\nabla\mathbb{N}_{j,a}^{T}\bullet\nabla\mathbb{N}_{j,a}d\Omega+\sum_{j,a}\sum_{k,b\neq j,a}\Theta_{j,a}\Theta_{k,b}\int_{E_{j,a}\cap E_{k,b}}\nabla\mathbb{N}_{j,a}^{T}\bullet\nabla\mathbb{N}_{k,b}d\Omega\right]\hat{\mathbf{u}}=\sum_{k,b}\Theta_{k,b}\int_{E_{k,b}}\mathbb{N}_{k,b}fd\Omega \tag{2.45}$$

Unlike the case of simplicial meshes, this Equation cannot be conveniently reduced further to include the term $K_{classic}$ since the integration is performed over sub-elements. While the integration is performed over all sub-elements of an element, the fact that $d\Omega$ must be taken in the positive sense requires the sub-elements be treated separately and not conglomerated. Thus, Equation (2.45) is directly implemented; see Section 3.3. While this certainly increases the complexity of implementing the finite element method for tangled non-simplicial element many simplifications, especially for quad elements, are nonetheless possible. Indeed, if the mesh is not tangled Equation (2.45) simplifies to the case of classic finite element.

Chapter 3: Implementations

3.1 Identifying Overlapping Regions

The proposed methodology requires integrating over regions that are common to two elements; see Equation (2.21). Therefore, the first step is to identify elements that overlap. A naïve $O(n^2)$ implementation entails checking all elements against all other elements; this is further complicated for non-simplicial elements. In order to avoid this high algorithmic cost, a more efficient strategy that makes direct use of element orientations is used. Specifically, as stated in the Lemma in Section 2.1.2, neighboring elements with opposite orientations will contain common points, i.e. they overlap. This starting point forms the basis of the following proposed algorithm for detecting overlapping elements:

1. Seed a stack with all pairs of neighboring elements with opposite orientation (recall that each such pair must necessarily overlap).
2. While the stack is not empty:
    a. Pop the top of the stack. This yields the pair (e_1,e_2).
    b. If this pair has been processed already, continue to step a.
    c. If the elements do not overlap, continue to step a.
    d. Record the overlap.
    e. For all neighbors e_n of e_1, add (e_n,e_2) to stack.
    f. For all neighbors e_n of e_2, add (e_n,e_1) to stack.

The most computationally expensive part of the above algorithm is step c, i.e., checking if two elements overlap. However, as only simplicial elements are consider here this amounts to convex polytope/convex polytope intersection and has been extensively studied and solved [58].

In the case of non-simplicial elements the above algorithm is modified to perform sub-element/sub-element intersection queries. As the sub-elements need not be convex polytopes, the intersection query must be adequately adapted. One possibility is to approximate the sub-element through a disjoint union of convex polytopes. Regardless, such operations have been extensively studied and solved [59].

3.1.1 Proof of Correctness

In order to establish the correctness of the algorithm, an assumption is made regarding the tangled mesh.

Assumption: If two elements overlap, there exists a neighbor of one of the elements that the other element overlaps, i.e., if $E_i \cap E_j \neq \varnothing$, then $\exists E_k \in N(E_i) : E_k \cap E_j \neq \varnothing$ or $\exists E_k \in N(E_j) : E_i \cap E_k \neq \varnothing$.

In other words, the general scenario illustrated in Figure 20 is valid; however scenarios where two geometrically identical elements overlap *exactly* are disallowed.

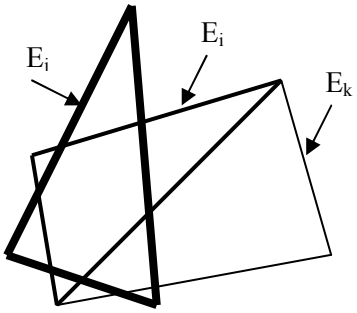

Figure 20: Three elements of a mesh in a general position. Element E_j is overlapping element E_i and E_i's neighbor E_k.

Given this assumption, the following theorem on the correctness of the above algorithm can be established.

Theorem: In a connected tangled mesh, all overlapping element pairs are detected via the above algorithm, i.e., recorded via the stack.

Proof: Assume the contrary, i.e., assume that there exists an overlapping pair (E_i, E_j) that was never inserted into the stack; in other words the algorithm failed to identify this pair. Now, let E_k be a neighbor of E_i. From the assumption above, $E_k \cap E_j \neq \varnothing$. Further, from steps 2e and 2f of the above algorithm it follows that the pair (E_k, E_j) must have gone undetected too, i.e., could not have been part of the stack. This argument holds *ad infinitum* leading to a conclusion that no element pair that is topologically connected to the pair (E_i, E_j) could have been in the stack. This is not possible since step 1 of the above

algorithm seeds the stack with all pairs of elements that are neighbors and of opposite orientation. Thus, by contradiction, the pair $\left(E_i, E_j\right)$ will be detected by the above algorithm. ■

In the case that two elements exactly overlap, i.e., the above assumption is violated, such element pairs are also seeded onto the stack. Since the elements must overlap exactly, this detection becomes a matter of finding duplicate nodes and analyzing the elements attached to the duplicates.

3.1.2 Algorithmic Complexity

Let m be the (worst case) number of neighbors for an element in a mesh. For example, $m = 3$ for a triangle mesh, $m = 4$ or a tetrahedral mesh, and $m = 4$ for a mixed triangle/quad mesh. Let n be the number of overlapping element pairs in a tangled mesh. For each overlapping element pair, steps 2e and 2f of the above algorithm will push at most $2m$ additional element pairs onto the stack. Further step 2b ensures that an element pair is only processed once. Therefore, the total number of element pairs that get pushed to the stack, and thus checked for overlap, is at most $2mn$, which is linear with respect to n.

3.2 The Finite Element Assembly Procedure – Simplicial Meshes

Performing FEA over a tangled simplicial mesh requires computing the terms in Equations (2.21) which are as follows. First, $K_{classic}$ is computed exactly as in classic FEA. Next, $f_{oriented}$ is computed exactly as in classic FEA with one modification: The orientation of the element must be accounted for. This is accomplished by multiplying the local components of $f_{oriented}$ for each element by the elements orientation (± 1) before adding to the global $f_{oriented}$ term.

In order to compute $K_{overlapping}$ the overlapping regions must first be computed (see previous Section). Each of these regions denotes the intersection of two elements that are overlapping and the parent elements that are involved in the intersection, see Figure 21a. In order to perform the integration over this region, the region is subsequently tessellated, see Figure 21b.

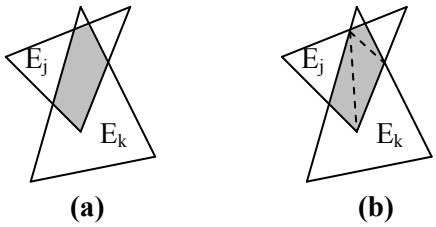

Figure 21: (a) Overlapping region of elements E_j and E_k, and (b) the triangulation of the overlapping region.

Now, as with classic FEA, numeric integration is performed over the Gauss points of the newly formed triangles. To compute $\nabla\mathbb{N}_j$ and $\nabla\mathbb{N}_k$ at the Gauss points, the Gauss points from the parametric element are mapped to the physical (x,y) space and then the parametric coordinates of the parent elements E_j, E_k that coincide with the (x,y) location are found. $\nabla\mathbb{N}_j$ and $\nabla\mathbb{N}_k$ are thus computed at their respective parametric coordinates. As with $f_{oriented}$, before adding the local stiffness matrix to the global $K_{overlapping}$ matrix the orientations of the parent elements are accounted for.

3.3 The Finite Element Assembly Procedure – Quadrilateral Meshes

Performing FEA over a tangled quadrilateral mesh requires computing the terms in Equation (2.45). The first step involves decomposing elements into sub-elements by analyzing the determinant of the Jacobian over the standard element. Fortunately, this decomposition is trivial for quads: Let the element be defined by nodes $\{(x_1,y_1),(x_2,y_2),(x_3,y_3),(x_4,y_4)\}$; see Figure 22 for ordering.

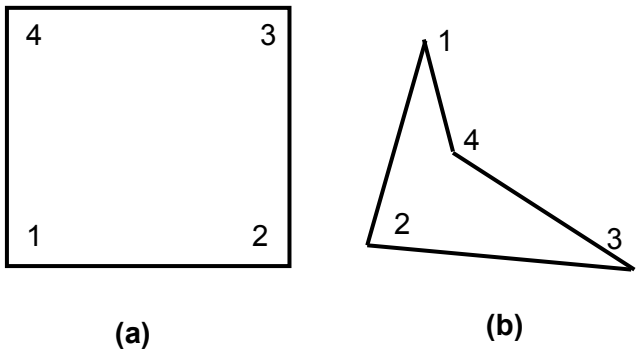

Figure 22: Node ordering of (a) the standard element and (b) the corresponding physical element.

Further, let the element shape functions be [3]

$$\mathbb{N} = \begin{Bmatrix} (1-s)(1-t) & s(1-t) & st & (1-s)t \end{Bmatrix} \tag{3.1}$$

The Jacobian is then

$$J = \begin{bmatrix} \frac{\partial x}{\partial s} & \frac{\partial y}{\partial s} \\ \frac{\partial x}{\partial t} & \frac{\partial y}{\partial t} \end{bmatrix} = \begin{bmatrix} t-1 & 1-t & t & -t \\ s-1 & -s & s & 1-s \end{bmatrix} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix} \tag{3.2}$$

Taking the determinant and simplifying yields

$$|J| = \psi_0 + \psi_1 s + \psi_2 t \tag{3.3}$$

where

$$\begin{aligned} \psi_0 &= (x_2 - x_1)(y_4 - y_1) - (y_2 - y_1)(x_4 - x_1) \\ \psi_1 &= (x_2 - x_1)(y_1 - y_2 + y_3 - y_4) - (y_2 - y_1)(x_1 - x_2 + x_3 - x_4) \\ \psi_2 &= (x_1 - x_2 + x_3 - x_4)(y_4 - y_1) - (y_1 - y_2 + y_3 - y_4)(x_4 - x_1) \end{aligned} \tag{3.4}$$

Since the determinant of the Jacobian is a linear field there will be at most 2 sub-elements, corresponding to the subsets of the standard element that are to the left and right of the $|J| = 0$ line. If there is only 1 sub-element (the $|J| = 0$ line does not intersect the standard element), the sub-element is the whole quad and necessarily a convex polytope. Once the decomposition is made, overlapping sub-elements are identified through the algorithm in Section 3.1.

Finite element assembly is now performed in a manner similar to that of tangled simplicial meshes. A key difference occurs when integrating over the overlapping sub-elements; more specifically in computing the parametric coordinates of the parent sub-elements $E_{j,a}, E_{k,b}$ that coincide with the (x, y) location of an integration point. Unlike the case of simplicial meshes, there may be multiple (s,t) values that map to the same (x, y) point. Of the multiple (s,t) values there is, however, only one that is in the sub-element of interest.

To illustrate this, consider the quad discussed in Section 2.4, henceforth labeled E_1. Further, let the negatively oriented sub-element ($E_{1,1}$) of E_1 be overlapping some other sub-element. During the numeric

integration, the value of $\nabla\mathbb{N}_{1,1}$ is required at the point $(x,y)=(1/4,1/3)$, and so the corresponding (s,t) must first be computed. For quad elements this is found by solving the following system of bilinear equations

$$\begin{bmatrix} x_1-x_2+x_3-x_4 & x_2-x_1 & x_4-x_1 \\ y_1-y_2+y_3-y_4 & y_2-y_1 & y_4-y_1 \end{bmatrix} \begin{bmatrix} st \\ s \\ t \end{bmatrix} = \begin{bmatrix} x-x_1 \\ y-y_1 \end{bmatrix} \tag{3.5}$$

Certainly, this system can be decoupled to a quadratic equation in s and a linear relationship between s and t. Upon solving Equation (3.5), two solutions are found, namely $(0.864,0.948)$ and $(0.386,0.469)$, of which only $(0.864,0.948)$ is in the negatively oriented sub-element. With the proper value of (s,t) in hand, $\nabla\mathbb{N}_{1,1}$ can be computed and the numeric integration performed.

Chapter 4: Validation

The theory presented is now validated through 'patch tests' [3], [4], [60] over tangled meshes. These tests are designed in this thesis to identify incorrect theory/implementation by checking whether the finite elements can capture, to within machine precision, a field within the space of the basis functions. In particular, classic FEA, including ANSYS 13 [12], a commercial FEA system, is compared against the proposed methodology.

While the following tests are not true patch tests (see pg. 329-350 of [3] for a complete description of the finite element patch test), they do capture the essence of the patch test by addressing the question, "Do the finite elements capture, to within machine precision, a specified field as expected?" As such, these tests can the thought of as necessary but not sufficient to show the correctness of the proposed methodology. The reason a true patch test is not performed is that it is unclear at present how to formulate the patch test for a tangled mesh.

4.1 Patch Test – Simple Tangling

As a first example of validating the proposed methodology consider returning to the problem of thermal conduction over a unit square first described in Section 1.2. For clarity the untangled and tangled domain are repeated in Figure 23a-b.

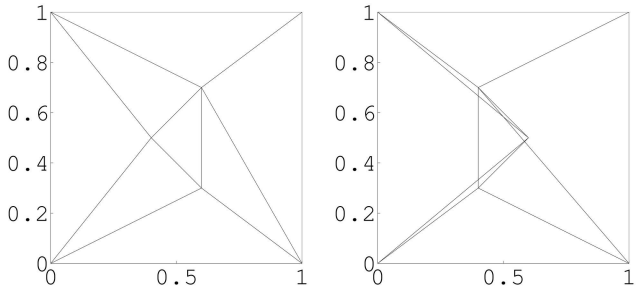

Figure 23: (a) Valid (not tangled) mesh and (b) tangled mesh.

Table 1 illustrates the solution for the two meshes at the location $(1,0)$, as produced by ANSYS 13 [12] and the proposed methodology. As expected, with a valid mesh the exact solution of 1.0 is recovered

both in ANSYS 13 and proposed methodology (to within machine precision). However, when the mesh is tangled, ANSYS results in a 1.2% error. Increasing the size of the inverted element by moving the internal nodes away from each other leads to errors as large as 10%. On the other hand, the proposed theory yields the exact solution even in the case of the tangled mesh.

Table 1: Temperature at x=1,y=0

	ANSYS 13	Proposed Theory
Valid Mesh	1.0000	1.0000
Tangled Mesh	0.9875	1.0000

4.2 Patch Test – Multiple Tangles

Next consider triangulating a unit square, see Figure 24a, and then randomizing the location of the interior nodes to produce the tangled mesh shown in Figure 24b. Random linear and quadratic fields are chosen as the exact solution to a Poisson problem; Dirichlet conditions, Neumann conditions, and body forces are computed from this field, and are applied on the tangled mesh as follows: Neumann conditions are applied at the bottom, right, and top boundaries, and Dirichlet conditions are applied to the left boundary. The objective is to recover the exact solution everywhere.

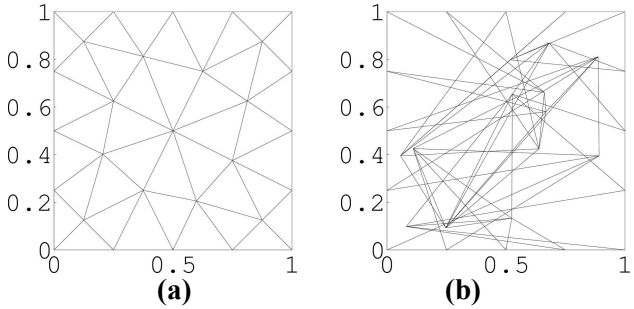

Figure 24: (a) Initial untangled mesh and (b) the actual tangled mesh used for patch tests.

Normalized errors over all nodal values (defined as $abs(\| u - u_{Exact} \|)/\| u_{Exact} \|$) from classic FEA and the proposed theory are summarized in Table 2. Observe that, once again, the proposed theory satisfies the patch test, whereas classic FEA fails the test.

Table 2: Normalized errors for 2-D Poisson problem patch test

	Classic FEA	Proposed theory
Linear	1.8438	4.9145e-13
Quadratic	7.5844	4.0306e-12

As a second patch test consider repeating the previous example but for the physics of plane stress. The tangled mesh is identical to the previous example, as are the types/locations of all boundary conditions. Normalized errors in solutions from classic FEA and the proposed theory are summarized in Table 3. As with the previous example, classic FEA fails the patch test for the tangled mesh, while the proposed theory satisfies the test.

Table 3: Normalized errors for 2-D plane stress problem patch test

	Classic FEA	Proposed theory
Linear	1.6037	2.2277e-12
Quadratic	66.4133	2.9278e-11

Two 3-D patch tests are now performed as follows. Similar to the 2-D case the domain is a unit cube, and again the interior nodal positions are randomized. The mesh consists of 69 tetrahedra, 29 vertices, and upon tangling, 553 overlapping regions were detected. A portion of the mesh before and after tangling is shown in Figure 25b-c, respectively; these are all the elements that have a face on the bottom or back of the unit cube.

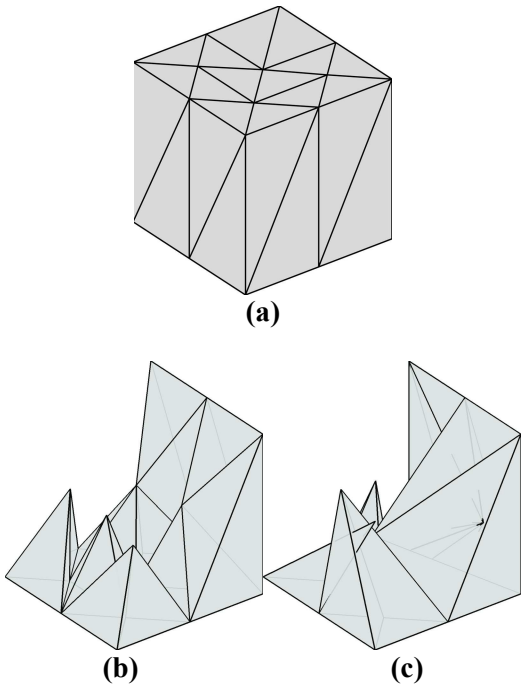

Figure 25: (a) Initial mesh, (b) portions of the initial untangled mesh, and (c) portions of the actual tangled mesh used for patch tests.

Random linear and quadratic fields are chosen as the exact solution to a Poisson problem and then an elasticity problem. Surface tractions are applied to five of the six faces and Dirichlet conditions are applied to the remaining face. Normalized errors in solutions from classic FEA and the proposed theory are summarized in Table 4 and Table 5 for the Poisson problem and elasticity problem, respectively.

Table 4: Normalized errors for 3-D Poisson problem patch test

	Classic FEA	Proposed theory
Linear	0.3296	1.7924e-15
Quadratic	1.6116	2.0738e-12

Table 5: Normalized errors for 3-D elasticity problem patch test

	Classic FEA	Proposed theory
Linear	1.4035	5.3276e-14
Quadratic	14.1704	1.9298e-11

4.3 Convergence Test

As a final means of validation the proposed methodology is tested for convergence and the convergence rate. Here, a function outside the span of the finite elements is chosen as the exact solution to a 2-D Poisson problem; this is contrary to the previous patch tests where the exact solution was within the span of the finite elements. In this case, the function is:

$$u(x,y) = e^{\pi x}\cos(\pi y) \tag{4.1}$$

As with the 2-D patch tests above, the domain is a unit square. Dirichlet conditions, Neumann conditions, and body forces are computed from the specified field, and are applied on the mesh as follows: Neumann conditions are applied at the bottom, right, and top boundaries, and Dirichlet conditions are applied to the left boundary.

The mechanism for tangling the mesh is as follows; see Figure 26 for a geometric description. First, nodes that lie a circle of radius 0.25 centered within the domain are mirrored about the center; this tangles the elements within the circle. Next, the remaining nodes are dragged towards the center of the domain according to the rule:

$$
\begin{aligned}
d_i &\leftarrow \left\| (0.5, 0.5) - p_i \right\| \\
p_i &\leftarrow p_i + 1.9 h e^{-10(0.25 - d_i)} \left((0.5, 0.5) - p_i \right)
\end{aligned}
\tag{4.2}
$$

The reason for this step is to remove the large elements created by the previous step. An example of this tangling is illustrated by the change from the initial mesh in Figure 27a to the tangled mesh in Figure 30b.

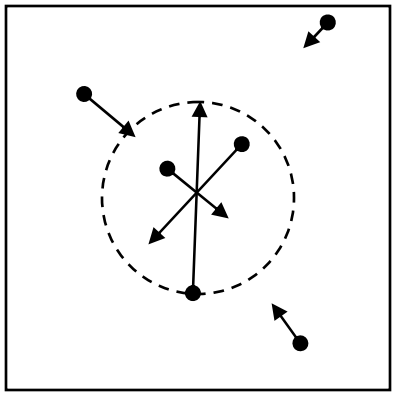

Figure 26: Geometric representation of how the mesh is tangled; circles are the initial position of a node and arrows are the final position. Nodes within the circle are mirrored about the center while the remaining nodes are attracted towards the center an amount based on their distance from the center.

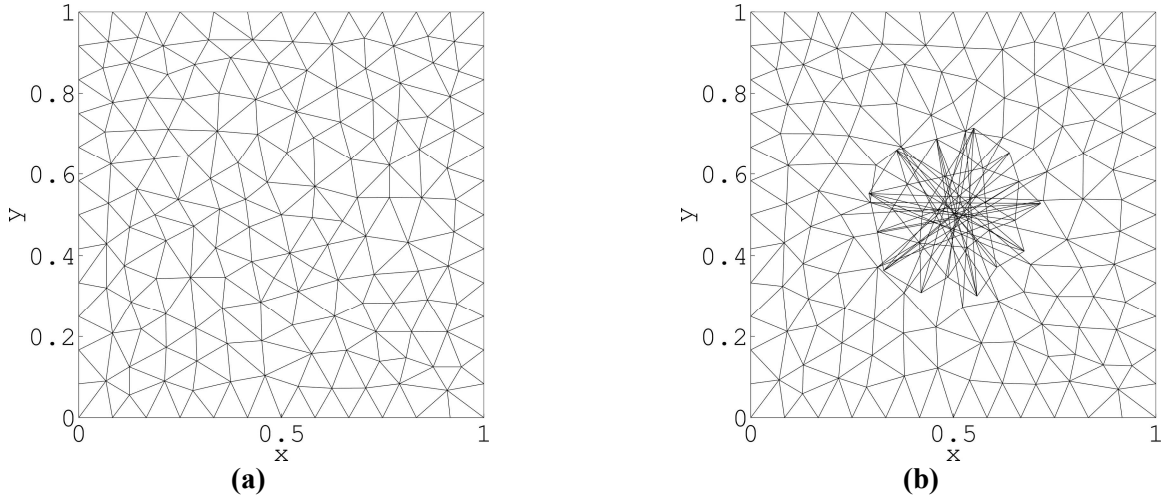

Figure 27: (a) Initial mesh and (b) tangled mesh used in convergence study.

Progressively finer meshes are used and the error in strain energy is computed for each case. In addition, this test is performed for both linear and quadratic shape functions. The results of the error in strain energy as a function of the number of degrees of freedom for all four cases are shown in Figure 28.

In Table 6 the convergence rates of all four cases are listed. Observe that the rate of convergence for the tangled mesh is equivalent to the non-tangled case for both linear and quadratic shape functions.

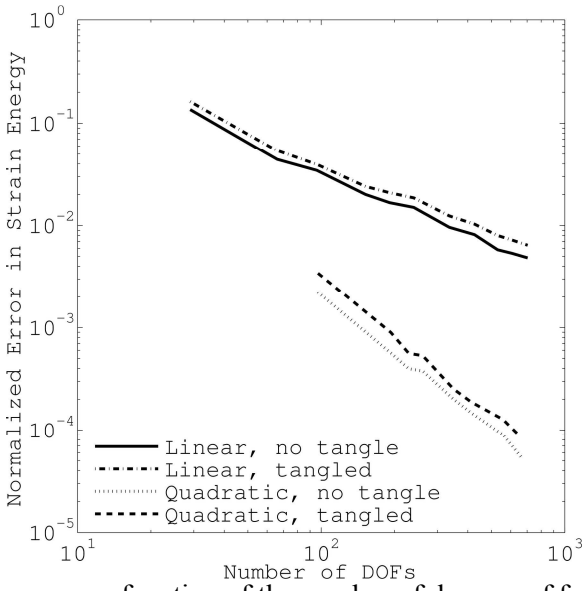

Figure 28: Error in strain energy as a function of the number of degrees of freedom for a non-tangled and tangled mesh with linear and quadratic shape functions.

Table 6: Convergence rates

Shape	Mesh	Convergence Rate
Linear	Not Tangled	1.02
Linear	Tangled	0.97
Quadratic	Not Tangled	1.88
Quadratic	Tangled	1.92

4.4 Rank Deficient Stiffness Matrix

In all of the tests performed in the previous Sections the correct solutions (to machine precision) were recovered, and the solutions were found to be unique. However, unlike in classic finite element analysis, uniqueness is not guaranteed for a tangled mesh. To motivate this, consider using linear shape functions over the specially constructed 1-D mesh defined by nodes (alternatively, see Figure 29)

$$\left\{0, 2/3, 1/3, 2/3, 1\right\} \tag{4.3}$$

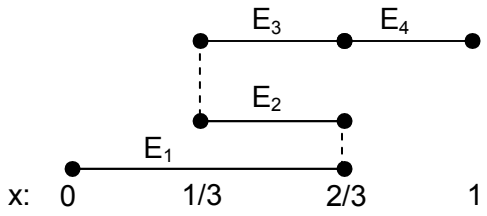

Figure 29: 1-D mesh specially crafted to cause rank deficiency in the stiffness matrix. Solid lines are elements, while dotted lines between nodes indicate the nodes are one and the same.

Observe that elements 2 and 3 lay exactly on top each other, with one positively oriented and the other negatively oriented. Now consider what this does to the 3rd nodal shape function. First, the cell decomposition for this shape function contains just one cell over the region $x = \left[\frac{1}{3}, \frac{2}{3} \right]$ with an index of $I = \{2,3\}$. Next, the two classic finite element shape functions attached to this node are

$$\begin{aligned} N_2^{(3)}(x) &= 2 - 3x \\ N_3^{(3)}(x) &= 2 - 3x \end{aligned} \tag{4.4}$$

The cell shape function is therefore

$$S(x) = \sum_{k=\{2,3\}} \Theta_k N_k^{(3)}(x) = N_2^{(3)}(x) - N_3^{(3)}(x) = 0 \tag{4.5}$$

In other words, no matter what value is assigned to $\hat{u}_3$ the contribution to the overall function $u(x)$ is zero – it is as though it is a completely superfluous node that is not connected to the mesh in any way.

As one would expect, this is cause for consternation. For example, if one attempts to solve the Poisson problem in Equation (2.1) with $f = 1$ by way of the proposed methodology, the resulting stiffness matrix (this corresponds to nodes 2,3,4 since $\hat{u}_1 = 0$ and $\hat{u}_5 = 0$)

$$K = \frac{3}{2} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 4 \end{bmatrix} \tag{4.6}$$

is rank deficient; moreover the deficiency corresponds to node 3. A singular stiffness matrix such as this will cause problems with typical linear solvers [3], [61]. Note, however, that if a pseudo-inverse, i.e.

$$pinv(K) = \frac{2}{9}\begin{bmatrix} 4 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{4.7}$$

is used to solve the system, the result is

$$\hat{u} = \left\{0, 2/9, 0, 1/9, 0\right\} \tag{4.8}$$

The complete approximate function $u(x)$ is shown in Figure 30. Observe that the degrees of freedom are not necessarily the field values at the nodes, i.e. the oriented linear combination must be evaluated to obtain field values.

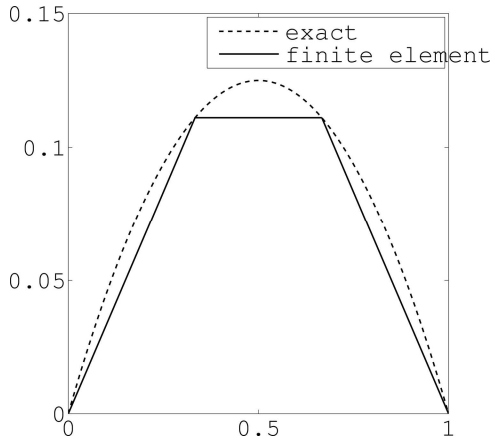

Figure 30: Solution to 1-D Poisson problem.

The exact solution to this Poisson problem is $u(x) = 1/2\, x(1-x)$ and it is easily verified that the nodal values are exact. Further, despite the mesh being tangled the solution $u(x)$ is identical to that of an untangled mesh with nodes

$$\left\{0, 1/3, 2/3, 1\right\} \tag{4.9}$$

The above rank deficiency is, of course, due to the exact placement of nodal coordinates. If the 4th node is moved a slight amount, say ε, the stiffness matrix is no longer rank deficient. Unfortunately, while the matrix is not rank deficient, i.e. there exists a unique solution, the matrix is nearly singular (condition number is $O(\frac{1}{\varepsilon})$) and therefore still problematic for linear solvers.

Even in the case of near singularity the above scenario is trivial to detect in a computationally efficient manner by way of an L^2-norm of the nodal shape functions. If for some nodal shape function this norm is much less than the norm of the adjacent nodes then the stiffness matrix will be nearly singular. Essentially, this means that the nodal shape function is poorly scaled.

The above scenario is not the only way in which the matrix can become (nearly) singular. It is also possible that, while every nodal shape function has a well-valued L^2-norm, some of them are linearly dependent. For instance, the 1-D mesh given by nodes

$$\left\{0, 3/4, 5/8, 1/2, 1/4, 3/8, 1/2, 5/8, 3/4, 1\right\} \tag{4.10}$$

does not have any nodal shape functions of the kind described above. It does, however, have a repeated shape function. Node 3 and node 8 have as shape functions the classic linear hat function that is zero at $x = 1/2$, one at $x = 5/8$, and zero at $x = 3/4$.

The linear dependence amongst nodal shape functions can certainly be more subtle, involving a vast number of degrees of freedom, than what was illustrated in the above examples. However, in all cases the matrix singularity is identical to what is known to occur in s-FEM [56]. But "*it is rare to come across such a problem*" as pointed out by the authors of [62]. *That is, singularity is not expected to occur in a tangled mesh unless the mesh is specially crafted to create such a problem.* Rarity notwithstanding, rank deficiency is possible and the exact cause is currently unknown; future work will concentrate on identifying the cause and providing a means to rectify the singularity.

As previously stated, the stiffness matrices for all examples in the previous Sections are well-conditioned (similar in value to their untangled counterpart), i.e. they are not singular. Furthermore, unless explicitly stated all subsequent examples also have well-conditioned stiffness matrices.

Chapter 5: Application 1 – Mesh Morphing

In this Section the proposed methodology is used to overcome the challenges of tangled meshes when mesh morphing is performed. Typical mesh morphing techniques include simplex-linear, simplex-natural neighbor, weighted residual, FEMWARP, and LBWARP [33]. The examples that follow use either (1) a naïve morphing in which an analytic function it prescribed to define the morph, or (2) the simplex-linear morphing technique [33], [63], [64]. For a discussion on how the simplex-linear method works see Appendix A. Observe, however, that the focus in on tangled meshes and not the specific morphing technique employed. Moreover, the finite element framework discussed in this paper is equally valid for other morphing techniques as well.

5.1 Simplistic Mesh Morphing

As a first example of mesh morphing and a final example of the difference between classic FEA and the proposed theory, consider performing a design change on the ubiquitous 'plate with a hole' problem, see Figure 31. In particular, consider the family of cases where a mesh is generated for a radius of $R_0 = 0.10$, and subsequently morphed to a hole radius of up to $1.8R_0$.

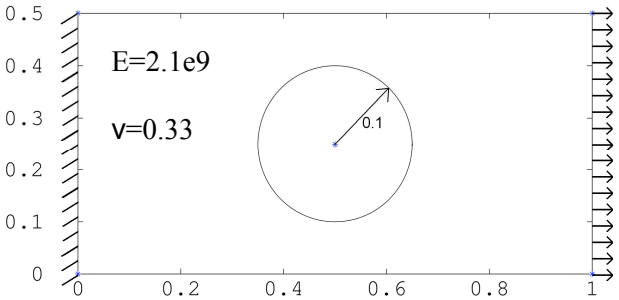

Figure 31: Plate with a hole problem with left-hand boundary clamped and a uniform pressure applied to the right-hand boundary.

A simple implementation of mesh morphing is as follows. Consider any node at a distance of r from the center of the hole. As the hole is expanded from a radius of R_0 to a radius $R \le 1.8R_0$, nodes surrounding the hole are pushed outwards to a new location at a radius of r' according following equation:

$$r' = \begin{cases} r + R - R_0; & r \le \dfrac{(5R - 3R_0)}{8} \\ \dfrac{(9R - R_0)}{4} - r; & \dfrac{(5R - 3R_0)}{8} < r \le \dfrac{(9R - R_0)}{8} \end{cases} \tag{5.1}$$

Effectively, this morphing scheme achieves the following objective:

1. All elements whose nodes lie between R and $(13R - 11R_0)/8$ remain positively oriented.
2. All elements whose nodes lie between $(13R - 11R_0)/8$ and $(9R - R_0)/8$ flip orientation.
3. Elements and nodes outside $(9R - R_0)/8$ remain unchanged.

For instance, the tangled mesh for $R = 0.18$ is illustrated by Figure 32, with a close-up of the top of the hole shown in Figure 33. The purpose of using this *ad hoc* morphing is to showcase the capabilities of the proposed methodology.

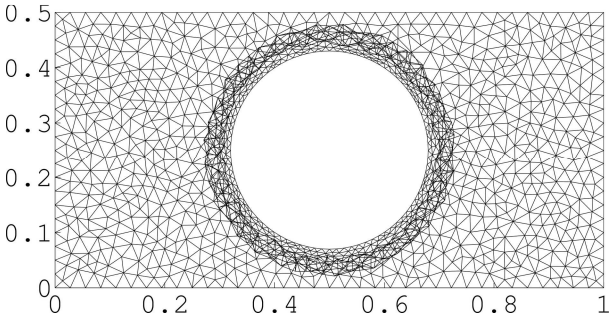

Figure 32: The resulting tangled mesh as a consequence of increasing the radius of the hole from 0.1 to 0.18.

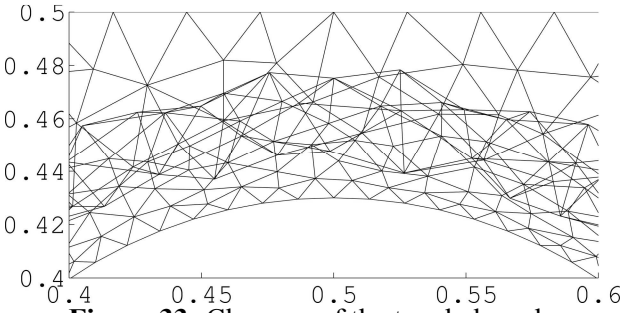

Figure 33: Close-up of the tangled mesh.

The von Mises stress at the top and bottom of the hole is queried for solutions obtained via classic FEA and the proposed methodology for the tangled mesh for various radius scaling $1 \le R/R_0 \le 1.8$. These quantities are compared against those obtained from a remeshed (not tangled) geometry. The stresses of the remeshed solution are shown in Figure 34, while classic FEA over the tangled mesh is shown in Figure 35, and finally the proposed methodology over the tangled mesh in Figure 36. Observe that classic FEA leads to erroneous results. In fact, not only do the values differ significantly from the remeshed solution, but classic FEA solution is not even symmetric. The proposed method, on the other hand, yields results in strong agreement with the remeshed solution; the two differ by at most 1.5% and is

most likely due to the fact that two different discretizations (meshes) are used, so two different approximate solutions are found.

It should be evident by now that classic FEA over tangled meshes yield erroneous solutions, and thus, this approach will not be pursued in any further examples.

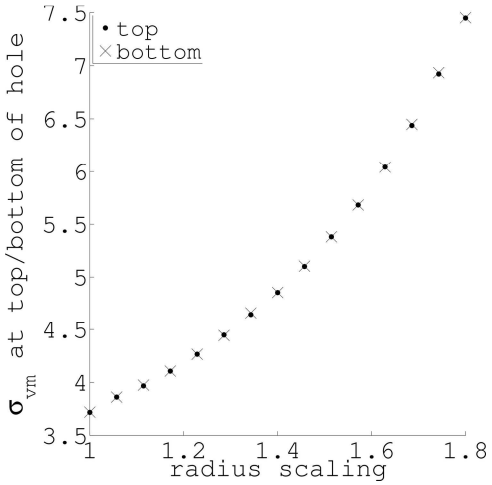

Figure 34: Von Mises stress at top and bottom of the hole for various radii when remeshing is used.

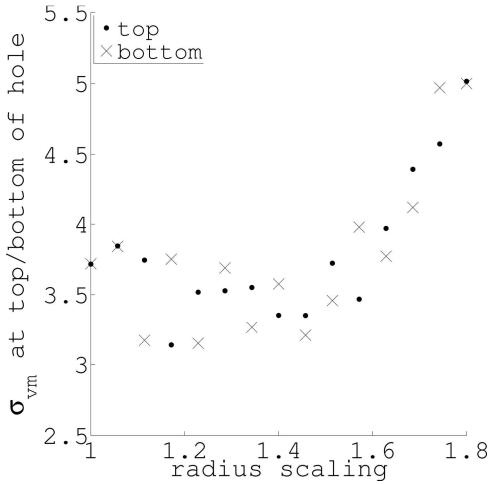

Figure 35: Von Mises stress at top and bottom of the hole for various radii when classic FEA is used over tangled mesh.

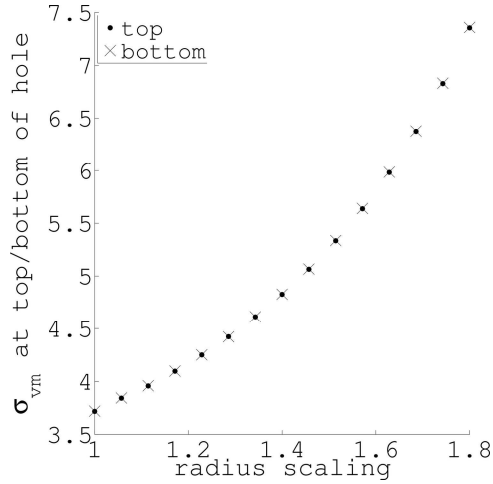

Figure 36: Von Mises stress at top and bottom of the hole for various radii when the proposed methodology is used over tangled mesh.

5.2 Morphing a Plate with 3 Holes

Next, consider the problem illustrated in Figure 37. The center hole starts centered about y=0.15 where an initial mesh is constructed, see Figure 38, and translated upwards, to a final value of y=0.35. The initial mesh is morphed at each instance, using the simplex-linear technique, to reflect the design change.

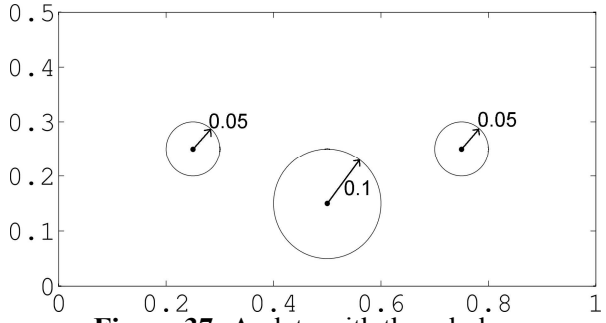

Figure 37: A plate with three holes.

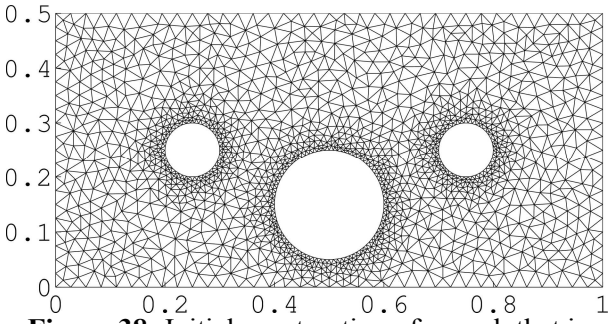

Figure 38: Initial construction of a mesh that is subsequently morphed.

An example of a morphed mesh is shown in Figure 39. Note the 'poor quality' of the resulting mesh (in addition to tangling). To improve mesh quality, an edge flipping technique (discussed in Appendix B) is used; this results in the mesh illustrated by Figure 40.

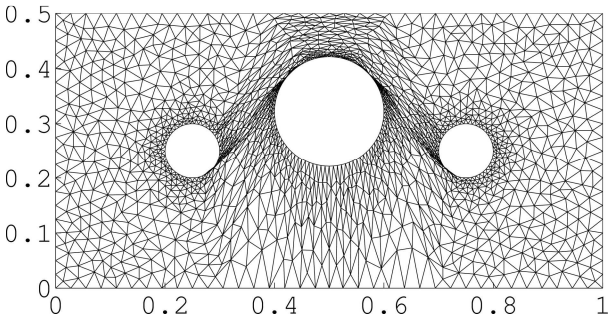

Figure 39: A morphed mesh with a poor function space quality.

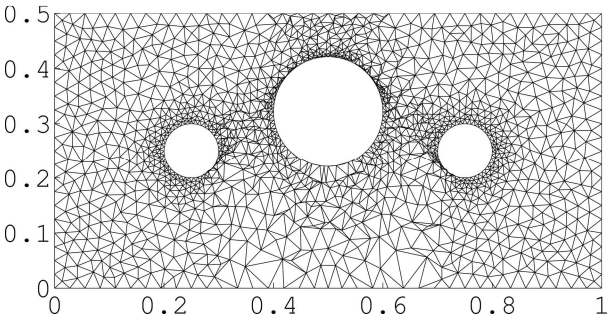

Figure 40: Morphed mesh that has undergone edge flipping to improve the function space.

In Figure 41, the number of overlaps (due to tangling) is summarized after: (1) morphing and (2) after quality improvement. Observe that edge flipping can increase or decrease the number of overlaps.

To illustrate the effectiveness of the proposed method a remeshed (not tangled) geometry is used for comparison. The specific quantity of interest is the maximum von Mises stress. Figure 42 shows that the proposed methodology over the tangled mesh is consistent with a full remesh. Both are approximations of

the (unknown) exact field, and they differ by at most 1.4% (Note that due to tangling, the functional space can increase, and therefore, it is possible that the morphed solution is more accurate than the remeshed solution in some instances.)

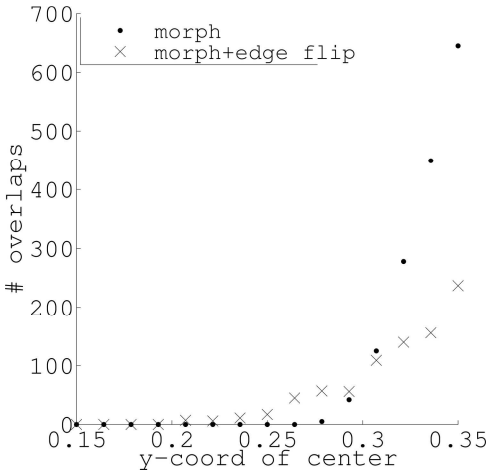

Figure 41: Number of overlaps in the mesh of the plate with three holes problem after morphing and quality improvement.

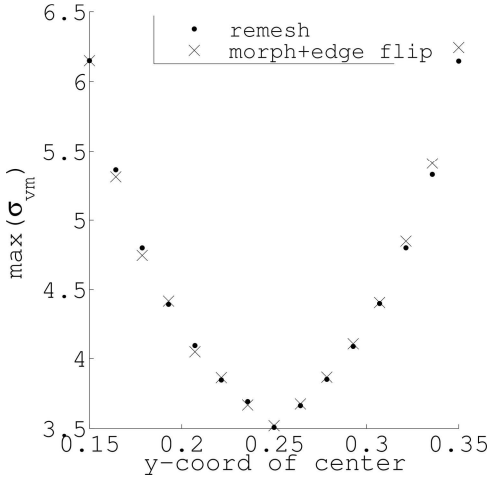

Figure 42: Maximum von Mises stress in various morphed configuration for an untangled (remeshed) mesh and the morphed mesh.

5.3 Simultaneously Moving and Growing a Hole

In this example, the plate with a single hole geometry is again used. The design modification being sought is a combination of moving the hole to the right and increasing the radius of the hole. Specifically, as a parameter t grows from 0.0 to 1.0, the x-coordinate of the center of the hole changes from 0.45 to 0.8 and the radius changes from 0.08 to 0.15.

The initial mesh is shown in Figure 43, while the morphed mesh for t=1.0 is shown in Figure 44. As before, the quality of the mesh becomes poor as the mesh is morphed; thus, the edge flipping technique is employed, yielding the mesh shown in Figure 45.

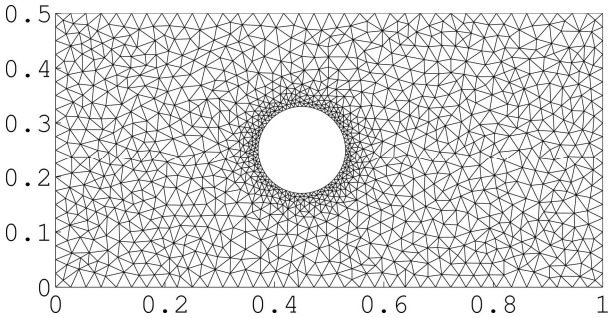

Figure 43: Initial construction of a mesh that is subsequently morphed.

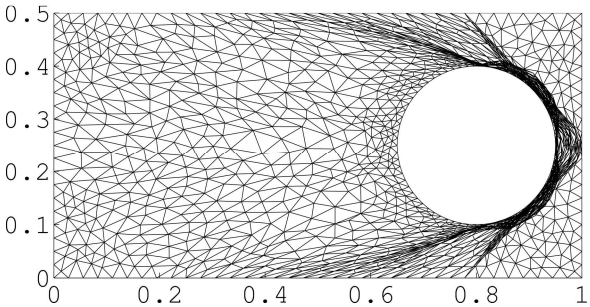

Figure 44: The morphed, unoptimized mesh resulting from shifting the hole to the right by 0.35 and scaling the radius by 1.875.

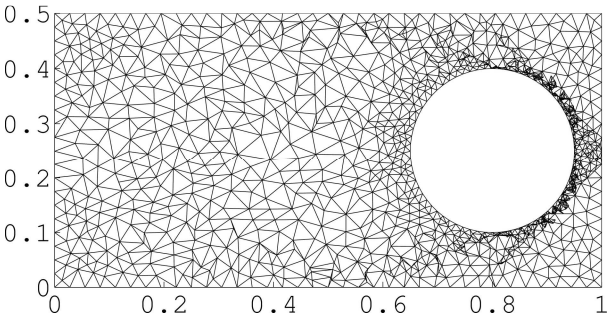

Figure 45: Morphed mesh optimized through edge flipping.

As with the previous example the number of overlapping regions after morphing and after quality improvement is summarized in Figure 46. The number of overlapping regions in the final configuration is approximately 0.75 times the number of triangles in the initial mesh (2388 triangles). This means that the assembly cost is nearly 1.75 times that of an untangled mesh; however, the linear system is still sparse, e.g. 99.8% sparse in the untangled case versus 99.63% sparse in the tangled case.

Finally, the maximum von Mises stress is compared against a remesh. The results illustrated in Figure 47 once again demonstrate the effectiveness of the proposed method for handling tangled meshes in FEA.

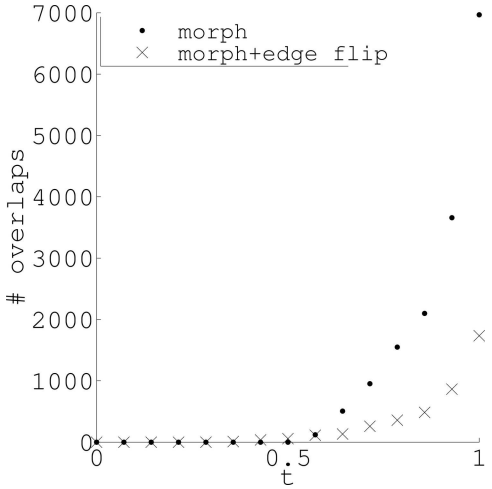

Figure 46: Number of overlaps in the mesh of the plate with a single hole problem after morphing and quality improvement.

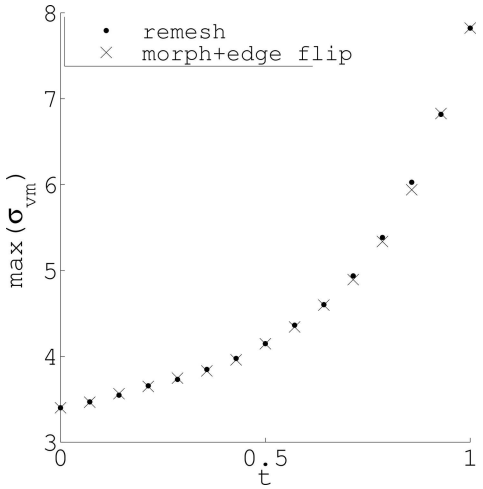

Figure 47: Maximum von Mises stress in various morphed configuration for an untangled (remeshed) mesh and the morphed mesh.

5.4 3-D Morphing

In this final example, 3-D mesh morphing is considered. The bearing block shown in Figure 48 is morphed so as to increase the diameter of the bearing surface (center hole) from 63.5 mm to 72 mm. Also shown in the figure are the boundary conditions; two of the mounting holes are fixed, while the third mounting hole has a force in the y and z direction.

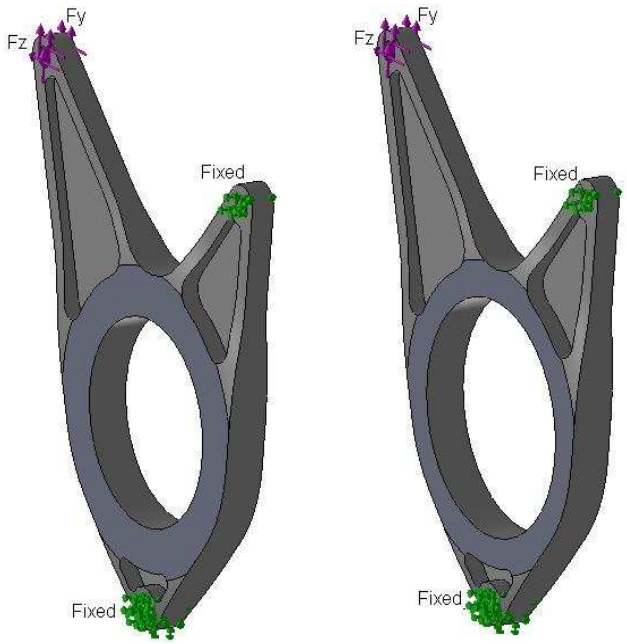

Figure 48: (a) Initial bearing block design with a hole diameter of 63.5 mm and (b) final design with a hole diameter of 72 mm.

The technique used to morph the mesh is similar to that used earlier in 2-D example where an explicit equation was used to morph the mesh. The initial untangled mesh is shown in Figure 49. After morphing, the overlapping regions of the tangled mesh are illustrated in Figure 50.

Figure 49: Initial mesh that is subsequently morphed and tangled.

Figure 50: Overlapping regions (dark portions) of the tangled mesh.

The quantity of interest being sought is the maximum total displacement. Table 7 lists this quantity for the initial configuration, the morphed, tangled configuration, and a remeshed configuration (all quantities are in mm). Not only is the trend correct, increasing the diameter of the hole increases the displacement, but the displacement of the tangled mesh closely matches that of the remeshed solution.

Table 7: Comparison of maximum total displacement

	Initial Configuration	Final Configuration	
		Morphed	Remesh
Maximum Total Displacement	2.1230e-2	2.5392e-2	2.5371e-2

Chapter 6: Application 2 – Geometric Quality Improvement Through Local Tangling

While the previous application addressed tangled meshes as an unavoidable consequence of mesh morphing, the application presented here intentionally tangles the mesh as a means to improve the geometry quality of an element. Here, geometric quality refers to the measure of how much an element deviates from an equilateral element; for some standard definition see [3], [6–10]. Such poor quality elements, when present in a finite element mesh, can lead to ill-conditioning of the underlying linear system. This potentially results in degradation of the solution, and slow convergence in iterative solvers [3]. Further, as the quality of a finite element deteriorates, the function space spanned by the element also deteriorates, and the element cannot accurately represent the underlying field [3], [6]. This is particularly evident in regions of high gradients, e.g. stress concentrations in solid mechanics.

While quality optimizers abound [65–68], each has thus far been constrained to improvements that do not tangle the mesh. As tangled meshes are appropriate they will be exploited by a technique called *element covers* – this can be seen as an extension, or alternative, to current mesh quality optimizers. Here, the principle idea is to locally tangle the mesh by covering (replacing) a poor quality element with a topologically valid collection of better quality elements. Specifically, node movement is used to generate a tangled mesh in which all elements – both positively and negatively oriented – have a guaranteed improvement in quality.

As an example, consider the poor quality element in Figure 51a. Further consider splitting the triangle by adding a center node (Figure 51a), and relocating the node *outside* the base triangle (Figure 51b). In performing this operation, the local mesh has been tangled; the resulting positively oriented elements are shown in Figure 51c, while the negatively oriented element is shown in Figure 51d. Observe that by removing the constraint that all triangles must be positively oriented the quality of the base triangle is easily improved.

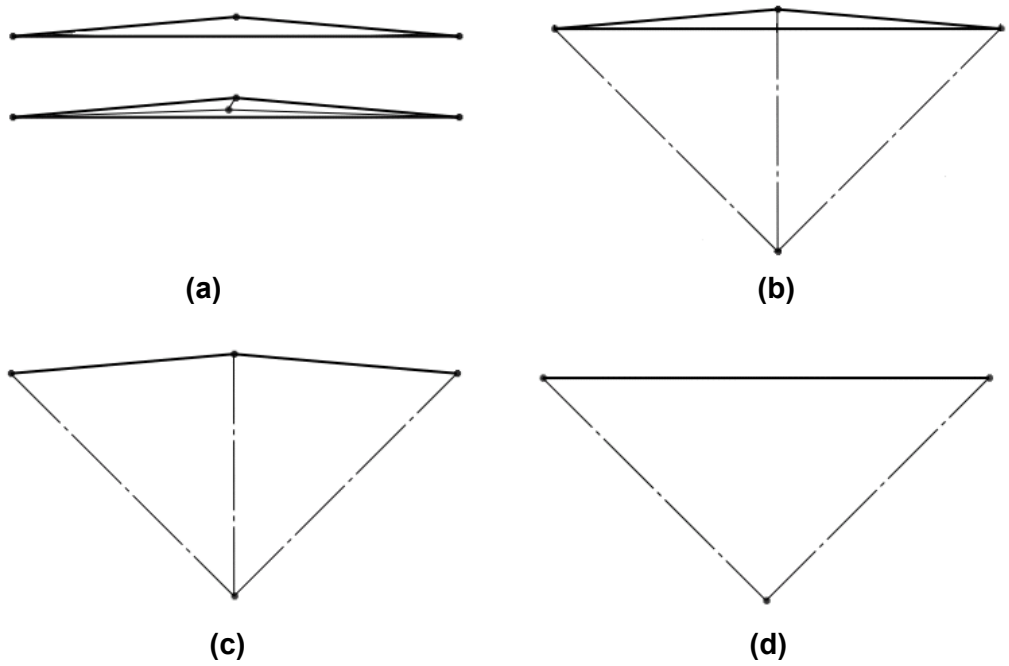

Figure 51: (a) Splitting of poor quality base triangle and (b) relocation of added node to outside the base triangle. (c) Positively oriented triangles and (d) negatively oriented triangle after the relocation.

6.1 Algorithm

Exploiting the above concept, I propose here an algorithm that is guaranteed to improve the quality of 2-D triangular meshes through local tangling. The essence of the cover algorithm is captured in Figure 52. The central concept is to split a poor quality base triangle and relocate the added node to a 'better' location which depends on the properties, or Class, of the base triangle. Further, the newly added triangles will always be of an equal or lower Class and the splitting is done recursively until all triangles are of sufficient quality. A detailed description of each Class is now provided.

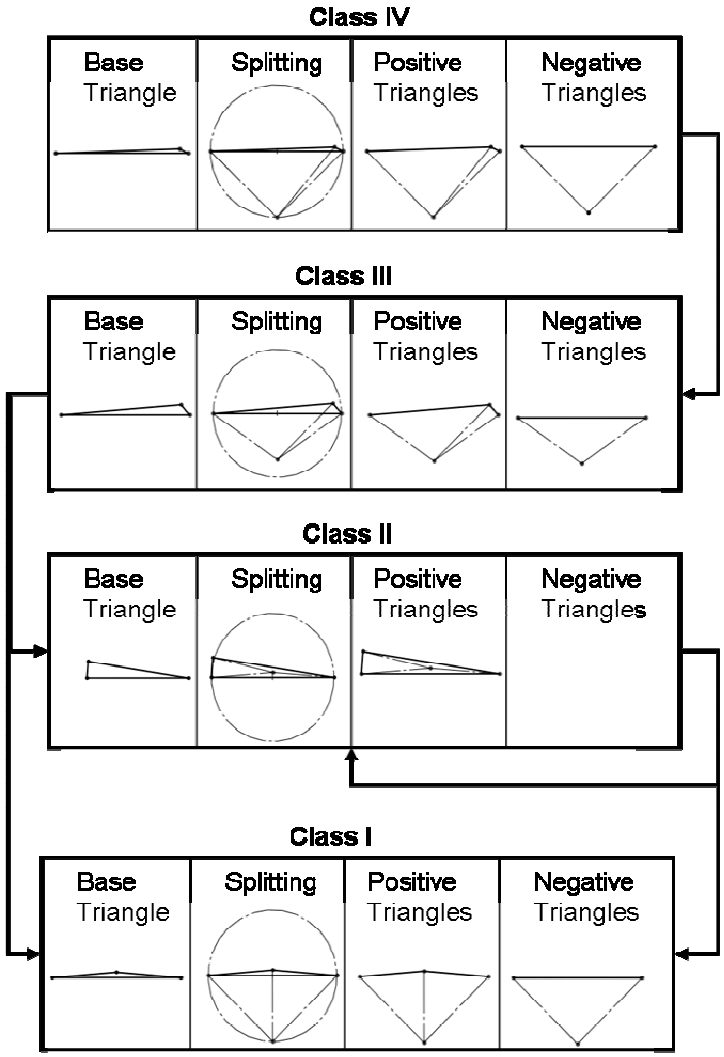

Figure 52: Flowchart of element cover algorithm, showing (1) poor quality base triangle, (2) relocation of added node, (3) the new positively oriented triangles, and (4) the new negatively oriented triangles for each Class. The arrows show what Class the newly added triangles are.

6.1.1 Class IV – General Triangle with Far Circumcenter

The most generic of all Classes, these triangles have a circumcenter that lies 'far' from the base triangle – perhaps even at infinity. In this algorithm 'far' is considered to be anything outside the smallest enclosing circle. When this case occurs, the point is moved to the location on the smallest enclosing circle that is farthest from the base triangle, see Figure 52. In using this point as the 'better' location the 3 newly added triangles will either be of sufficient quality or a lower Class. Recursive splitting is required for the latter case.

6.1.2 Class III – General Triangle with Close Circumcenter

The differentiating factor between this Class and the previous Class is that here the circumcenter lies within the smallest enclosing circle. In using the circumcenter as the 'better' location, the newly added triangles are necessarily isosceles – Class II or Class I – and recursive splitting is required if they are not of sufficient quality.

6.1.3 Class II – Tall Isosceles

Tall isosceles triangles, which always have a circumcenter that lies within the base triangle, are split accordingly. This split results in two short isosceles triangles (Class I) and one tall isosceles triangle (Class II) which has the same base edge as the base triangle but a guaranteed smaller height, see Figure 52. Indeed, this new tall isosceles triangle is of higher quality as it is closer to equilateral than the base triangle. In the case that the new tall isosceles triangle is not of sufficient quality it is recursively split.

6.1.4 Class I – Short Isosceles

For short isosceles triangles, the point on the smallest enclosing circle that is farthest from the base triangle is used as the 'better' location. This yields three triangles that are never split further as they are necessarily of sufficient quality.

6.1.5 Summary

In summary, we see that recursively splitting any poor quality triangle always results in Class I and Class II triangles or triangles of sufficient quality. Moreover, it can be shown that all triangles produced by this algorithm have a guaranteed minimum quality of 0.866 (see Appendix C), and the runtime of this algorithm is logarithmic with respect to quality improvement (see Appendix D). For example, to improve the quality of a Class II triangle (the hardest of all Classes as it requires recursive splitting) from 10^{-6} to 10^{-2} take 30 splits and from 10^{-6} 10^{-1} takes 48 splits.

6.2 Additional Remarks

While the above algorithm has useful properties in the form of quality and complexity bounds, there are a few additional properties that must be highlighted.

1. Since the node placement decision does not account for neighboring elements it is possible that a node is moved within a close proximity of another node. This, along with other possible geometric states, can lead to a linear dependence of the nodal shape functions. If this occurs, the smallest eigenvalue of the stiffness matrix will tend towards zero, making the condition number worse. This scenario was addressed in Section 4.3.
2. The only modification to the original mesh data structures is the removal of triangles, addition of triangles, and addition of points. Furthermore, the original points are never moved.
3. The nodes/elements introduced by the covers also add local resolution to field. This also means that the linear system is both larger (degrees of freedom are added) and less sparse (coupling from overlapping element). Fortunately, both of these are proportional to the number of nodes added and the amount of tangling that occurs.

6.3 Examples

The usefulness of the element covers algorithm put forth in the previous Section is now demonstrated through a series of examples. Recall the intent of improving the geometric quality of an element is to fix ill-conditioning of the stiffness matrix due to poor quality. Thus, before showcasing the covers algorithm the effects of poor quality on the condition number are studied. As modern meshers employ a number of quality improvement steps (that currently only consider improvements which don't involve tangling) after initial mesh generation, high quality triangle meshes are typical. Artificial quality deterioration is therefore employed.

To this end, consider the plate with a hole problem from Section 5.1 (see Figure 31) with associate mesh shown in Figure 53. The highlighted node in Figure 54 is moved along the specified trajectory towards the highlighted edge inducing a deteriorated element quality. The relationship between minimum quality and stiffness matrix condition number along the trajectory is illustrated by Figure 55; linear shape functions are used. Observe that as quality decreases, the largest eigenvalue of the stiffness matrix increases. Consequently, the condition number, defined as the ratio of largest to smallest magnitude eigenvalue, also increases.

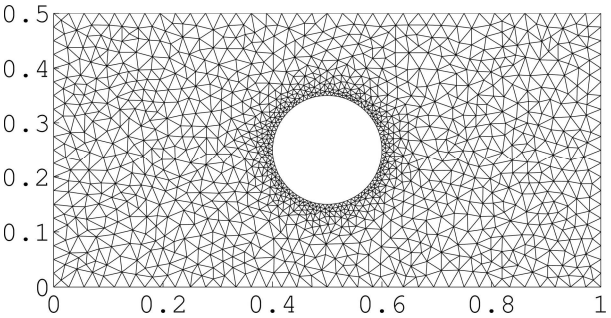

Figure 53: Mesh of the plate with a hole geometry.

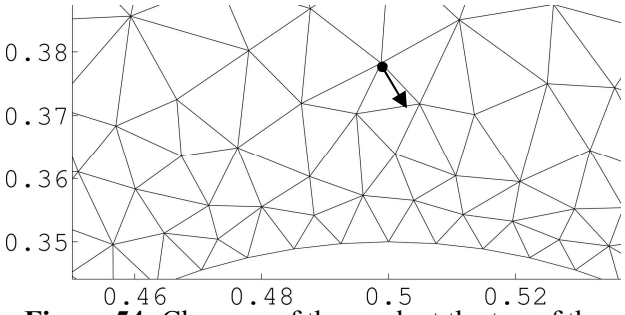

Figure 54: Close-up of the mesh at the top of the hole and the trajectory of the node for creating a poor quality element.

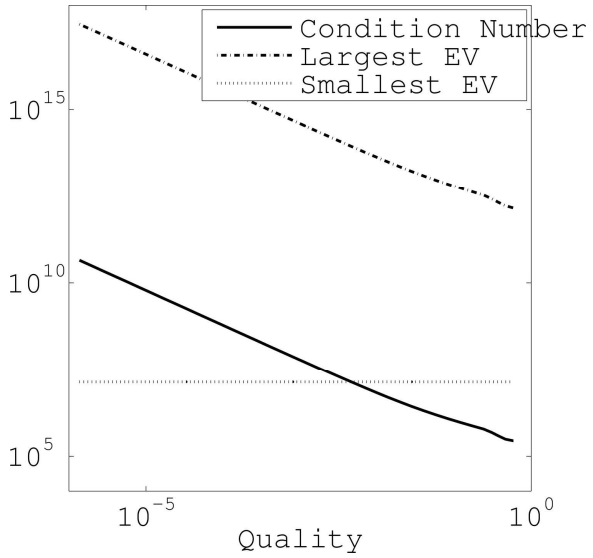

Figure 55: Impact on condition number of the stiffness matrix due to poor element quality.

The element cover algorithm is now used to recover the minimum element quality. While the algorithm does provide a means to guarantee a minimum quality of 0.866, this level of quality is unnecessary. Thus, the algorithm is instead run until element qualities are at least 0.2. An example covering to poor quality element is shown in Figure 56; this corresponds to a starting quality of 1.37e-6 and a covered quality of 0.368.

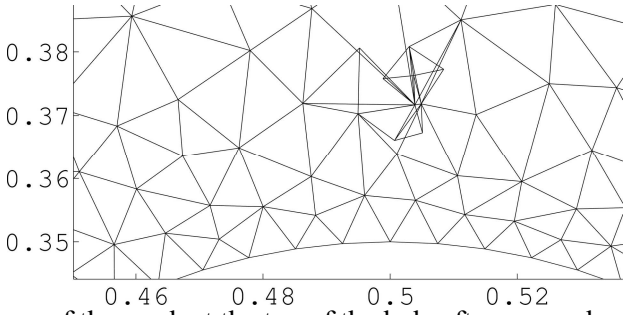

Figure 56: Close-up of the mesh at the top of the hole after covers have been generated.

Figure 57 shows the original deteriorated quality (before covers are applied) along the trajectory as well as the minimum quality after element covers are generated. Observe that the covers are indeed able to improve the geometric quality to the requested level of at least 0.2. In Figure 58 the largest magnitude eigenvalue with and without element covers is presented. Observe that when covers are used, the largest magnitude eigenvalue is stable; i.e. it does not increase unbounded as before.

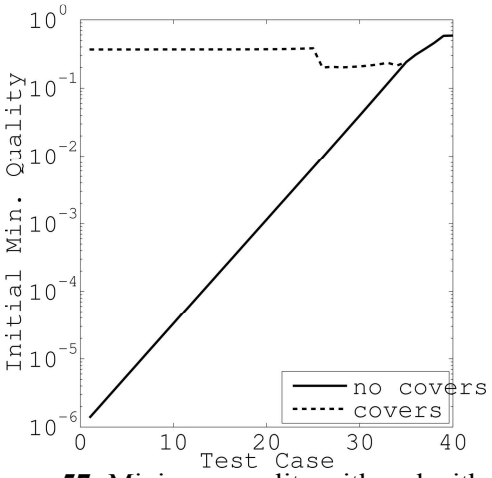

Figure 57: Minimum quality with and without element covers along the trajectory.

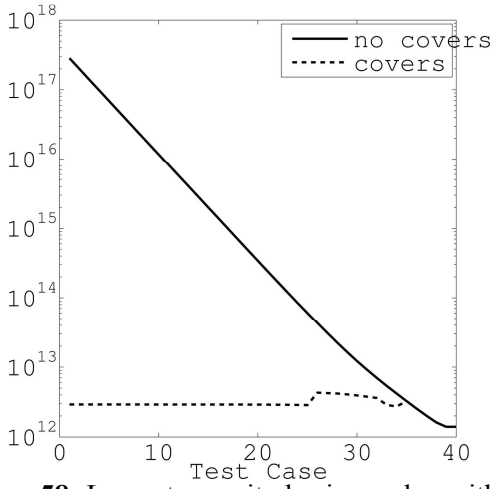

Figure 58: Largest magnitude eigenvalue with and without element covers along the trajectory.

While it is clear that element covers can be used to overcome poor geometric quality and prevent the largest magnitude eigenvalue from becoming overly large, the smallest magnitude eigenvalue can become problematic by approaching zero. This is shown in Figure 59 where applying covers to an element whose quality is worse that 1e-4 results in a diminishing smallest eigenvalue. As the condition number is the ratio of largest eigenvalue to smallest eigenvalue, the overall condition number is not much improved.

Despite the matrix becoming singular, the resulting displacement field is not erroneous; stress predictions are comparable to that of a good quality untangled mesh, for example.

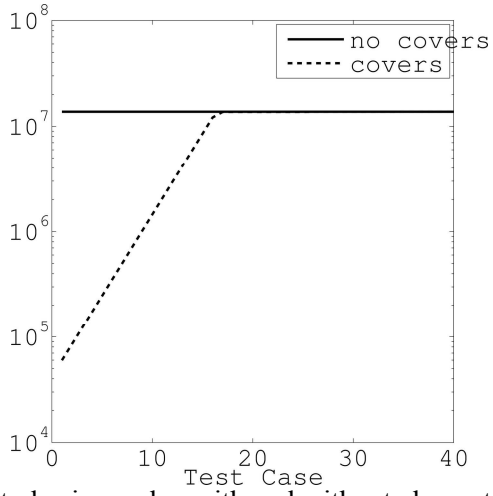

Figure 59: Smallest magnitude eigenvalue with and without element covers along the trajectory.

The reason for the diminished eigenvalue is the cover algorithm tends to place nodes in geometric configurations that lead to poorly scaled/nearly linearly dependent shape functions. This is precisely what was addressed in Section 4.3 when discussing the possibility of near-singular stiffness matrices when tangled meshes are used. Indeed, it may even be possible to overcome this limitation of element covers by analyzing the geometric configuration and removing the null space from the stiffness matrix.

Regardless of this, it should be evident that element covers offer an entirely new direction for quality improvement algorithms by making use of a tangled mesh.

Chapter 7: Conclusion

Prior to this work the general consensus was that tangled meshes are not appropriate for use in the finite element method as they lead to physically invalid solutions. In this document an approach is taken that extends the mathematical framework behind the continuous Galerkin formulation of FEA so as to accommodate tangled meshes. Specifically, an oriented linear combination of the classic finite element shape functions is used to construct a new class of nodal shape functions that are mathematically proven to lie within the correct function space, namely $H_0^1(\Omega)$, independent of the state of tangling. It is also shown that the proposed methodology exactly recovers classic FEA when the mesh is not tangled.

Starting from these new nodal shape functions, a detailed derivation of the finite element method over tangled simplicial and non-simplicial meshes is provided for elliptic problems. This derivation is also supported by a thorough discussion of the implementation of the proposed method. Upon analyzing the implementation it is clear that the costs associated with handling a tangled mesh are a strong function of tangling – this includes costs in the form of (1) detecting the overlapping regions, (2) assembling the stiffness matrix, and (3) fill in (how much sparsity is lost) of the stiffness matrix. In other words, if the amount of tangling is low compared to the total number of nodes/element in the mesh then the total overhead of assembling and solving a finite element system is equally low. Exact numbers were not gathered. Instead, algorithmic efficiency, in terms of runtime and memory usage, is studied. This does not, however, deny the fact that in some scenarios it may be better to untangle the mesh.

The final testament to the effectiveness of the proposed methodology is illustrated by way of examples starting in Chapter 4. Specifically, numerous patch tests are performed with each one showing that classic finite element fails to provide the correct solution, whereas the proposed methodology recovers the exact solution to within machine precision.

This validation is then followed by a series of mesh morphing examples; a useful application that has seen recent attention from the meshing and finite element communities. Classically, a morphed mesh that has become tangled is either untangled or remeshed. Here it is shown that the tangled mesh can be used

directly while still providing accurate results – in the worst example there was a 1.5% difference between stress predictions arising from solving over a tangled mesh with the proposed methodology and a remeshed (untangled) geometry.

As another application, a local tangling of a mesh, deemed element covers, is used to improve the geometric quality of an element. While the end goal of this algorithm is to improve the conditioning of the stiffness matrix, only a partial solution is currently provided. Specifically, by tangling the mesh in the manner described in this document the offending largest eigenvalue is fixed, though the resulting stiffness matrix may be near-singular. Nonetheless, solutions over this tangled mesh are comparable to solution of a good quality non-tangled mesh. While not currently a complete solution, element covers provide a new way to improve element quality that is only possible with a tangled mesh.

Based on the above it is evident that, as opposed to the current treatment, tangled meshes are not only appropriate for use in the finite element method, provided the proposed methodology is used, but also useful.

Chapter 8: Future Work

While this document lays out the groundwork necessary for properly handling tangled meshes in FEA, there are a number of related research topics that have not been addressed. Below are some of these topics, split into theoretical, implementation, and application contributions, that warrant further investigation.

8.1 Theoretical

Patch Tests for Tangled Meshes:

As noted in Chapter 4, a true patch test does not exist for tangled meshes. Further, while the modified patch tests of Chapter 4 are necessary, these tests are not sufficient to prove that a finite element implementation over tangled meshes is correct. Such a test should ensure that [3] (1) as element sizes reduce to zero, the finite element approximation converges to the exact differential equation, and (2) the solution to the linear system of equations should be stable (no spurious changes in the solution) in regards to nodal positions. Since it is already known that the stiffness matrix can become rank deficient (under certain pathological conditions of a tangled mesh), the stability requirement must be modified accordingly.

Exact Function Representation:

It was observed that when a field was within the span of the element shape functions, the nodal values were precisely the field values; i.e. after evaluating the oriented linear combination at a node, the field value was the same as the nodal value. For example, if linear elements are used and the field is linear, then the nodal values are the same as the field values at the nodes. This trend was observed with 1-, 2-, and 3-D linear and quadratic simplicial elements. While only a conjecture at this point, it may be possible to prove that exact function representations imply nodal values are the same as field values; such a proof could be useful in developing the aforementioned patch tests.

Extension to Other Field Problems:

In this thesis, it was established that static linear problems of an elliptic kind can be solved via finite element over a tangled mesh. Further, while the properties of the proposed nodal shape functions are not dependent on the physics being solved, it is unclear whether these nodal shape functions can be used to be solve other field problems. For instance, it is known that in the current form (implementation) the stiffness matrix may contain negative eigenvalues. This poses a challenge for certain nonlinear problems such as buckling analysis where a negative eigenvalue is indicative of the buckling phenomena [69]. It may be possible to overcome this deficiency. For instance, the negative eigenvalues may be due to improperly scaled nodal shape functions. An example of this is a node whose connected elements are all negative, see Figure 60. The shape function for this node is the classic finite element hat function, but upside-down (due to the orientation); instead of the shape function attaining a value of one at the node, it attains a value of negative one. Such cases can be easily detected and the associated rows and columns of the stiffness matrix can be rescaled appropriately.

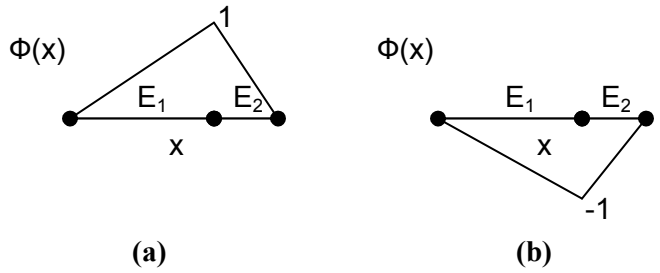

Figure 60: Nodal shape function when the elements attached to a node are (a) positively oriented and (b) negatively oriented.

When modeling structural mechanics with plasticity, the state of plasticity (whether plastic deformation has occurred and by how much) is tracked per element [70]. When a mesh becomes tangled this information will most likely need to be broadcast to the overlapping elements or perhaps the information should be tracked at a cell level.

Quality of a Tangled Mesh:

Quality metrics are used as an a priori indicator of whether a mesh can faithfully represent a good solution to finite element problems. As such, they measure quantities that affect the conditioning of the

stiffness matrix, e.g. element angles or aspect ratios, and the overall function space, e.g. edge lengths or element sizes. With the inclusion of tangled meshes, these metrics may not be representative any longer. For instance, a mesh with large elements is classically viewed as being inferior to one with small elements – as far as the function space is concerned. But when a mesh is tangled, large element can be equally as good as small elements. Take the case of the tangled 1-D mesh in Figure 61a. Despite all the elements having a length of approximately 0.5, the finite element solution over this tangled mesh is identical to one over the mesh in Figure 61b. In fact, the set of functions that can be represented is identical for both meshes.

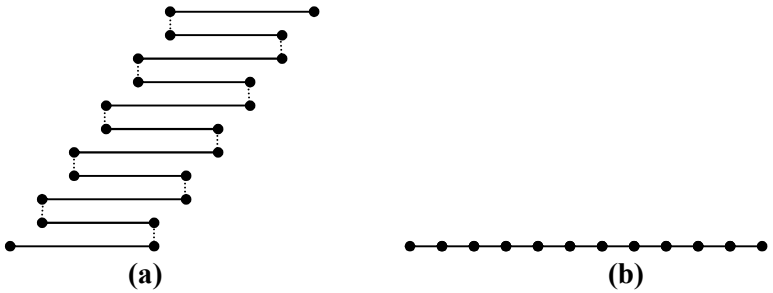

Figure 61: (a) A tangled mesh with large elements and (b) a non-tangled mesh with small element. Solid lines are elements, while dotted lines between nodes indicate the nodes are one and the same.

8.2 Implementations

Intersection Test:

The performance and accuracy of the element/element (or sub-element/sub-element) intersection query required to identify overlapping regions depends on how the elements (sub-elements) are represented. In addition, this query must support: (1) Checking if two element overlap (a simple yes/no), and (2) Representing the region of overlap so that numeric integration can be performed. Thus, optimal representation and supporting algorithms that facilitate this query must be developed.

Extension to other Elements:

While implementations for finite element analysis over simplicial meshes and quad meshes are directly addressed in this thesis, other elements, such as hexahedra, were not considered. In particular,

simplifications such as those used to split quad elements into sub-element must be identified. Without such simplifications FEA over tangled hexahedral meshes may be too costly.

8.3 Applications

Mesh Morphing with Tangling:

Mesh morphing techniques are judged by, amongst other requirements, how robust the method is to not producing tangled meshes [33]. But, as demonstrated, this requirement is unnecessary, and thus morphing techniques that are deemed 'poor' due to tangling may need to be reevaluated. Specifically, in [33] the authors show that the so-called smoothing technique for mesh morphing is computationally efficient, but it "*produces a severely tangled mesh for large transformations*" and "*is not a reliable morphing technique*."

Mesh Optimization with Tangling:

Current algorithms for improving element quality must be reevaluated in light of tangled meshes. The Laplacian smoother [15], an efficient yet powerful quality optimizer, unfortunately cannot handle tangled meshes. The reason is as follows: The primary step behind the Laplacian smoother is to move a node towards the center of all the nodes connected to it. For an untangled mesh this works well as seen in Figure 62. Now consider what happens when the same nodes are tangled as in Figure 63. Moving the center node is this manner has caused the triangle comprised of the center node, node 1, and node 2 to become nearly degenerate – even worse than the starting configuration.

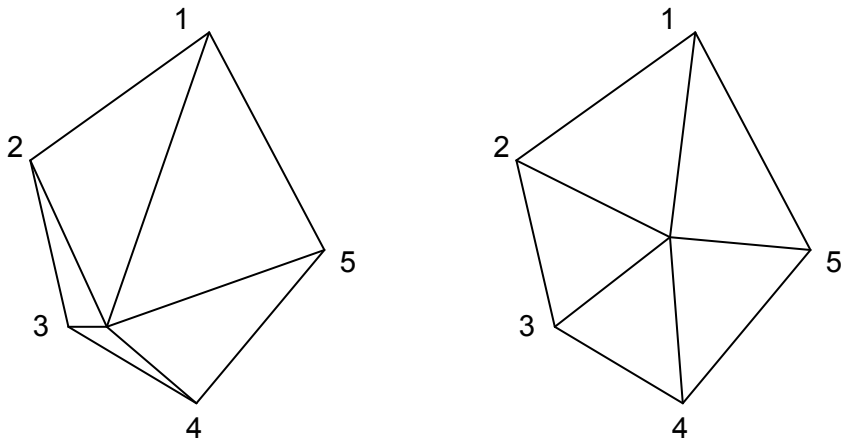

Figure 62: (left) A portion of a non-tangled, poor quality mesh that is (right) subsequently modified via Laplacian smoothing.

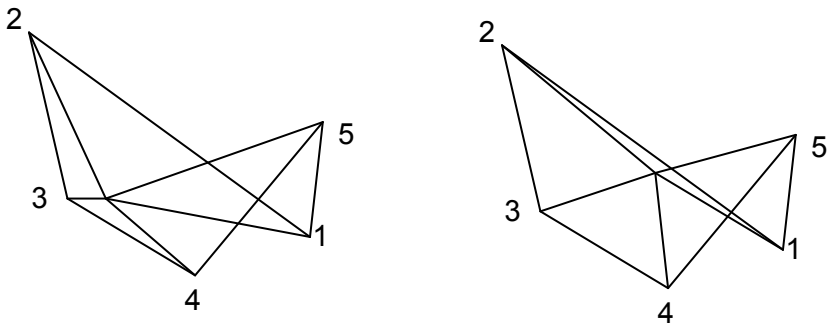

Figure 63: (left) A portion of a tangled, poor quality mesh that is (right) subsequently modified via Laplacian smoothing.

New techniques for improving element quality may also be possible now that untangling is not a strict requirement. For example, the element covers algorithm proposed in Chapter 6 utilized a local tangling of the mesh to improve the geometric quality of a mesh. While this algorithm certainly addresses the large eigenvalue problem associated with poor quality element, it also happens to cause a small eigenvalue problem due to node placement. Thus, element covers need to be investigated further. An additional possibility is to simply allow the nodes of a poor quality element to be relocated to a better position where tangling occurs; see Figure 64 for example.

(a) **(b)**

Figure 64: (a) A mesh with a poor quality element that is subsequently fixed by (b) moving the node to a new position where the mesh happens to be tangled.

Non-Conforming Meshes:

It is often advantageous to use a so called non-conforming mesh, where the mesh conforms to the boundary of the associated domain, but not to adjacent meshes. In doing so, not only does the meshing process become simpler, but a multitude of opportunities arise.

1. By meshing regions separately the mesh size controls set within each region become isolated and no longer poison the mesh of adjacent regions. This effectively allows the analyst to refine the mesh only where needed.
2. This isolation allows designers to operate semi-autonomously as each one need not be concerned about how the other designers are meshing their respective parts. Furthermore, this level of isolation permits companies the opportunity to release a mesh for use in analysis without releasing potentially proprietary CAD models.
3. Use of a non-conforming mesh grants the freedom to assign different element types (e.g. hexahedron or tetrahedron), orders (e.g. linear or quadratic), and physics (e.g. fluid or solid) to each region separately. By properly choosing the type and order within each region analysts can make the best use of available computational resources.

Beyond the aforementioned benefits of using a non-conforming mesh, there are instances where such a mesh is simply unavoidable. Such is the case of modeling contact problems or any other scenario where one mesh is moving relative to another.

In any case, when a non-conforming mesh is used the physics defined on each mesh must be numerically coupled back together at the interface between adjacent meshes. While multiple techniques already exist to perform this step [71–77] I posit tangled meshes can also be used as follows. Consider the non-conforming mesh depicted in Figure 65 and imagine extending the inner mesh so it overlaps the outer mesh as in Figure 66. Now, connect the boundary of the extended mesh to the boundary of the outer mesh with inverted elements – this is a constrained tessellation as the mesh must respect the already existing discretized boundaries. The result is a conforming, tangled mesh that is comprised of the three distinct regions, as in Figure 67.

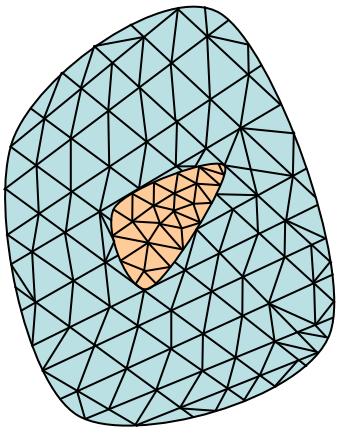

Figure 65: Example non-conforming mesh.

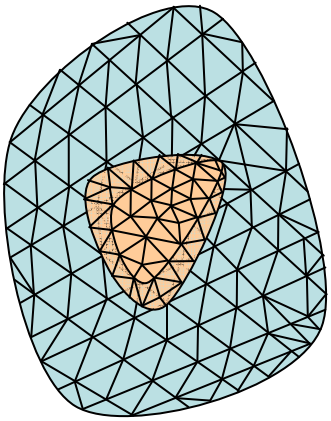

Figure 66: Extension of the inner mesh so that it overlaps the outer mesh.

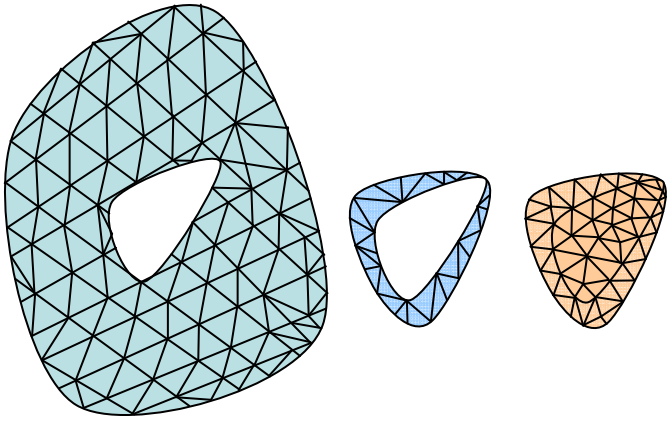

Figure 67: Three distinct regions of the conforming, tangled mesh. The orientation of each region is (from left to right) positive, negative, and positive.

Bibliography

[1] C. S. Chong, A. Senthil Kumar, and H. P. Lee, "Automatic mesh-healing technique for model repair and finite element model generation," *Finite Elements in Analysis and Design*, vol. 43, no. 15, pp. 1109–1119, 2007.

[2] M. W. Beall, "Accessing CAD geometry for mesh generation," Santa Fe, New Mexico, 2003, pp. 33–42.

[3] O. C. Zienkiewicz, *The Finite Element Method: Its Basis and Fundamentals*. Elsevier Butterworth Heinemann, 2005.

[4] G. Strang, *An Analysis of the Finite Element Method*. Englewoods Cliff, NJ: Prentice Hall, 1973.

[5] B. H. V. Topping and et. al., *Finite Element Mesh Generation*. Saxe-Coburg Publications, 2002.

[6] J. Shewchuk, "What is a Good Linear Element? Interpolation, Conditioning, and Quality Measures," 2002, pp. 115–126.

[7] P. M. Knupp, "Algebraic Mesh Quality Metrics," *SIAM J. Sci. Comput.*, vol. 23, no. 1, pp. 193–218, 2001.

[8] P. M. Knupp, "Algebraic mesh quality metrics for unstructured initial meshes," *Finite Elements in Analysis and Design*, vol. 39, no. 3, pp. 217–241, 2003.

[9] P. M. Knupp, "Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. Part I—a framework for surface mesh optimization," *Int. J. Numer. Meth. Engng.*, vol. 48, no. 3, pp. 401–420, 2000.

[10] P. M. Knupp, "Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. Part II—A framework for volume mesh optimization and the condition number of the Jacobian matrix," *Int. J. Numer. Meth. Engng.*, vol. 48, no. 8, pp. 1165–1185, 2000.

[11] S. Bhowmick and S. M. Shontz, "Towards high-quality, untangled meshes via a force-directed graph embedding approach," *Procedia Computer Science*, vol. 1, no. 1, pp. 357–366, May 2010.

[12] *ANSYS 13*. ANSYS; www.ansys.com, 2012.

[13] *Comsol 3.2*. COMSOL.

[14] V. N. Parthasarathy and S. Kodiyalam, “A constrained optimization approach to finite element mesh smoothing,” *Finite Elements in Analysis and Design*, vol. 9, no. 4, pp. 309–320, 1991.

[15] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, “Laplacian mesh optimization,” in *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, Kuala Lumpur, Malaysia: ACM, 2006, pp. 381–389.

[16] L. A. Freitag, “On Combining Laplacian And Optimization-Based Mesh Smoothing Techniques,” ASME, 1997, vol. 220, pp. 37–43.

[17] S. A. Canann, M. B. Stephenson, and T. Blacker, “Optismoothing: An optimization-driven approach to mesh smoothing,” *Finite Elements in Analysis and Design*, vol. 13, no. 2–3, pp. 185–190, 1993.

[18] S. Yamakawa and K. Shimada, “Removing Self Intersections of a Triangular Mesh by Edge Swapping, Edge Hammering, and Face Lifting,” in *Proceedings of the 18th International Meshing Roundtable*, B. W. Clark, Ed. Springer Berlin Heidelberg, 2009, pp. 13–29–29.

[19] L. A. F. a. C. Ollivier-gooch, “A Comparison of Tetrahedral Mesh Improvement Techniques,” 1996, pp. 87–100.

[20] B. M. Klingner and J. R. Shewchuk, “Aggressive Tetrahedral Mesh Improvement,” in *Proceedings of the 16th International Meshing Roundtable*, M. L. Brewer and D. Marcum, Eds. Springer Berlin Heidelberg, 2008, pp. 3–23–23.

[21] C. O.-G. Lori A. Freitag, “Tetrahedral mesh improvement using swapping and smoothing,” *International Journal for Numerical Methods in Engineering*, vol. 40, pp. 3979–4002, 1997.

[22] J. Tournois, R. Srinivasan, and P. Alliez, “Perturbing Slivers in 3D Delaunay Meshes,” in *Proceedings of the 18th International Meshing Roundtable*, B. W. Clark, Ed. Springer Berlin Heidelberg, 2009, pp. 157–173–173.

[23] S. H. Lo, “Finite element mesh generation and adaptive meshing,” *Prog. Struct. Engng Mater.*, vol. 4, no. 4, pp. 381–399, 2002.

[24] J. M. Escobar, E. Rodríguez, R. Montenegro, G. Montero, and J. M. González-Yuste, “Simultaneous untangling and smoothing of tetrahedral meshes,” *Computer Methods in Applied Mechanics and Engineering*, vol. 192, no. 25, pp. 2775–2787, Jun. 2003.

[25] E. J. López, N. M. Nigro, and M. A. Storti, "Simultaneous untangling and smoothing of moving grids," *Int. J. Numer. Meth. Engng.*, vol. 76, no. 7, pp. 994–1019, 2008.

[26] P. Vachal, R. V. Garimella, and M. J. Shashkov, "Untangling of 2D meshes in ALE simulations," *Journal of Computational Physics*, vol. 196, no. 2, pp. 627–644, May 2004.

[27] L. Moresi, F. Dufour, and H.-B. Mühlhaus, "A Lagrangian integration point finite element method for large deformation modeling of viscoelastic geomaterials," *Journal of Computational Physics*, vol. 184, no. 2, pp. 476–497, Jan. 2003.

[28] K. Chrysafinos and N. J. Walkington, "Lagrangian and moving mesh methods for the convection diffusion equation," *EsaimMathematical Modelling and Numerical AnalysisModelisation Mathematique Et Analyse Numerique*, vol. 42, no. 1, pp. 25–55, 2008.

[29] Q. W. Ma, G. X. Wu, and R. Eatock Taylor, "Finite element simulation of fully non-linear interaction between vertical cylinders and steep waves. Part 1: methodology and numerical procedure," *Int. J. Numer. Meth. Fluids*, vol. 36, no. 3, pp. 265–285, 2001.

[30] G. Irving, C. Schroeder, and R. Fedkiw, "Volume conserving finite element simulations of deformable models," *ACM Trans. Graph.*, vol. 26, no. 3, p. 13, 2007.

[31] K. Stein, T. Tezduyar, and R. Benney, "Mesh Moving Techniques for Fluid-Structure Interactions With Large Displacements," *J. Appl. Mech.*, vol. 70, no. 1, pp. 58–63, Jan. 2003.

[32] T. Tezduyar, S. Sathe, K. Stein, and L. Aureli, "Fluid-Structure Interaction," vol. 53, Springer Berlin Heidelberg, 2006, pp. 50–81.

[33] M. L. Staten and et. al., "A Comparison of Mesh Morphing Methods for 3D Shape Optimization," 2011.

[34] J. F. Shepherd, "Proceedings of the 18th International Meshing Roundtable," Springer Berlin Heidelberg, 2009, pp. 85–102.

[35] K. H. Shivanna, S. C. Tadepalli, and N. M. Grosland, "Feature-based multiblock finite element mesh generation," *Computer-Aided Design*, vol. 42, no. 12, pp. 1108–1116, Dec. 2010.

[36] K. Shivanna, N. Grosland, and V. Magnotta, "Proceedings of the 19th International Meshing Roundtable," Springer Berlin Heidelberg, 2010, pp. 85–102.

[37] K. Xu, Z.-Q. Cheng, Y. Wang, Y. Xiong, and H. Zhang, “Quality encoding for tetrahedral mesh optimization,” *Computers & Graphics*, vol. 33, no. 3, pp. 250–261, Jun. 2009.

[38] E. Ruiz-Gironés, X. Roca, and J. Sarrate, “The receding front method applied to hexahedral mesh generation of exterior domains,” *Engineering with Computers*, pp. 1–18.

[39] P. K. Agarwal, B. Sadri, and H. Yu, “Untangling triangulations through local explorations,” in *Proceedings of the twenty-fourth annual symposium on Computational geometry*, College Park, MD, USA, 2008, pp. 288–297.

[40] L. A. Freitag and P. Plassmann, “Local optimization-based simplicial mesh untangling and improvement,” *Int. J. Numer. Meth. Engng.*, vol. 49, no. 1–2, pp. 109–125, 2000.

[41] P. M. Knupp, “Hexahedral and tetrahedral mesh untangling,” *Engineering with Computers*, vol. 17, pp. 261–268, 2001.

[42] L. Xiang-yang and L. A. Freitag, “Optimization-Based Quadrilateral and Hexahedral Mesh Untangling and Smoothing Techniques,” 1999.

[43] V. P. Nguyen, T. Rabczuk, S. Bordas, and M. Duflot, “Meshless methods: A review and computer implementation aspects,” *Mathematics and Computers in Simulation*, vol. 79, no. 3, pp. 763–813, 2008.

[44] T. Belytschko, Y. Y. Lu, and L. Gu, “Element-free Galerkin methods,” *Int. J. Numer. Meth. Engng.*, vol. 37, no. 2, pp. 229–256, 1994.

[45] Q. Duan and T. Belytschko, “On the Stabilization of Stress-Point Integration in the Element Free Galerkin Method,” in *Meshfree Methods for Partial Differential Equations IV*, vol. 65, M. Griebel and M. A. Schweitzer, Eds. Springer Berlin Heidelberg, 2008, pp. 47–68.

[46] C. A. Duarte and J. T. Oden, “H-p cloud -- an h-p meshless method,” *Numer. Methods Partial Differential Eq.*, vol. 12, no. 6, pp. 673–705, 1996.

[47] S. N. Atluri and T. Zhu, “A new Meshless Local Petrov-Galerkin (MLPG) approach in computational mechanics,” in *Computational Mechanics*, vol. 22, Springer Berlin / Heidelberg, 1998, pp. 117–127.

[48] T. Strouboulis, K. Copps, and I. Babuska, “The generalized finite element method,” *Computer Methods in Applied Mechanics and Engineering*, vol. 190, no. 32–33, pp. 4081–4193, 2001.

[49] C. A. Duarte, T. J. Liszka, and W. W. Tworzydlo, “Clustered generalized finite element methods for mesh unrefinement, non-matching and invalid meshes,” *Int. J. Numer. Meth. Engng.*, vol. 69, no. 11, pp. 2409–2440, 2007.

[50] S. R. I. a. E. O. a. N. C. a. F. D. Pin, “The meshless finite element method,” *International Journal for Numerical Methods in Engineering*, vol. 58, pp. 893–912, 2003.

[51] N. S. a. B. M. a. T. Belytschko, “The Natural Element Method in Solid Mechanics,” *International Journal for Numerical Methods in Engineering*, pp. 839–887, 1998.

[52] M. Freytag, V. Shapiro, and I. Tsukanov, “Field modeling with sampled distances,” *Computer-Aided Design*, vol. 38, no. 2, pp. 87–100, 2006.

[53] M. K. Freytag, V. Shapiro, and I. Tsukanov, “Scan and Solve: Acquiring the Physics of Artifacts,” *ASME Conf. Proc.*, vol. 2007, no. 48035, pp. 345–356, 2007.

[54] I. Tsukanov and V. Shapiro, “Meshfree modeling and analysis of physical fields in heterogeneous media,” in *Advances in Computational Mathematics*, vol. 23, Springer U.S., 2005, pp. 95–124.

[55] W. Han, *A posteriori error analysis via duality theory*. Springer, 2005.

[56] J. Fish, S. Markolefas, R. Guttal, and P. Nayak, “On adaptive multilevel superposition of finite element meshes for linear elastostatics,” *Applied Numerical Mathematics*, vol. 14, no. 1–3, pp. 135–164, Apr. 1994.

[57] T. J. Tautges, “Canonical numbering systems for finite-element codes,” *Int. J. Numer. Meth. Biomed. Engng.*, vol. 26, no. 12, pp. 1559–1572, 2010.

[58] J. Shewchuk, “Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates,” *Discrete & Computational Geometry*, vol. 18, no. 3, pp. 305–363, 1997.

[59] N. M. Patrikalakis, T. Maekawa, N. M. Patrikalakis, and T. Maekawa, “Shape Interrogation for Computer Aided Design and Manufacturing,” Springer Berlin Heidelberg, 2010, pp. 109–160.

[60] R. D. Cook, *Concepts and Applications of Finite Element Analysis*. John Wiley & Sons, 2002.

[61] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer, 1999.

[62] T. Ooya, S. Tanaka, and H. Okada, “On the Linear Dependencies of Interpolation Functions in s-Version Finite Element Method,” *JCST*, vol. 3, no. 1, pp. 124–135, Sep. 2008.

[63] R. Vurputoor, N. Mukherjee, J. Cabello, and M. Hancock, “A mesh morphing technique for geometrically dissimilar tesselated surfaces,” 2007, pp. 315–334.

[64] M. L. Staten, S. A. Canann, and S. Owen, “BMSWEEP: locating interior nodes during sweeping,” *Eng. Comput.,*, vol. 15, no. 3, pp. 212–218, 1999.

[65] S. Chalasani, “Quality improvements in extruded meshes using topologically adaptive generalized elements,” Mississippi State UniversityEditor, 2003.

[66] S. Bhowmick and S. M. Shontz, “Towards high-quality, untangled meshes via a force-directed graph embedding approach,” *Procedia Computer Science*, vol. 1, no. 1, pp. 357–366, May 2010.

[67] P. M. Knupp, “Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. Part I A framework for surface mesh optimization,” *Int. J. Numer. Meth. Engng.*, vol. 48, no. 3, pp. 401–420, 2000.

[68] P. M. Knupp, “Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. Part II A framework for volume mesh optimization and the condition number of the Jacobian matrix,” *Int. J. Numer. Meth. Engng.*, vol. 48, no. 8, pp. 1165–1185, 2000.

[69] E. Riks, *The Application of Newton’s Method to the Problem of Elastic Stability*. American Society of Mechanical Engineers, 1972.

[70] J. N. Reddy, *An introduction to nonlinear finite element analysis*. Oxford University Press, 2004.

[71] C. R. Dohrmann, S. W. Key, and M. W. Heinstein, “Methods for connecting dissimilar three-dimensional finite element meshes,” *Int. J. Numer. Meth. Engng.*, vol. 47, no. 5, pp. 1057–1080, 2000.

[72] C. Japhet, Y. Maday, and F. Nataf, “A new Cement to Glue non-conforming Grids with Robin interface conditions: the finite element case,” 2007.

[73] A. Pantano and R. C. Averill, “A penalty-based finite element interface technology,” *Computers & Structures*, vol. 80, no. 22, pp. 1725–1748, Sep. 2002.

[74] Q. Hu, “A regularized domain decomposition method with Lagrange multiplier,” *Advances in Computational Mathematics*, vol. 26, no. 4, pp. 367–401, 2007.

[75] C. Lacour and Y. Maday, “Two different approaches for matching nonconforming grids: the mortar element method and the FETI method.,” HAL - CCSD L2 - HAL:http://hal.archives-ouvertes.fr/hal-00369517/en/, 1997.

[76] F. B. Belgacem, “The Mortar finite element method with Lagrange multipliers,” *Numerische Mathematik*, vol. 84, no. 2, pp. 173–197, 1999.

[77] X. -c. Cai, M. Dryja, and M. Sarkis, “Overlapping Non-Matching Grid Mortar Element Methods For Elliptic Problems,” 1998.

Appendix A: The Simplex-Linear Mesh Morphing Technique

To perform the simplex-linear mesh morphing technique it is assumed that an initial geometry, an initial mesh, and the final morphed geometry are given. The objective is to morph the initial mesh to conform to the final geometry. The simplex-linear method proceeds as follows [33], [63], [64] (based on the mesh from Section 5.2 – figures repeated for clarity):

1. A Delaunay tessellation of the boundary nodes of the initial mesh is constructed (see Figure 68).
2. For each interior node of the initial mesh (see Figure 69) the barycentric coordinates within the Delaunay tessellation is computed.
3. The boundary nodes of initial mesh are morphed to the boundary on the final geometry, using a suitable geometric transformation. The topology of the original Delaunay tessellation is not modified (see Figure 70).
4. Each interior node is now relocated using its original barycentric coordinates, resulting in the mesh of Figure 71; the topology of the mesh remains unchanged.

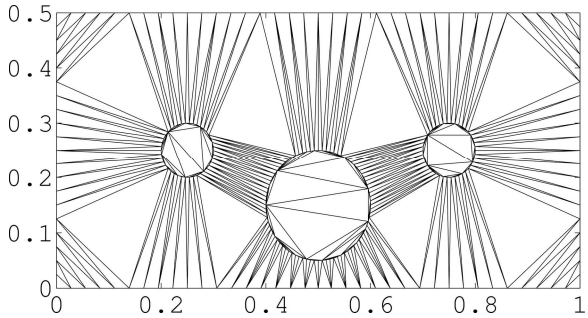

Figure 68: Delaunay triangulation of the boundary nodes.

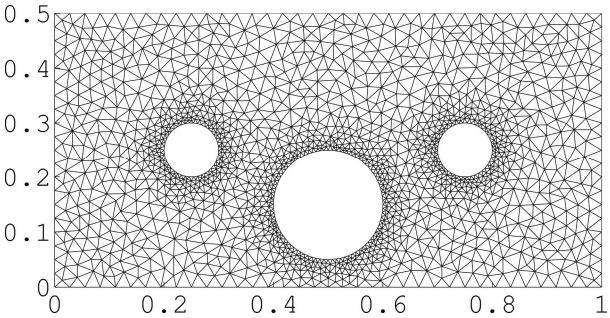

Figure 69: Initial construction of a mesh that is subsequently morphed.

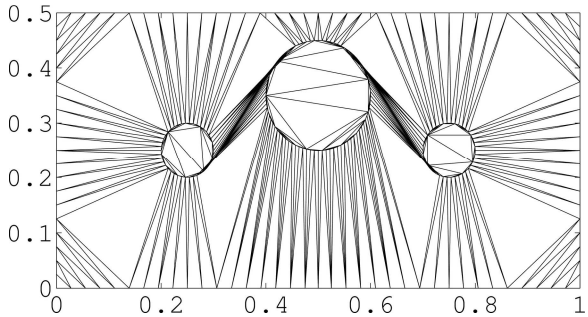

Figure 70: Morphed boundary node triangulation.

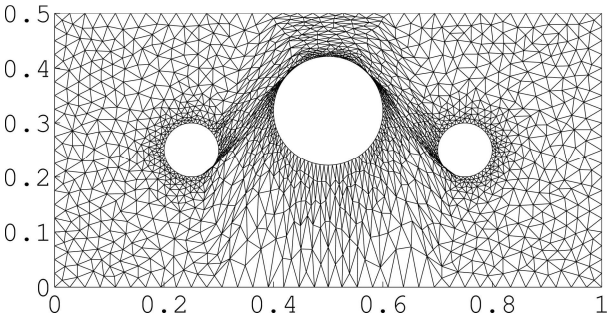

Figure 71: A morphed mesh with a poor function space quality.

Appendix B: Improving the Quality of a Morphed Mesh Through Edge-Swapping

Depending on precisely how a mesh is morphed, the resulting mesh may contain poor quality elements. For the examples used in this document a simple edge-swapping algorithm was implemented. The reason behind this choice is that with tangled meshes the canonical definitions of quality aren't necessarily representative. Moreover, quality improvement methods such as Laplacian smoothing, where a node is moved towards the centroid of its topologically neighboring nodes, do not work in conjunction with tangled meshes. This is due to neighbors being one-sided for a node on a fold; if the node is moved to the centroid the attached elements will become less equilateral, and hence, worse quality. In addition, the edge-swapping algorithm was chosen based on experimental data which shows that the mesh morphing causes the elements to become stretched in one direction, i.e. long edges are created. It thusly seemed appropriate to perform an edge-swap and replace the long edge with a potentially shorter edge.

The algorithm is simple and effective, though not necessarily efficient. The concept is shown below:

1. For each triangle record the edges attached to it.
2. For each edge record (1) the current length and (2) the hypothetical length (swap-length) of the edge if an edge-swap were to occur, and (3) the triangles connected to the edge.
3. Create a sorted view of the edges so that:
    a. Edges whose swap-length that is less that their current length come before those that don't.
    b. For edges with a shorter swap-length order by descending current length.
4. Repeat while the front of the sorted view is an edge with a shorter swap-length:
    a. If the edge-swap would result in an edge that already exists or a (nearly) degenerate triangle:
        i. Set the swap-length to an arbitrarily large number (so it gets moved to the end of the sorted view).
        ii. Update the sorted view.
    b. Else, perform the edge-swap:

i. Destroy the two triangles connected to the edge and create two triangles that are connected to the swapped edge.

ii. Update the current and swap-length of all edges damaged by the swap operation.

iii. Update the sorted view.

Appendix C: Quality Bounds of Cover Algorithm

Recall the cover algorithm developed in Chapter 6. In the description of this algorithm the topic of quality was conveniently avoided. And while the algorithm above stops performing triangle splits when the quality is above a sufficient level, that level was not established. Fortunately, only Class I and Class II triangles need thorough analysis since those are the terminating classes. In performing the analysis, the definition of quality used hereafter is

$$q = \frac{4\sqrt{3}A}{l_1^2 + l_2^2 + l_3^2} \tag{C.1}$$

where A is the area and l_i is the length of edge i. As Class I and II are both isosceles triangles, quality can be reduced to

$$q = \frac{2\sqrt{3}h}{3 + h^2} \tag{C.2}$$

where the base (odd edge) is normalized to length 2 and h is the height; see Figure 72.

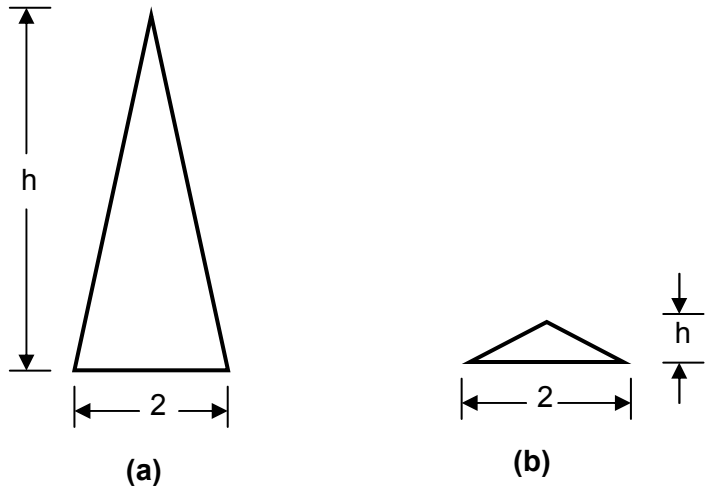

Figure 72: Tall isosceles (a) and short isosceles (b) with normalized height.

Class I – Short Isosceles

Recall that the farthest point on the smallest enclosing circle is used in the split operation. This leads to the triangles shown in Figure 73. By construction the triangle on the right is a 90^o isosceles triangle with a quality of

$$q_1 = \sqrt{3}/2 \tag{C.3}$$

independent of the height of the base triangle. The two triangles on the left are mirror images with a quality of

$$q_2 = \frac{\sqrt{3}(h+1)}{h^2+h+2} \tag{C.4}$$

Figure 74 shows the quality of the base triangle and the quality of the newly added triangles as a function of h. Clearly, when $h>1$ the base triangle should not be split as that would result in lesser quality triangles, and vice versa. Moreover, it can be shown that the minimum quality for this class of triangle is

$$q_{\min} = \frac{\sqrt{3}}{2} \tag{C.5}$$

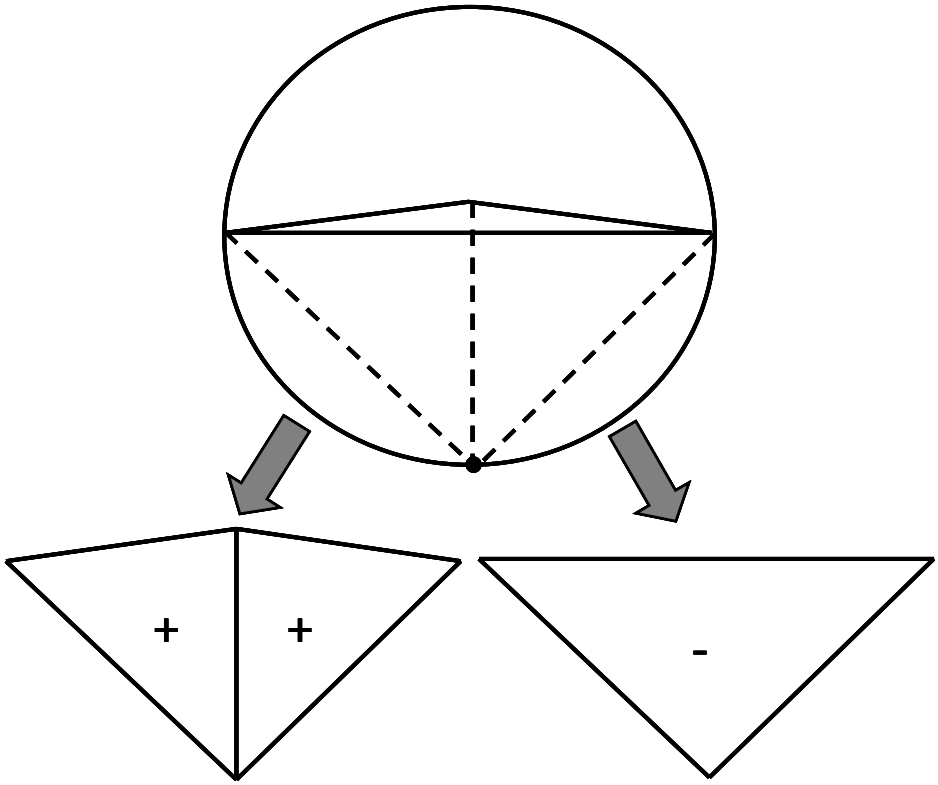

Figure 73: Splitting of a small isosceles triangle into two positively oriented triangle (on left) and one negatively oriented element (on right).

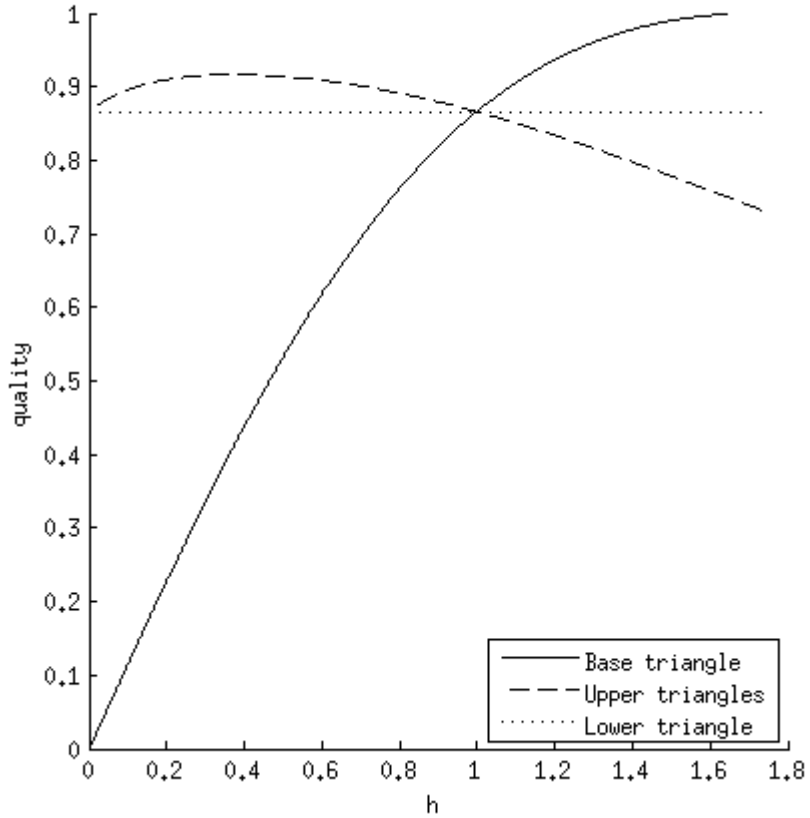

Figure 74: Quality of the base triangle and the newly added triangles as a function of the base triangle normalized height.

Class II – Tall Isosceles

Recall that the split operation for a triangle of this class produces two Class I triangles. While it can be shown that these new triangles will have a normalized height of

$$H = \frac{1}{h} \tag{C.6}$$

we know that the minimum quality upon further splitting is bounded by (C.5). In other words, the Class I triangles that are produced when splitting a Class II triangle will be split exactly once more and then terminate with a quality bound by $\sqrt{3}/2$.

The new Class II triangle generated in the split operation will have a normalized height given by

$$H = \frac{h^2 - 1}{2h} \tag{C.7}$$

which leads to a quality scaling of

$$\lambda = \frac{q(H)}{q(h)} = \frac{2\left(h^2+3\right)\left(h^2-1\right)}{h^4+10h^2+1} \tag{C.8}$$

and is shown in Figure 75. As long as $h > \sqrt{7}$, $\lambda > 1$; and thus, quality will improve. At a height of $h = \sqrt{7}$ the quality of this Class II triangle is $\sqrt{21}/5$ which is still greater than $\sqrt{3}/2$. Therefore, triangles of this class are limited by the two Class I triangles generated during the split.

To recapitulate, by walking over a triangular mesh and enacting this algorithm (with recursion) over each triangle the lower bound on the minimum quality of the mesh is $\sqrt{3}/2$, or approximately 0.866!

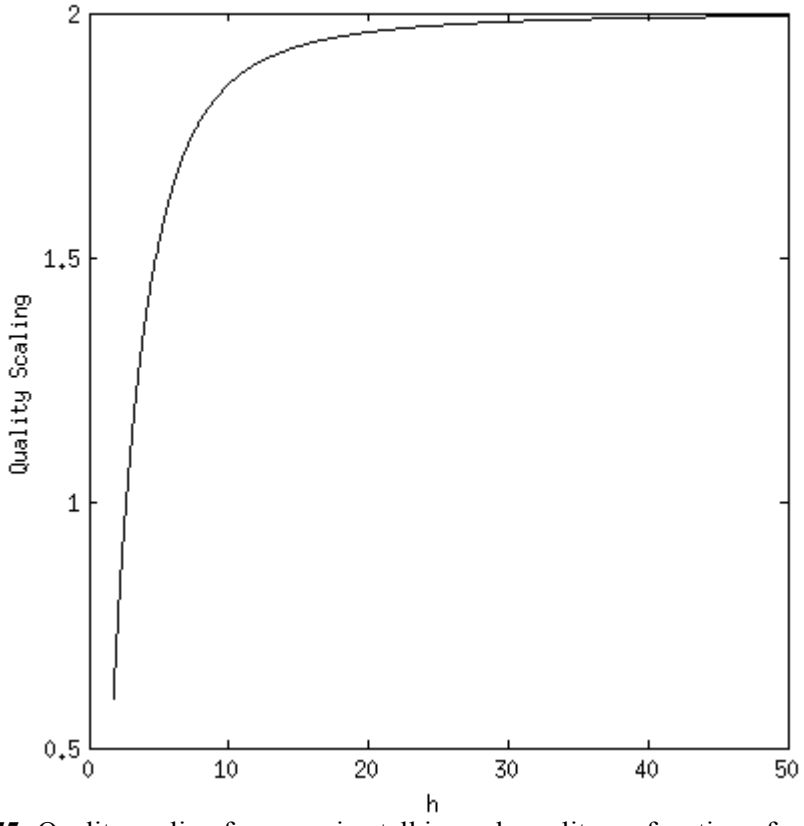

Figure 75: Quality scaling for recursive tall isosceles split as a function of normalized height.

Appendix D: Complexity Analysis of the Cover Algorithm

The very nature of the algorithm defined in Chapter 6 is a cascade from a generic triangle to Class I or Class II triangles, with immediate termination or Class I and eventual (after recursion) termination of Class II. The cascade is at most two steps – Class IV to Class III to Class I and Class II – which can be considered a constant time operation. Consequently, the only operation of interest is the recursive splitting of a Class II triangle into a better Class II triangle.

Recall the quality growth factor λ defined by (C.8). When h is large, i.e. it is a very tall, terrible quality triangle, λ is approximately 2. As the triangle is recursively split into better quality Class II triangles the growth factor decreases. Since the quality was bound at $\sqrt{3}/2$ it only pays to do the recursive split until quality reaches this point. It can be shown that the triangle before the final split will have a normalized height of

$$h = 3 + \sqrt{10} \tag{D.1}$$

a quality of

$$q = \frac{\sqrt{3}\left(3 + 2\sqrt{10}\right)}{31} \approx 0.521 \tag{D.2}$$

and a growth factor of

$$\lambda = \sqrt{10} - 3/2 \approx 1.66 \tag{D.3}$$

Thus, at all times in the recursive splitting of Class II triangles $\lambda \in [1.66, 2)$. This leads to a conservative bound on number of splits at

$$k \sim \log_{1.66}\left(\frac{q}{q_0}\right) \tag{D.4}$$

where k is the number of splits, q_0 is the starting quality (before splitting) and q is the final quality desired (maximum of $\sqrt{3}/2$). More realistically, though, when quality is very low this bound becomes

$$k \sim \log_{2}\left(\frac{q}{q_0}\right) \tag{D.5}$$

In either case, the asymptotic number of splits required is logarithmic with respect to the desired quality improvement. This logarithmic nature also carries over to the number of triangles and points added to the original mesh. This is easily shown since each split operation adds exactly one point and adds three triangles but removes one.