Tangled Finite Element Method (TFEM) for Quadrilateral Meshes

Bhagyashree Prabhune[a], Saketh Sridhara[a], Krishnan Suresh[a,*]

[a]*Department of Mechanical Engineering, University of Wisconsin-Madison, WI, USA*

Abstract

In finite element method (FEM), a mesh is said to be tangled if it contains inverted elements. Tangling can occur, for example, during mesh generation, mesh morphing, shape optimization, and/or large deformation simulation. In standard FEM, tangled meshes can lead to erroneous results; further, untangling may not always be possible.

The objective of this paper is to introduce a tangled finite element method (TFEM) for 2D quadrilateral meshes. In particular, we consider here *implicitly* tangled meshes where some of the elements are only partially inverted, i.e., are concave; such meshes are harder to untangle than *explicitly* tangled meshes where the elements are fully inverted.

The proposed TFEM framework fundamentally rests on an unambiguous definition of the field in the tangled region. This definition naturally leads to certain correction terms in the FEM stiffness matrix. Once these corrections and an equality condition are included, we demonstrate that accurate solutions can be obtained over even severely tangled meshes. The theoretical properties of the proposed TFEM are established, and the implementation is discussed in detail. Several numerical experiments are carried out to illustrate the robustness of the proposed method.

Keywords: Tangled Mesh, Quadrilateral mesh, Finite Element Method, TFEM, Concave elements, Self-intersection

1. Introduction

The finite element method (FEM) [1, 2] is a ubiquitous choice for solving boundary-value problems [3]. FEM fundamentally relies on constructing a mesh that must satisfy various requirements: (1) be simply connected (2) conform to the boundary (3) be of good quality and (4) not contain inverted elements [2]. The focus of this paper is on a finite element formulation for handling *inverted* elements.

An element is said to be inverted if the determinant of the Jacobian associated with the *parametric mapping* [2], that underlies FEM, is negative at any point within the element. A mesh containing one or more inverted elements is said to be *tangled* (or folded). Further, one can distinguish between *explicit* tangling and *implicit* tangling [4, 5]. Explicit tangling is when an element is completely inverted, i.e., the

*Corresponding author

Email addresses: bprabhune@wisc.edu (Bhagyashree Prabhune), ssridhara@wisc.edu (Saketh Sridhara), ksuresh@wisc.edu (Krishnan Suresh)

determinant of the Jacobian is negative everywhere within the element as in Fig. 1a; such elements will explicitly overlap with their neighbors. This typically occurs in linear triangle and tetrahedral meshes, but can also occur in other types of meshes [6]. On the other hand, *implicit* tangling is when an element is partially inverted, i.e., the determinant of the Jacobian is negative within parts of the element as in Fig. 1b; such elements only implicitly overlaps with their neighbors. This is elaborated later in the paper, and is more common in quadrilateral and hexahedral meshes.

(a) Explicit tangling.

(b) Implicit tangling.

Figure 1: Inverted elements are highlighted.

It is well-known that inverted elements lead to erroneous results in FEM. To quote [7], "*Even a single concave, or inverted element makes a mesh unusable for simulation*". Unfortunately, tangling can occur, for example, during: (a) mesh generation [8], (b) mesh optimization [9, 10, 11], [12], (c) large deformation [13], and (d) shape optimization [14].

Various strategies have been proposed to address tangling; untangling is perhaps the most common [8, 9, 10, 11]. However, untangling is not always possible/reliable. To quote [12], "*...there are no known a priori test to determine if a mesh can be untangled*". Further, untangling can be expensive, and can pose challenges in mapping of simulation data.

Methods to directly handle inverted elements have also been proposed. For instance, the method of invertible finite elements [15, 16] can handle inverted *simplicial* elements formed during large deformation simulations; however the theoretical basis for this method was not established. Methods such as smoothed finite element [17], and various polygonal finite element methods [18, 19] using harmonic [20] and maximum-entropy shape functions [21], virtual element method [22], n-sided smoothed finite element method [23] allow for partially inverted (concave) quadrilateral elements. However, they entail major implementation changes to FEM. For instance, many of these are based on shape functions different from those of FEM.

An alternate strategy was proposed in [4, 5] for handling *explicitly* tangled triangle and tetrahedral meshes. The main feature of this method is that it uses the same basis functions as FEM, and tangling is accounted for by simply incorporating correction terms to the standard stiffness matrix (see Section 2), thus requiring minimal modification to the existing FEM framework. This was later extended to handle *explicitly* tangled quadrilateral meshes in [6].

The objective of this paper is to generalize this method to handle *implicitly* tangled quadrilateral meshes,

that are more difficult to handle, and more common than explicitly tangled meshes [24]. The proposed method, referred to here as *tangled finite element method* (TFEM), fundamentally rests on an unambiguous definition of the unknown field in the tangled region. This definition naturally leads, once again, to certain correction terms in the FEM stiffness matrix. In addition, an equality condition must also be imposed in implicit tangling. When these corrections and constraint are included, we demonstrate that accurate solutions can be obtained over even severely tangled meshes.

The remainder of paper is organized as follows. The background on handling explicitly tangled meshes is reviewed in Section 2. Section 3 describes the proposed TFEM formulation for implicit tangling, and its theoretical properties. Section 4 discusses the implementation of TFEM. This is followed by numerical experiments in Section 5, and conclusions in Section 6.

2. Background

In this section, the concept for handling explicit tangling is briefly reviewed.

2.1. Motivation

To illustrate explicit tangling, we consider the following Poisson problem:

Find $u \in H_0^1(\Omega)$

$$\int_\Omega (\nabla v)^\top \boldsymbol{D} (\nabla u)\, d\Omega = \int_\Omega v f d\Omega \tag{1}$$

$\forall v \in H_0^1(\Omega)$

where, $\Omega \subset \mathbb{R}^n, n = 1, 2, 3$ is a domain and $\boldsymbol{D}$ is a constitutive matrix. To solve the above problem over a finite element mesh, recall that in FEM, the field at any point within a mesh element E_j is approximated using shape functions as follows:

$$u^h(p) = \boldsymbol{N}_j(p)\hat{\boldsymbol{u}}_j \tag{2}$$

where, $\boldsymbol{N}_j = \{N_j^{i_1}(p) \;\; N_j^{i_2}(p) \ldots N_j^{i_m}(p)\}$ are the shape functions associated with the nodes of that element, and $\hat{\boldsymbol{u}}_j = \{u^{i_1} \;\; u^{i_2} \ldots u^{i_m}\}^\top$ are the unknown nodal degrees of freedom associated with E_j. In the Galerkin formulation [2], this leads to:

$$\left[\int_\Omega \left(\nabla \sum_j \boldsymbol{N}_j\right)^\top \boldsymbol{D} \left(\nabla \sum_k \boldsymbol{N}_k\right) d\Omega\right] \hat{\boldsymbol{u}} = \int_\Omega \left(\sum_j \boldsymbol{N}_j\right)^\top f d\Omega \tag{3}$$

Upon numerical integration, this yields a linear system of equations:

$$\boldsymbol{K}^0 \hat{\boldsymbol{u}} = \boldsymbol{f} \tag{4}$$

where $\hat{\boldsymbol{u}}$ represent unknown nodal degrees of freedom, and

$$\boldsymbol{K}^0 = \prod_{Assemble} \int_{E_j} \nabla \boldsymbol{N}_j^\top \boldsymbol{D} \nabla \boldsymbol{N}_j d\Omega, \tag{5}$$

$$\boldsymbol{f} = \prod_{Assemble} \int_{E_j} \boldsymbol{N}_j^\top f d\Omega. \tag{6}$$

To understand the impact of tangling, consider a 1D domain with $f = 1$. The domain is discretized into 3 elements, each associated with linear shape functions as illustrated in Fig. 2b; observe that the mesh is not tangled. The computed FEM solution is illustrated in Fig. 2c, together with the exact solution $u = x(1-x)/2$.

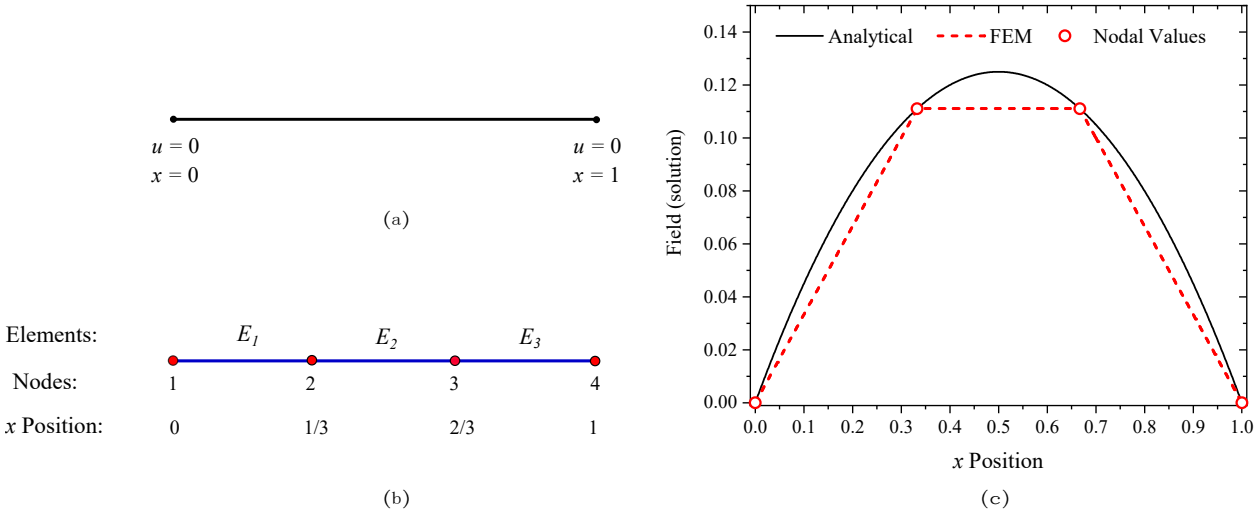
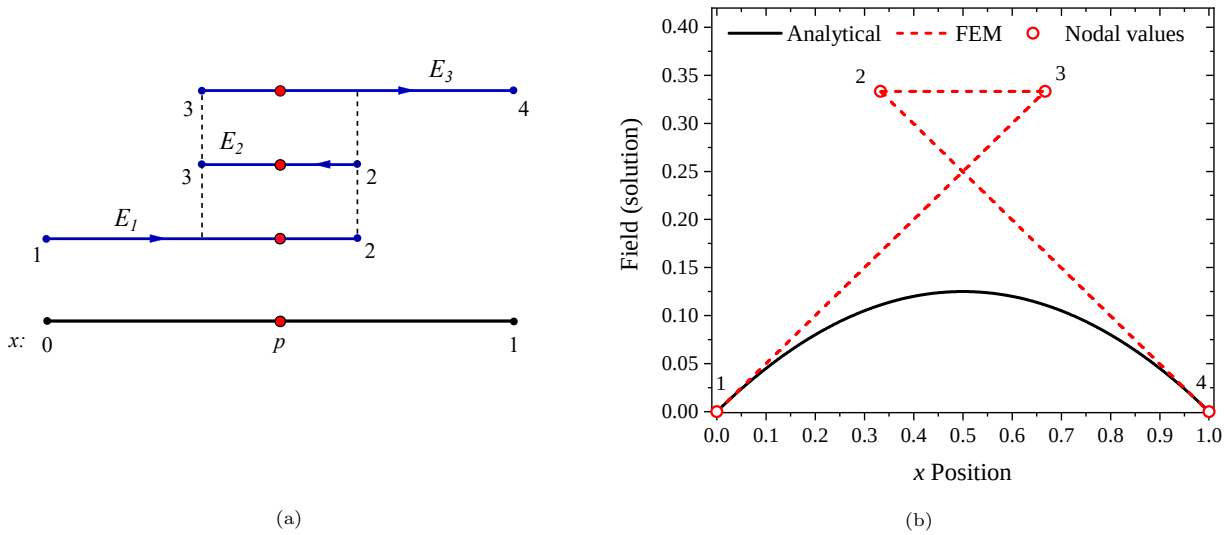


Figure 2: (a) One-dimensional domain, (b) discretized into linear three elements (c) solution obtained by FEM

Now, we will tangle the mesh as follows: vertex 2 is moved to the right (to 2/3) , while vertex 3 is moved to the left (to 1/3) as illustrated in Fig. 3a. In other words, element E_2 is flipped, i.e., the determinant of the Jacobian for E_2 is negative everywhere within the element. If we employ ANSYS APDL 2020 R2 to solve Eq. 1 over the given explicitly tangled mesh, the resulting solution is illustrated in Fig. 3b. The solution at vertex 2 and vertex 3 is $u = 1/3$. Observe the following: (1) since there are three elements that occupy the region $x \in (1/3, 2/3)$, the field is ambiguous in this folded region (indeed, one obtains different values for the field at $x = 0.5$, depending on the element we use to compute the field), and (2) in the regions $x \in [0, 1/3)$ and $x \in (2/3, 1]$, the field is unambiguous but deviates significantly from the exact solution.



(a) (b)

Figure 3: (a) Explicit tangling in 1D with three elements (b) Solution obtained by FEM.

2.2. Concept for handling explicit tangling

Of course, it is easy to untangle the mesh in the above example. However, we will study how one can handle tangling in a modified FEM framework. As noted earlier, any point in the tangled region can be interpreted as belonging to multiple elements; see Fig. 3a. Thus the field is ambiguous in the tangled region, and Eq. 2 is invalid. Solving this ambiguity is considered the first step to solving the tangling problem. Towards this end, we define the orientation θ_j of an element E_j as the sign of the determinant of Jacobian associated with that element.

$$\theta_j \equiv sgn(|\boldsymbol{J}_j|) \tag{7}$$

In the tangled mesh in Fig. 3a, $\theta_1 = \theta_3 = +1$, and $\theta_2 = -1$. Then, the field at any point is *defined* [4, 5] as the oriented sum of the contributions from all elements the point belongs to:

$$u^h(p) \equiv \sum_{j|p\in E_j} \theta_j \boldsymbol{N}_j(p)\hat{\boldsymbol{u}}_j \tag{8}$$

Observe the fundamental difference between Eq. 8 and Eq. 2. Substituting Eq. 8 in the Galerkin formulation of Eq. 1 leads to:

$$\left[\int_\Omega \left(\nabla \sum_j \theta_j \boldsymbol{N}_j\right)^\top \boldsymbol{D} \left(\nabla \sum_k \theta_k \boldsymbol{N}_k\right) d\Omega\right] \hat{\boldsymbol{u}} = \int_\Omega \left(\sum_j \theta_j \boldsymbol{N}_j\right)^\top f d\Omega \tag{9}$$

The left-hand side (LHS) of Eq. 9 can be regrouped as:

$$\sum_j \int_\Omega \theta_j^2 \left(\nabla \boldsymbol{N}_j\right)^\top \boldsymbol{D} \left(\nabla \boldsymbol{N}_j\right) d\Omega + \sum_j \sum_{k\neq j} \int_\Omega \theta_j\theta_k \left(\nabla \boldsymbol{N}_j\right)^\top \boldsymbol{D} \left(\nabla \boldsymbol{N}_k\right) d\Omega$$

Since $\theta_j = \pm 1$, $\theta_j^2 = 1$, Eq. 9 simplifies to:

$$\left[\sum_j \int_{E_j} \nabla \boldsymbol{N}_j^\top \boldsymbol{D} \nabla \boldsymbol{N}_j d\Omega + \sum_j \sum_{k\neq j} \int_{E_j\cap E_k} \theta_j\theta_k \nabla \boldsymbol{N}_j^\top \boldsymbol{D} \nabla \boldsymbol{N}_k d\Omega\right] \hat{\boldsymbol{u}} = \sum_j \int_{E_j} \theta_j \boldsymbol{N}_j^\top f d\Omega \tag{10}$$

The first term on the left hand side of Eq. 10 corresponds exactly to the stiffness matrix in Eq. 4. The second term captures the coupling between overlapping elements. Observe that, while there may be many overlapping elements that contain the same point, only one element pair need to be considered at a time. The right-hand side (RHS) of Eq. 10 is also different from the RHS of Eq. 4. In summary, Eq. 10 can be written as:

$$\left(\boldsymbol{K}^0 + \boldsymbol{K}^N\right)\hat{\boldsymbol{u}} = \boldsymbol{f}^\theta \tag{11}$$

where

$$\begin{aligned}
\boldsymbol{K}^0 &= \prod_{Assemble} \int_{E_j} \nabla \boldsymbol{N}_j^\top \boldsymbol{D} \nabla \boldsymbol{N}_j d\Omega \\
\boldsymbol{K}^N &= \prod_{Assemble} \int_{E_j \cap E_{k\neq j}} \theta_j \theta_k \nabla \boldsymbol{N}_j^\top \boldsymbol{D} \nabla \boldsymbol{N}_k d\Omega \\
\boldsymbol{f}^\theta &= \prod_{Assemble} \int_{E_j} \theta_j \boldsymbol{N}_j^\top f d\Omega
\end{aligned} \tag{12}$$

Thus the cross-term $\boldsymbol{K}^N$ and signed forcing term $\boldsymbol{f}^\theta$ must be computed; the superscript 'N' denotes stiffness components due to explicit tangling, typically involving neighboring elements. Observe that if there are no overlaps, $\boldsymbol{K}^N$ is identically zero and $\boldsymbol{f}^\theta$ is exactly equal to $\boldsymbol{f}$. The solution obtained by solving Eq. 11 over the tangled mesh is illustrated in Fig. 4. Not only is the field unambiguous everywhere, the accuracy matches that of FEM over an untangled mesh.

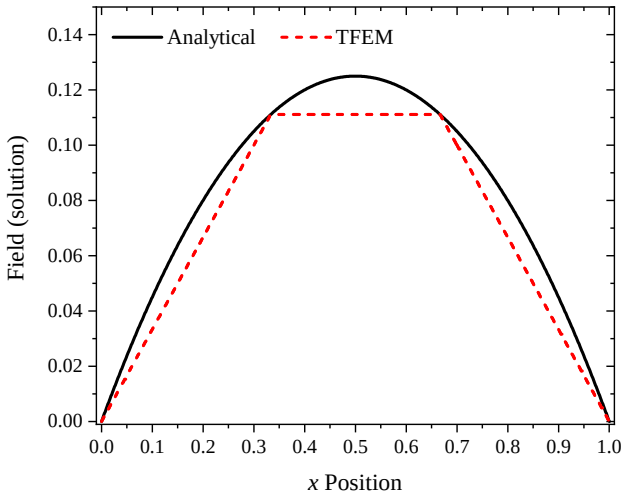


Figure 4: Solution for tangled mesh using Eq. 11.

2.3. Explicit Tangling in 2D

The above concept for explicit tangling is applicable in higher dimension as well. For example, consider the 2D domain Ω discretized into bilinear quadrilateral elements as in Fig. 5a. The elements are convex and non-overlapping. One can easily create an explicitly tangled mesh as illustrated in Fig. 5b and Fig.

5c. For this tangled mesh, the orientation of the flipped elements are negative *throughout* that element, and thus the mesh is said to explicitly tangled. The formulation discussed in the previous section (from Eq. 7 to Eq. 10) can be generalized to handle such higher dimensional problems. Specifically, explicitly tangled simplicial meshes are discussed in [4] while explicitly tangled quadrilateral meshes are handled in [6].

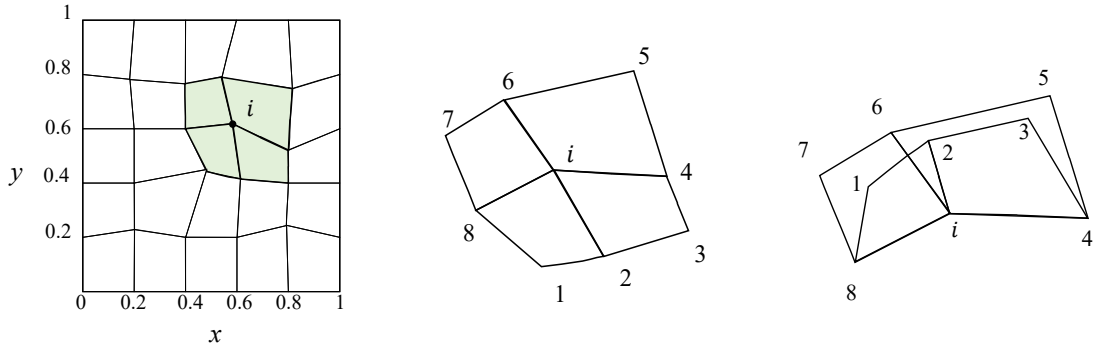


Figure 5: Explicit tangling in quads: (a) regular mesh (b) elements associated with vertex i (c) explicit tangling.

3. Proposed TFEM Framework for Implicit Tangling

The objective now is to generalize the above formulation to handle implicitly tangled quadrilateral meshes. The fundamental characteristic of an implicitly tangled mesh is that some of the elements may be concave, i.e., only partially inverted. As an example, consider the unit square domain with two quadrilateral elements in Fig. 6a, one of which is concave as illustrated in Fig. 6b. Although the two elements are not explicitly overlapping, there is implicit tangling due to the concave element, as explained below. This, as is well known, will lead to erroneous results in FEM.

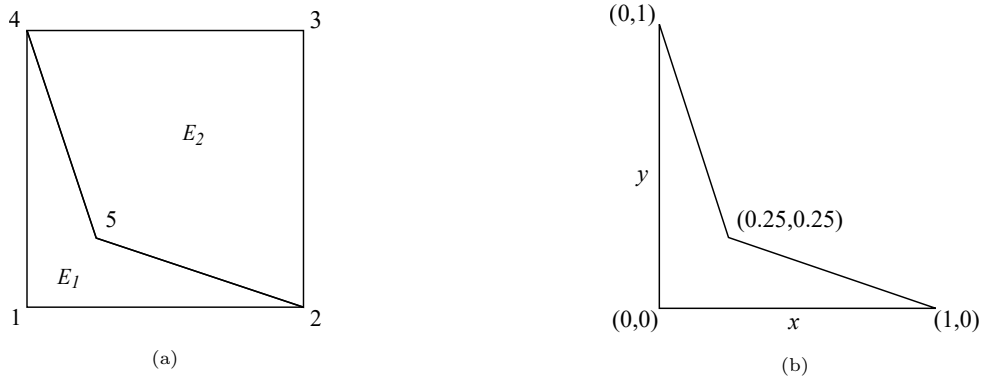


Figure 6: (a) 2-D domain discretized into two bilinear quads. (b) Concave element.

3.1. Challenges in Implicit Tangling

To understand the challenges underlying implicit tangling, consider the standard isoparametric bilinear mapping from the parametric (ξ, η) space of a concave element in Fig. 7a to the physical space (x, y) in Fig. 7b. Given the coordinates of the four nodes (x^i, y^i) of a quad element, recall that the parametric mapping is defined via the standard bilinear shape functions $N^i(\xi, \eta)$ [2]:

$$x(\xi,\eta) = \sum_{i=1}^{4} N^i(\xi,\eta)x^i; \quad y(\xi,\eta) = \sum_{i=1}^{4} N^i(\xi,\eta)y^i \tag{13}$$

For the particular concave element in Fig. 7b this reduces to:

$$x\,(\xi,\eta) = \frac{(1+\xi)(5-3\eta)}{16}; \quad y\,(\xi,\eta) = \frac{(5-3\xi)(1+\eta)}{16} \tag{14}$$

There are several implications of this mapping.

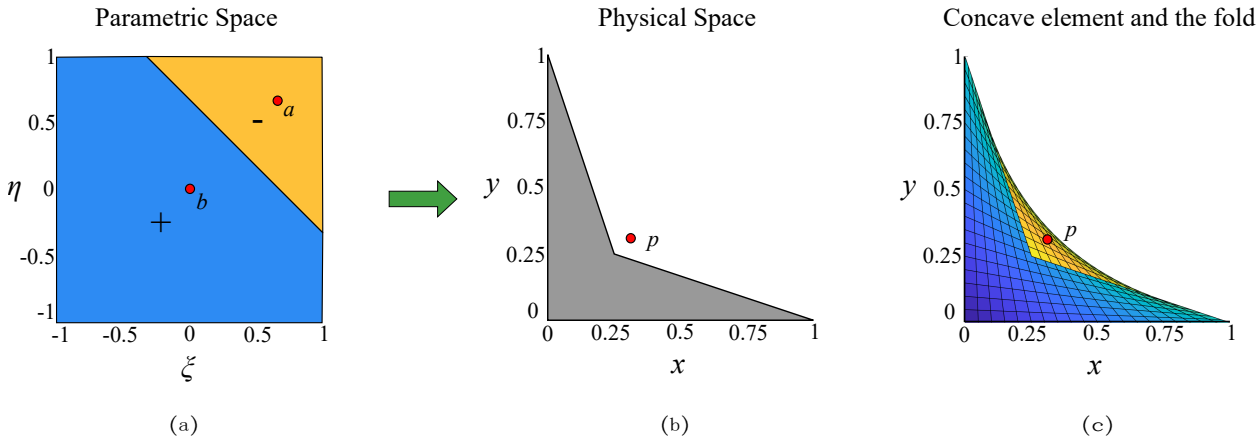


Figure 7: (a), (b) Parametric mapping of the concave quad onto the physical space (c) Concave element with the folded region.

First, note that the determinant of the Jacobian associated with this mapping is given by:

$$|\boldsymbol{J}| = \begin{vmatrix} x_{,\xi} & x_{,\eta} \\ y_{,\xi} & y_{,\eta} \end{vmatrix} = \frac{2-3\xi-3\eta}{32}. \tag{15}$$

Therefore, $|\boldsymbol{J}|$ vanishes on the line $3\xi + 3\eta = 2$, dividing the parametric space into a positive $|\boldsymbol{J}|$ region and a negative $|\boldsymbol{J}|$ region as illustrated in Fig. 7a. In other words, the determinant of the Jacobian changes sign within the element. Although $|\boldsymbol{J}| = 0$ corresponds to a straight line in the parametric space, the corresponding curve in the physical space is quadratic (see Fig. 7c).

Second, all points in the parametric space with a negative $|\boldsymbol{J}|$ map to points outside the element. For example, the point $a(\xi = 2/3, \eta = 2/3)$ in the parametric space maps to the point $p(x = 5/16, y = 5/16)$ that is outside the concave element as depicted in Fig. 7a and Fig. 7b.

Third, for every point a in the negative $|\boldsymbol{J}|$ space, there is a corresponding point b in the positive $|\boldsymbol{J}|$ space that maps to the same physical point. As illustrated in Fig. 7a parametric points $a(\xi = 2/3, \eta = 2/3)$

and $b(\xi = 0, \eta = 0)$ map to the same physical point $p(x = 5/16, y = 5/16)$ in Fig. 7b. Since two different points in the parametric space map to the same physical point, the mapping is non-invertible. In other words, the element overlaps with itself, creating a folded region, and resulting in a self-tangled or implicitly tangled element as illustrated in Fig. 7c.

Given these observations, for the concave element E_1, we define a positive (negative) component $C_1^+(C_1^-)$ as the set of points in the physical space that map from positive (negative) part of the parametric space. Thus E_1 can be expressed as the difference between the two components (see Fig. 8a and Fig. 8b):

$$E_1 = C_1^+ - C_1^- \tag{16}$$

Further, it can be observed from Fig. 8b that

$$C_1^+ \cap C_1^- = C_1^- \tag{17}$$

since C_1^- is the subset of C_1^+. On the other hand, the convex element E_2 has only the positive component (see Fig. 8c):

$$E_2 = C_2^+;\ C_2^- = \emptyset. \tag{18}$$

Finally, since the folded region C_1^- lies outside the concave element, it overlaps with the neighboring convex element as illustrated in Fig. 8d. In this case, we have only one neighboring element. But in general, the folded region can overlap with multiple neighboring elements as illustrated later in Section 4.

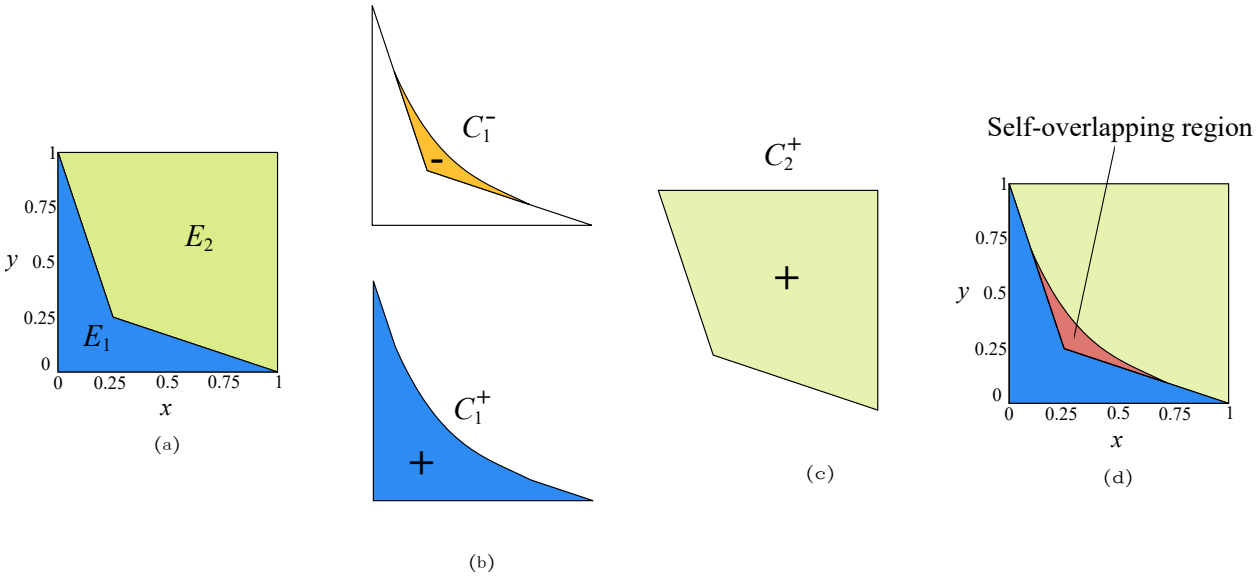


Figure 8: (a) Implicitly tangled mesh with two elements (b) Positive and negative $|\boldsymbol{J}|$ regions of the concave element (c) Convex element of the mesh (d) Final physical space is self-overlapping.

In summary, for a concave bilinear quad element, (1) $|\boldsymbol{J}|$ takes both positive and negative values, (2) all points in the negative $|\boldsymbol{J}|$ space map to points outside the physical element, (3) the parametric mapping is non-invertible, and (4) a mathematical fold exists in the physical space which makes the element not

only self-intersect, but also intersect with the neighboring element(s). The proposed TFEM methodology to handle such implicitly tangled meshes is discussed next. Note that, in this paper, we do not consider twisted quadrilateral elements such as the one in Fig. 9.

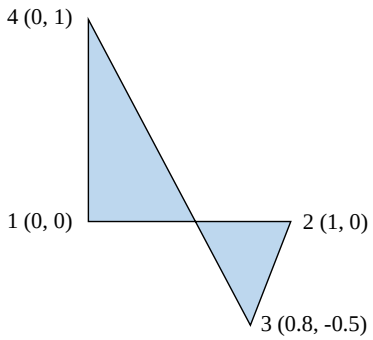


Figure 9: Twisted quadrilateral elements are not considered here.

3.2. Field Uniqueness and Equality Condition

As noted earlier, any point in the implicitly tangled region can be interpreted as belonging to: (a) different parametric regions of the same element, and (b) multiple elements. Thus, the field is ambiguous in the tangled/folded region and Eq. 2 is invalid. As in explicit tangling, resolving this ambiguity is the first step. Towards this end, observe that since a point within an element can belong to two different parametric regions, the notion of shape functions at a physical point needs to be clarified. We therefore propose the following natural definition.

Definition: For a concave element E_j, $\boldsymbol{N}_j^+$ are the four shape functions of element E_j evaluated at point p corresponding to the component C_j^+, while $\boldsymbol{N}_j^-$ are the four shape functions evaluated at point p corresponding to the component C_j^-.

For example, in Fig. 7b, $\boldsymbol{N}_1^+(p)$, where $p = (x = 5/16, y = 5/16)$, are the shape functions of element E_1 evaluated at $(\xi = 0,\ \eta = 0)$ whereas $\boldsymbol{N}_1^-(p)$ are the shape functions of element E_1 evaluated at $(\xi = 2/3,\ \eta = 2/3)$.

Definition: For a convex element E_k, $\boldsymbol{N}_k^+(p)$ are the four shape functions of element E_k evaluated at point p, while $\boldsymbol{N}_k^-(p)$ are assumed to be zero since the component C_k^- does not exist.

With these definitions, the field at any point p is defined in TFEM as the oriented sum of the contributions from all *components* the point belongs to:

$$u^h(p) \equiv \sum_{j|p\in C_j^+} \boldsymbol{N}_j^+(p)\hat{\boldsymbol{u}}_j - \sum_{j|p\in C_j^-} \boldsymbol{N}_j^-(p)\hat{\boldsymbol{u}}_j \tag{19}$$

Observe that the first term on RHS of Eq. 19 corresponds to C_j^+, and the second to C_j^-. We refer to Eq. 19 as the *unambiguity* condition, i.e., given the nodal values, the field is now unambiguously defined everywhere. This will be used later to derive the TFEM stiffness matrix.

In addition we will need a second set of equations. Specifically, consider the point q in Fig. 10. Since this point belongs to 3 different components C_1^+, C_1^- and C_2^+, from Eq. 19, the field at q is given by:

$$u^h(q) = \boldsymbol{N}_1^+(q)\hat{\boldsymbol{u}}_1 - \boldsymbol{N}_1^-(q)\hat{\boldsymbol{u}}_1 + \boldsymbol{N}_2^+(q)\hat{\boldsymbol{u}}_2 \tag{20}$$

where $\hat{\boldsymbol{u}}_1$ are the four nodal values of element E_1, and $\hat{\boldsymbol{u}}_2$ are the four nodal values of element E_2.

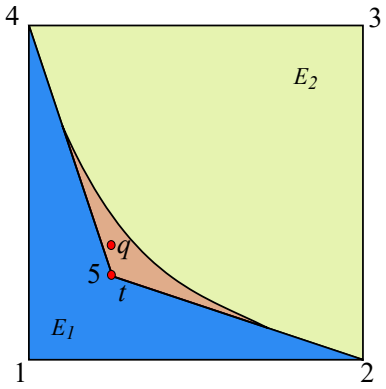


Figure 10: Equality condition must be imposed at all concave vertices.

In the limit $q \to t$, i.e., in the limit of q approaches the physical location corresponding to vertex v_5, the last two terms in Eq. 20 cancel each other, resulting in

$$u^h(t) = \boldsymbol{N}_1^+(t)\hat{\boldsymbol{u}}_1 \tag{21}$$

i.e.,

$$u^h(t) = N_1^{1+}(t)u^1 + N_1^{2+}(t)u^2 + N_1^{3+}(t)u^5 + N_1^{4+}(t)u^4 \tag{22}$$

We will require here that $u^h(t)$ must match the corresponding nodal value u^5, i.e., we impose $u^h(t) = u^5$, resulting in

$$N_1^{1+}(t)u^1 + N_1^{2+}(t)u^2 + N_1^{4+}(t)u^4 + (N_1^{3+}(t) - 1)u^5 = 0 \tag{23}$$

Eq. 23 is referred to here as the *equality* condition that must be imposed at re-entrant vertices of all concave elements in the mesh. In the next section, we will show that this condition, together with the unambiguity condition, is essential to capture constant strain fields.

3.3. Theoretical Properties of TFEM

Recall that in standard FEM, the field must satisfy three conditions for convergence [2]:

1. Continuity: The field must be continuous within the element, and across element boundaries.
2. Rigid body: The element must be strain free under rigid body (constant field) conditions.
3. Constant strain: One must be able reproduce constant strain conditions exactly.

We show here that these three conditions are precisely met in TFEM.

3.3.1. Continuity

To establish continuity, we consider the field at a few points belonging to different regions of an implicitly tangled mesh in Fig. 11. For points such as p and o (see Fig. 11), that are not in the fold, the field can be computed as in classic FEM (see Eq. 2):

$$u^h(p) = \boldsymbol{N}_2^+(p)\hat{\boldsymbol{u}}_2 \tag{24}$$

$$u^h(o) = \boldsymbol{N}_1^+(o)\hat{\boldsymbol{u}}_1 \tag{25}$$

where $\hat{\boldsymbol{u}}_2$, for example, are the four nodal solutions for element 2. From the continuity of the shape functions, it follows that the field is also continuous at these points.

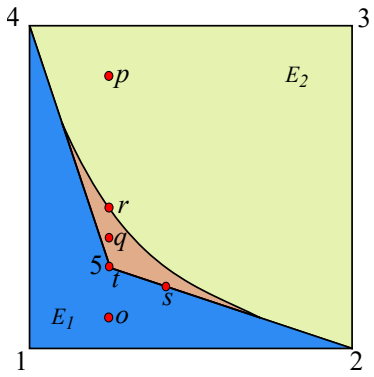


Figure 11: Post processing in an implicitly tangled mesh.

On the other hand, for points such as q that belongs to multiple components C_1^+, C_1^- and C_2^+, the field is computed from Eq. 19 as follows:

$$u^h(q) = \left(\boldsymbol{N}_1^+(q) - \boldsymbol{N}_1^-(q)\right)\hat{\boldsymbol{u}}_1 + \boldsymbol{N}_2^+(q)\hat{\boldsymbol{u}}_2 \tag{26}$$

Once again it is clear that the field is continuous within the fold. Next consider a point r located on the curved edge (corresponding to $|\boldsymbol{J}| = 0$) of the fold. This is a special case of Eq. 26 in that:

$$u^h(r) = \lim_{q \to r}\left(\boldsymbol{N}_1^+(q) - \boldsymbol{N}_1^-(q)\right)\hat{\boldsymbol{u}}_1 + \boldsymbol{N}_2^+(q)\hat{\boldsymbol{u}}_2 \tag{27}$$

Recall that, since point r lies on the $|\boldsymbol{J}| = 0$ curve in the physical space, it maps to a unique point in the parametric space, i.e., $\boldsymbol{N}_1^+(r) = \boldsymbol{N}_1^-(r)$. Thus:

$$u^h(r) = \boldsymbol{N}_2^+(r)\hat{\boldsymbol{u}}_2 \tag{28}$$

Exactly the same expression is obtained using Eq. 24 in the limit as $p \to r$, establishing the continuity of the field across the curved edge of the fold.

Next, consider a point s in Fig. 11 on the straight edge of the fold. The field at point s can be obtained by evaluating the expression Eq. 26 in the limit as $q \to s$. By the definition of FEM shape functions,

$\boldsymbol{N}_1^-(s) = \boldsymbol{N}_2^+(s)$. Therefore the field at point s can be written as:

$$u^h(s) = \boldsymbol{N}_1^+(s)\hat{\boldsymbol{u}}_1 \tag{29}$$

Once again, this is exactly the expression obtained from Eq. 25 in the limit as $o \to s$, establishing the continuity of the field across the straight edge of the fold. *Thus the field defined by Eq. 19 is continuous within an implicitly tangled mesh.*

3.3.2. Rigid body

Now consider a special case where the field is a constant, i.e., $u = c$. To ensure that one can capture such constant fields exactly, we must prove that for any point q located inside the folded region $u^h(q) = c$.

From Eq 26, since $\hat{\boldsymbol{u}}_1 = \{c, c, c, c\}^\top$, $\hat{\boldsymbol{u}}_2 = \{c, c, c, c\}^\top$, $\boldsymbol{N}_1^+(q) = \left\{N_1^{1+}(q), N_1^{2+}(q), N_1^{3+}(q), N_1^{4+}(q)\right\}$, etc. we have:

$$u^h(q) = \left(\sum_{i=1}^{4}\left(N_1^{i+}(q) - N_1^{i-}(q) + N_2^{i+}(q)\right)\right)c \tag{30}$$

This can be re-grouped as:

$$u^h(q) = \left(\sum_{i=1}^{4} N_1^{i+}(q) - \sum_{i=1}^{4} N_1^{i-}(q) + \sum_{i=1}^{4} N_2^{i+}(q)\right)c$$

By the partition of unity property of shape functions, each summation equals to 1, i.e.,

$$u^h(q) = (1 - 1 + 1)c = c$$

Thus the field defined by Eq. 19 can capture a constant field exactly.

3.3.3. Constant strain

Without the loss of generality, consider a constant strain field $u = x$. The x-coordinates of the nodes of element E_1 are denoted by $\hat{\boldsymbol{x}}_1 = \left\{x^1, x^2, x^5, x^4\right\}^\top$ and that for element E_2 by $\hat{\boldsymbol{x}}_2 = \left\{x^2, x^3, x^4, x^5\right\}^\top$. Note that, here, the node numbers are denoted using superscripts while element numbers are denoted using subscripts. Assume that field at all vertices except the concave vertex is exactly equal to the x-coordinate at that point. From the equality condition (Eq. 23), it immediately follows that $u^5 = x^5$. Hence, we have $\hat{\boldsymbol{u}}_1 = \hat{\boldsymbol{x}}_1$ and $\hat{\boldsymbol{u}}_2 = \hat{\boldsymbol{x}}_2$.

Further, consider any point q inside the fold. From Eq. 26, we have

$$u^h(q) = \left(\boldsymbol{N}_1^+(q) - \boldsymbol{N}_1^-(q)\right)\hat{\boldsymbol{x}}_1 + \boldsymbol{N}_2^+(q)\hat{\boldsymbol{x}}_2 \tag{31}$$

From the isoparametric mapping of x (see Eq. 13), it follows that

$$u^h(q) = \boldsymbol{N}_1^+(q)\hat{\boldsymbol{x}}_1 - \boldsymbol{N}_1^-(q)\hat{\boldsymbol{x}}_1 + \boldsymbol{N}_2^+(q)\hat{\boldsymbol{x}}_2 = x^q - x^q + x^q \tag{32}$$

Thus, the field at any point q is the fold is $u^h(q) = x^q$. *Hence, the field defined by Eq. 19 along with equality condition (Eq. 23) can exactly capture a constant strain field.*

3.4. TFEM Assembly

Having established the theoretical properties of TFEM, we will now proceed to derive the underlying linear system of equations. In particular, to derive the stiffness matrix, we substitute the unambiguous definition of the field defined per Eq. 19 in the Galerkin formulation of Eq. 1 leading to:

$$\boldsymbol{K} = \int_{\Omega} \left[\nabla \sum_j \left(\boldsymbol{N}_j^+ - \boldsymbol{N}_j^- \right) \right]^\top \boldsymbol{D} \left[\nabla \sum_k \left(\boldsymbol{N}_k^+ - \boldsymbol{N}_k^- \right) \right] d\Omega \tag{33}$$

Regrouping:

$$\begin{aligned} \boldsymbol{K} = \int_{\Omega} \Bigg\{ & \left(\nabla \sum_j \left(\boldsymbol{N}_j^+ - \boldsymbol{N}_j^- \right) \right)^\top \boldsymbol{D} \left(\nabla \sum_j \left(\boldsymbol{N}_j^+ - \boldsymbol{N}_j^- \right) \right) \\ & + \left(\nabla \sum_j \left(\boldsymbol{N}_j^+ - \boldsymbol{N}_j^- \right) \right)^\top \boldsymbol{D} \left(\nabla \sum_{k \neq j} \left(\boldsymbol{N}_k^+ - \boldsymbol{N}_k^- \right) \right) \Bigg\} d\Omega \end{aligned} \tag{34}$$

Further, the first term of Eq. 34 can be expanded as

$$\sum_j \int_{C_j^+} \nabla \boldsymbol{N}_j^{+\top} \boldsymbol{D} \nabla \boldsymbol{N}_j^+ d\Omega + \sum_j \int_{C_j^-} \nabla \boldsymbol{N}_j^{-\top} \boldsymbol{D} \nabla \boldsymbol{N}_j^- d\Omega - \sum_j \int_{C_j^+ \cap C_j^-} \left(\nabla \boldsymbol{N}_j^{+\top} \boldsymbol{D} \nabla \boldsymbol{N}_j^- + \nabla \boldsymbol{N}_j^{-\top} \boldsymbol{D} \nabla \boldsymbol{N}_j^+ \right) d\Omega \tag{35}$$

Observe that the first term of Eq. 35 involves integration over C_j^+ with integrand containing $\nabla \boldsymbol{N}_j^+$ terms. Similarly the second term of Eq. 35 involves integration over C_j^- with integrand containing $\nabla \boldsymbol{N}_j^-$ terms. These two terms together form the standard stiffness matrix $\boldsymbol{K}^0$ as discussed later in Section 4. On the other hand, the third term of Eq. 35 involves integration over $C_j^+ \cap C_j^-$, i.e., over the fold, and the integrand involves both the terms $\nabla \boldsymbol{N}_j^+$ and $\nabla \boldsymbol{N}_j^-$. It captures the coupling between the components of the same element. We refer to the third term as $\boldsymbol{K}^S$, where the superscript S alludes to self-intersection.

Next the second term in Eq. 34 captures the coupling between components of different elements, and can be expanded as

$$\begin{aligned} \boldsymbol{K}^N = \sum_j \sum_{k \neq j} \Bigg(& \int_{C_j^+ \cap C_k^+} \nabla \boldsymbol{N}_j^{+\top} \boldsymbol{D} \nabla \boldsymbol{N}_k^+ d\Omega - \int_{C_j^- \cap C_k^+} \nabla \boldsymbol{N}_j^{-\top} \boldsymbol{D} \nabla \boldsymbol{N}_k^+ d\Omega - \\ & \int_{C_j^+ \cap C_k^-} \nabla \boldsymbol{N}_j^{+\top} \boldsymbol{D} \nabla \boldsymbol{N}_k^- d\Omega + \int_{C_j^- \cap C_k^-} \nabla \boldsymbol{N}_j^{-\top} \boldsymbol{D} \nabla \boldsymbol{N}_k^- d\Omega \Bigg) \end{aligned}$$

where N alludes to intersection between neighbors. Recall that $C_k^- = \emptyset$ for convex elements. In such cases, the terms involving C_k^- vanishes.

In summary, Eq. 33 can be written as:

$$\boldsymbol{K} = \boldsymbol{K}^0 + \boldsymbol{K}^S + \boldsymbol{K}^N \tag{36}$$

where,

$$\boldsymbol{K}^0 = \prod_{Assemble} \left(\int_{C_j^+} \nabla \boldsymbol{N}_j^{+\top} \boldsymbol{D} \nabla \boldsymbol{N}_j^+ d\Omega + \int_{C_j^-} \nabla \boldsymbol{N}_j^{-\top} \boldsymbol{D} \nabla \boldsymbol{N}_j^- d\Omega \right) \tag{37}$$

$$\boldsymbol{K}^S = \prod_{Assemble} \left(- \int_{C_j^+ \cap C_j^-} \nabla \boldsymbol{N}_j^{+\top} \boldsymbol{D} \nabla \boldsymbol{N}_j^- d\Omega \; - \int_{C_j^+ \cap C_j^-} \nabla \boldsymbol{N}_j^{-\top} \boldsymbol{D} \nabla \boldsymbol{N}_j^+ d\Omega \right) \tag{38}$$

$$\begin{aligned} \boldsymbol{K}^N = \prod_{Assemble} \Bigg(& \int_{C_j^+ \cap C_k^+} \nabla \boldsymbol{N}_j^{+\top} \boldsymbol{D} \nabla \boldsymbol{N}_k^+ d\Omega \; - \int_{C_j^- \cap C_k^+} \nabla \boldsymbol{N}_j^{-\top} \boldsymbol{D} \nabla \boldsymbol{N}_k^+ d\Omega \\ & - \int_{C_j^+ \cap C_k^-} \nabla \boldsymbol{N}_j^{+\top} \boldsymbol{D} \nabla \boldsymbol{N}_k^- d\Omega \; + \int_{C_j^- \cap C_k^-} \nabla \boldsymbol{N}_j^{-\top} \boldsymbol{D} \nabla \boldsymbol{N}_k^- d\Omega \Bigg) \end{aligned} \tag{39}$$

Similarly, substituting Eq. 19 on the right-hand side of the Galerkin formulation of Eq. 1, we get

$$\boldsymbol{f}^\theta = \int_\Omega \left(\sum_j \left(\boldsymbol{N}_j^+ - \boldsymbol{N}_j^- \right) \right)^\top f d\Omega = \prod_{Assemble} \int_{C_j^+} \boldsymbol{N}_j^{+\top} f d\Omega \; - \int_{C_j^-} \boldsymbol{N}_j^{-\top} f d\Omega \tag{40}$$

Finally, the equality condition defined in Eq. 23 when imposed at the re-entrant vertices of all the concave elements, will lead to a set constraint equations:

$$\tilde{\boldsymbol{C}} \hat{\boldsymbol{u}} = \boldsymbol{0} \tag{41}$$

where the number of rows in $\tilde{\boldsymbol{C}}$ is equal to the number of concave vertices. In summary, in TFEM we must solve the following linear system of equations:

$$\begin{bmatrix} \boldsymbol{K} & \tilde{\boldsymbol{C}}^\top \\ \tilde{\boldsymbol{C}} & \boldsymbol{0} \end{bmatrix} \begin{Bmatrix} \hat{\boldsymbol{u}} \\ \boldsymbol{\mu} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{f}^\theta \\ \boldsymbol{0} \end{Bmatrix}. \tag{42}$$

The Lagrange multipliers $\boldsymbol{\mu}$ are not used further in this work.

3.5. Special Cases

Eq. 42 provides a generalized framework for handling not only implicitly tangled meshes, but also non-tangled and explicitly tangled meshes.

3.5.1. Non-tangled mesh

First, consider the case where the mesh does not contain any tangled elements. All integrals containing negative components C_j^-, C_k^- vanish. This implies that there are no overlapping regions i.e., $C_j^+ \cap C_k^+ = \emptyset$, $C_j^+ \cap C_j^- = \emptyset$, and $C_j^+ \cap C_k^- = \emptyset$. Further C_j^+ and C_k^+ can be replaced simply by E_j and E_k respectively.

Moreover, $\nabla \boldsymbol{N}_j^+$ can be replaced by $\nabla \boldsymbol{N}_j$ while $\nabla \boldsymbol{N}_j^-$ is zero. Thus, the terms $\boldsymbol{K}^S$ and $\boldsymbol{K}^N$ vanish, while $\boldsymbol{K}^0$ and $\boldsymbol{f}^\theta$ terms reduce:

$$\boldsymbol{K}^0 = \prod_{Assemble} \int_{E_j} \nabla \boldsymbol{N}_j^\top \boldsymbol{D} \nabla \boldsymbol{N}_j d\Omega, \qquad \boldsymbol{f}^\theta = \prod_{Assemble} \int_{E_j} \boldsymbol{N}_j^\top f d\Omega$$

These equations correspond precisely to Eq. 4. Further, the constraint matrix $\tilde{\boldsymbol{C}}$ does not exist.

3.5.2. Explicitly tangled mesh

Next, consider the case where the quadrilateral mesh is only explicitly tangled, i.e. it has negatively oriented elements with no concave elements as in Fig. 5c. In this case, any element can either have a positive component or a negative component, but not both, i.e. $C_j^+ \cap C_j^- = \emptyset$, i.e., $\boldsymbol{K}^S$ vanishes. Further, since the elements are not divided into multiple components, we write $C_j^\pm$ simply as E_j, and the terms $\boldsymbol{N}_j^+$ and $-\boldsymbol{N}_j^-$ can be replaced by $\theta_j \boldsymbol{N}_j$ where θ_j either takes the value of +1 or -1. Finally, only one out of the four terms of $\boldsymbol{K}^N$ remains depending on the orientations of E_j and E_k. With these notations, Eq. 36 can be expressed as:

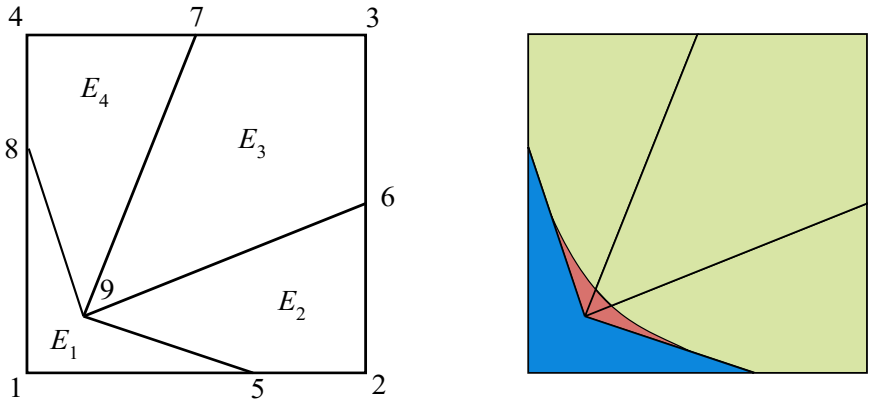
$$\boldsymbol{K}^0 = \prod_{Assemble} \int_{E_j} \nabla \boldsymbol{N}_j^\top \boldsymbol{D} \nabla \boldsymbol{N}_j d\Omega, \qquad \boldsymbol{f}^\theta = \prod_{Assemble} \int_{E_j} \theta_j \boldsymbol{N}_j^\top f d\Omega$$

$$\boldsymbol{K}^N = \prod_{Assemble} \int_{E_j \cap E_{k \neq j}} \theta_j \theta_k \nabla \boldsymbol{N}_j^\top \boldsymbol{D} \nabla \boldsymbol{N}_k d\Omega \qquad \boldsymbol{K}^S = \boldsymbol{0}$$

Thus we recover Eq. 11 for explicitly tangled meshes. Once again, the constraint matrix $\tilde{\boldsymbol{C}}$ does not play a role.

4. Numerical Implementation

In the previous section, we considered a two element mesh patch to explain the theory behind TFEM. In this section, we discuss the implementation of TFEM, using a more complex four element patch in Fig. 12, where the fold overlaps with multiple neighboring elements.



(a) (b)

Figure 12: Implicitly tangled mesh with element E_1 being concave.

4.1. Geometric Processing

The first step in the implementation of TFEM is the detection of concave elements. This amounts to computing and checking the four interior angles of every quad element. Thus, in Fig. 12, element E_1 is concave. Given a concave element E_j, the next step is to decompose its parametric space into two parts, corresponding to the two components C_j^+ and C_j^-; see Eq. 16. This decomposition can be carried out easily as follows. Let the element be defined by vertices $[(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)]$. Given the bilinear mapping, the Jacobian is given by

$$\boldsymbol{J} = \begin{bmatrix} x_{,\xi} & x_{,\eta} \\ y_{,\xi} & y_{,\eta} \end{bmatrix} = \frac{1}{4} \begin{bmatrix} \eta - 1 & 1 - \eta & 1 + \eta & -(1 + \eta) \\ \xi - 1 & -(1 + \xi) & 1 + \xi & 1 - \xi \end{bmatrix} \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \end{bmatrix}^T$$

The determinant simplifies to:

$$|\boldsymbol{J}| = \det(\boldsymbol{J}) = c_0 + c_1\xi + c_2\eta \tag{43}$$

where,

$$\begin{aligned} c_0 &= [(x_1 - x_3)(y_2 - y_4) - (x_2 - x_4)(y_1 - y_3)]/8 \\ c_1 &= [(x_3 - x_4)(y_1 - y_2) - (x_1 - x_2)(y_3 - y_4)]/8 \\ c_2 &= [(x_2 - x_3)(y_1 - y_4) - (x_1 - x_4)(y_2 - y_3)]/8 \end{aligned}$$

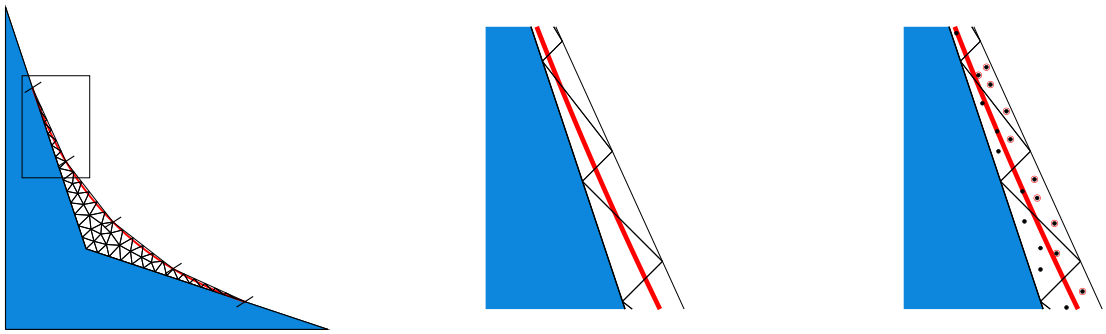
We conclude that $|\boldsymbol{J}| = 0$ is a straight line in the parametric space, defined by the constants c_0, c_1 and c_2. Thus, the parametric space is decomposed into two parts on opposite sides of the $|\boldsymbol{J}| = 0$ line.

Next, we consider all elements E_k that share the re-entrant vertex with E_j. For example, in Fig. 12 three elements E_2, E_3 and E_4 share the re-entrant vertex 9 with element E_1. Therefore, elements E_2, E_3 and E_4 intersect with different parts of the fold. Consider, for example, the intersection of element E_4 with the fold. Observe that boundary of the intersection is made up of three edges: two of them are straight lines in the physical space, while the third is a curve obtained by mapping the $|\boldsymbol{J}| = 0$ line. Thus for every

neighboring element E_k intersecting the fold with E_j, we determine the boundary of the intersection region. This intersection region must be triangulated for numerical integration.

4.2. Triangulating the Folded Region

To triangulate the folded region, the $|\boldsymbol{J}| = 0$ curve has to be approximated by a finite number of line segments. Sufficient number of segments is needed to ensure that all quadrature points lie within the fold. To illustrate, let the $|\boldsymbol{J}| = 0$ curve (highlighted) be approximated by four line segments, and the polygonal fold triangulated, as in Fig. 13a. Observe that the integration points of certain triangles lie outside the folded region, as seen in Fig. 13c. This will lead to erroneous results, including non-invertible Jacobians.



(a) (b) (c)

Figure 13: Approximation of $|\boldsymbol{J}| = 0$ curve using four segments: (a) Triangulation of the folded region, (b) Zoomed-in view of region of interest (c) Integration points marked which lie outside the fold are marked with a circle.

To avoid these issues, sufficient number of segments must be used to ensure that all the integration points lie within the folded region. We adopt a simple iterative scheme by doubling the number of segments until all the integration points lie within the folded region. For example, with 128 segments, it is easy to confirm for this case that all the integration points lie within the folded region. One may also employ an adaptive procedure to sub-divide the segments. An alternate approach (but not pursued here) is to approximate the fold by a non-convex polygon, and use direct integration techniques over such non-convex polygons; see [25, 26, 27] for example.

4.3. Computing $\boldsymbol{K}^S$

Now consider computing $\boldsymbol{K}^S$ matrix in Eq. 38, where the superscript S refers to self-intersection. $\boldsymbol{K}^S$ is comprised of contributions from all concave elements within a mesh. A typical contribution $\boldsymbol{k}_j^S$ from a concave element E_j is given by :

$$\boldsymbol{k}_j^S = -\int\limits_{C_j^+ \cap C_j^-} \left(\nabla \boldsymbol{N}_j^{+\top} \boldsymbol{D} \nabla \boldsymbol{N}_j^- + \nabla \boldsymbol{N}_j^{-\top} \boldsymbol{D} \nabla \boldsymbol{N}_j^+ \right) d\Omega$$

Given the triangulation discussed previously, we have:

$$\boldsymbol{k}_j^S = \sum_{triangles} - \int\limits_{triangle} \left(\nabla \boldsymbol{N}_j^{+\top} \boldsymbol{D} \nabla \boldsymbol{N}_j^- + \nabla \boldsymbol{N}_j^{-\top} \boldsymbol{D} \nabla \boldsymbol{N}_j^+ \right) dxdy$$

As is typical, each triangle is mapped to a standard triangle in (γ, ζ) space. Let $|\boldsymbol{J}_t|$ be the determinant of Jacobian associated with this triangle mapping. Thus, we have:

$$\boldsymbol{k}_j^S = \sum_{triangles} -\int_{\zeta}\int_{1-\gamma} \left(\nabla \boldsymbol{N}_j^{+\top} \boldsymbol{D} \nabla \boldsymbol{N}_j^- + \nabla \boldsymbol{N}_j^{-\top} \boldsymbol{D} \nabla \boldsymbol{N}_j^+\right) |\boldsymbol{J}_t| d\gamma d\zeta$$

Consider a quadrature point (γ_q, ζ_q) of the standard triangle with weight w_q. The corresponding point (x_q, y_q) in the physical space belongs to both components C_j^+ and C_j^-. Let (ξ_j^+, η_j^+) and (ξ_j^-, η_j^-) be the respective coordinates in the parametric space (numerically determined via Newton-Raphson algorithm). For brevity, the subscript q has been dropped. We compute the Jacobian matrices $\left(\boldsymbol{J}_j^+, \boldsymbol{J}_j^-\right)$ associated with the quad parametric mapping at these quadrature points. Let

$$\boldsymbol{B}_j^+ = (\boldsymbol{J}_j^+)^{-1} \nabla_{\xi\eta} \boldsymbol{N}_j^+(\xi_j^+, \eta_j^+)$$

and

$$\boldsymbol{B}_j^- = (\boldsymbol{J}_j^-)^{-1} \nabla_{\xi\eta} \boldsymbol{N}_j^-(\xi_j^-, \eta_j^-)$$

Summing the contribution from all triangles results in:

$$\boldsymbol{k}_j^S = \sum_{triangles} -\sum_q \left(\boldsymbol{B}_j^{+\top} \boldsymbol{D} \boldsymbol{B}_j^- + \boldsymbol{B}_j^{-\top} \boldsymbol{D} \boldsymbol{B}_j^+\right) |\boldsymbol{J}_t| w_q \tag{44}$$

The sub-matrices $\boldsymbol{k}_j^S$ from each concave element are then assembled to form $\boldsymbol{K}^S$.

4.4. Computing $\boldsymbol{K}^N$

Next, consider computing $\boldsymbol{K}^N$ matrix in Eq. 39 where the superscript N refers to overlap between neighboring elements. $\boldsymbol{K}^N$ is comprised of contributions from pairs of neighboring elements within a mesh, where one or both of the elements are concave. A typical sub-matrix $\boldsymbol{k}_{jk}^N$ due to an overlap of neighboring elements E_j and E_k is given by:

$$\begin{aligned}\boldsymbol{k}_{jk}^N = &\int_{C_j^+ \cap C_k^+} \nabla \boldsymbol{N}_j^{+\top} \boldsymbol{D} \nabla \boldsymbol{N}_k^+ d\Omega - \int_{C_j^- \cap C_k^+} \nabla \boldsymbol{N}_j^{-\top} \boldsymbol{D} \nabla \boldsymbol{N}_k^+ d\Omega \\ &- \int_{C_j^+ \cap C_k^-} \nabla \boldsymbol{N}_j^{+\top} \boldsymbol{D} \nabla \boldsymbol{N}_k^- d\Omega + \int_{C_j^- \cap C_k^-} \nabla \boldsymbol{N}_j^{-\top} \boldsymbol{D} \nabla \boldsymbol{N}_k^- d\Omega\end{aligned}$$

Observe that not all terms need to be considered depending on the concavity of the two elements. For example, if element E_k is convex, then the third and fourth term can be disregarded since C_k^- does not exist.

To illustrate the computation of these terms, consider:

$$\int_{C_j^+ \cap C_k^+} \nabla \boldsymbol{N}_j^{+\top} \boldsymbol{D} \nabla \boldsymbol{N}_k^+ d\Omega$$

We must rely on the triangulation of the region $C_j^+ \cap C_k^+$. For example, Fig. 14 illustrates the triangulation of $C_1^+ \cap C_4^+$ involving elements E_1 and E_4.

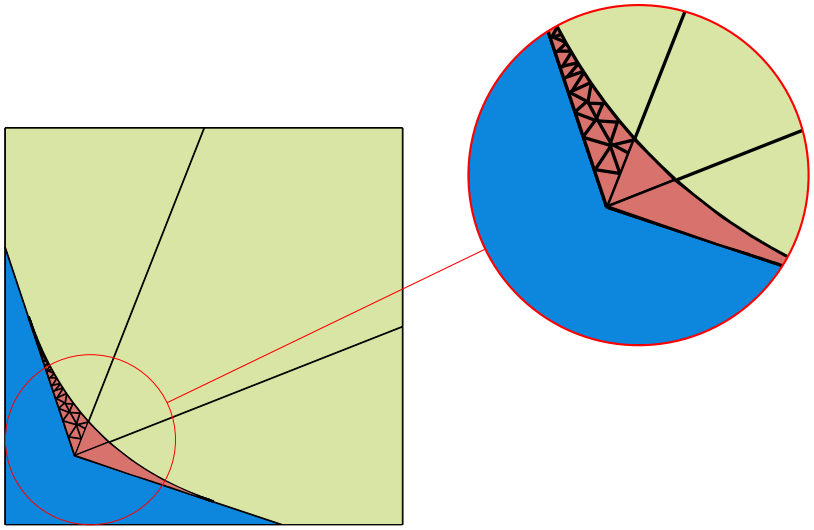


Figure 14: Triangulation of the intersection region of the components C_1^+ and C_4^+.

As before, consider a quadrature point $(x_q,\ y_q)$ within a triangle. The corresponding points in the parametric space $(\xi_j^+,\ \eta_j^+)$ and $(\xi_k^+,\ \eta_k^+)$ are determined (via Newton-Raphson). Once these points are computed, gradients of the respective shape functions $\left(\nabla_{\xi\eta}\boldsymbol{N}_j^+,\ \nabla_{\xi\eta}\boldsymbol{N}_k^+\right)$ and the Jacobian matrices $\left(\boldsymbol{J}_j^+, \boldsymbol{J}_k^+\right)$ associated with the parametric mapping to the standard quad can be determined at these points. Let

$$\boldsymbol{B}_j^+ = (\boldsymbol{J}_j^+)^{-1}\nabla_{\xi\eta}\boldsymbol{N}_j^+(\xi_j^+, \eta_j^+)$$

and

$$\boldsymbol{B}_k^+ = (\boldsymbol{J}_k^+)^{-1}\nabla_{\xi\eta}\boldsymbol{N}_k^+(\xi_k^+, \eta_k^+)$$

The integration can be obtained by summing the contribution from all triangles:

$$\int\limits_{C_j^+\cap C_k^+} \nabla\boldsymbol{N}_j^{+\top}\nabla\boldsymbol{N}_k^+ d\Omega = \sum_{triangles}\sum_{q}\left(\boldsymbol{B}_j^+\right)^\top \boldsymbol{D}\left(\boldsymbol{B}_k^+\right)|\boldsymbol{J}_t|w_q \tag{45}$$

Similarly, all other terms can be evaluated to compute $\boldsymbol{k}_{jk}^N$, and assembled to form $\boldsymbol{K}^N$.

4.5. Computing $\boldsymbol{K}^0$

We will now consider computing $\boldsymbol{K}^0$ in Eq. 37, where $\boldsymbol{K}^0$ is comprised of contributions from all elements within a mesh. A typical contribution $\boldsymbol{k}_j^0$ from element E_j is given by:

$$\boldsymbol{k}_j^0 = \int\limits_{C_j^-} \nabla\boldsymbol{N}_j^{-\top}\boldsymbol{D}\nabla\boldsymbol{N}_j^- d\Omega + \int\limits_{C_j^+} \nabla\boldsymbol{N}_j^{+\top}\boldsymbol{D}\nabla\boldsymbol{N}_j^+ d\Omega$$

If $C_j^- = \emptyset$, this reduces to standard numerical integration over the parametric space. On the other hand, when $C_j^- \neq \emptyset$, we must exploit the triangulation of the fold C_j^- and C_j^+ as discussed earlier, and illustrated again in Fig. 15a.

The triangulation for C_j^+ consists of two parts: the triangulation over the fold, and the triangulation over the non-folded region as in Fig. 15b. Note that these two triangulations need not be conforming since they are used primarily for integration. Once the regions are triangulated, the two terms can be computed as:

$$\boldsymbol{k}_j^0 = \sum_{\text{triangles in } C_j^-} \sum_q \left(\boldsymbol{B}_j^-\right)^\top \boldsymbol{D}\left(\boldsymbol{B}_j^-\right) |\boldsymbol{J}_t| w_q + \sum_{\text{triangles in } C_j^+} \sum_q \left(\boldsymbol{B}_j^+\right)^\top \boldsymbol{D}\left(\boldsymbol{B}_j^+\right) |\boldsymbol{J}_t| w_q \tag{46}$$

The sub-matrices $\boldsymbol{k}_j^0$ are then assembled to form $\boldsymbol{K}^0$.

(a) (b)

Figure 15: Triangulation of the (a) C_j^- region and (b) C_j^+ region.

4.6. Computing $\boldsymbol{f}^\theta$

Finally, the forcing term $\boldsymbol{f}^\theta$ is computed similar to classical FEM with two modifications – integration takes place over each component, accounting for the orientation of the component. The latter is accomplished by subtracting the contribution from the negatively oriented component C_j^- before assembling.

$$\boldsymbol{f}^\theta = \prod_{Assemble} \int_{C_j^+} \boldsymbol{N}_j^{+\top} f d\Omega \; - \int_{C_j^-} \boldsymbol{N}_j^{-\top} f d\Omega$$

Similar to $\boldsymbol{K}^0$ computation, the integration over C_j^+ and C_j^- regions can be performed by employing the triangulation in Fig. 15.

4.7. Solving and Post-processing

Once all the terms in Eq. 42 are computed, the nodal values $\hat{\boldsymbol{u}}$ can be determined. Next, the field at any point can be computed as follows. For points *outside* the fold, the field is computed similar to FEM. For points *inside* the fold, contributions from each of the components have to be considered, as in discussed in Section 3.2.

Further, the gradient of the field ∇u^h can be computed easily, as follows. Consider Eq. 19 which represents the field at any point p. Using the chain rule:

$$\begin{Bmatrix} \frac{\partial u^h}{\partial \xi} \\ \frac{\partial u^h}{\partial \eta} \end{Bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \begin{Bmatrix} \frac{\partial u^h}{\partial x} \\ \frac{\partial u^h}{\partial y} \end{Bmatrix} \tag{47}$$

On rearranging, we get:

$$\begin{Bmatrix} \frac{\partial u^h}{\partial x} \\ \frac{\partial u^h}{\partial y} \end{Bmatrix} = \boldsymbol{J}^{-1} \begin{Bmatrix} \frac{\partial u^h}{\partial \xi} \\ \frac{\partial u^h}{\partial \eta} \end{Bmatrix} \tag{48}$$

where:

$$\frac{\partial u^h}{\partial \xi} = \sum_{j|p\in C_j^+} \frac{\partial \boldsymbol{N}_j^+}{\partial \xi}\bigg|_p \hat{\boldsymbol{u}}_j - \sum_{j|p\in C_j^-} \frac{\partial \boldsymbol{N}_j^-}{\partial \xi}\bigg|_p \hat{\boldsymbol{u}}_j \tag{49}$$

$$\frac{\partial u^h}{\partial \eta} = \sum_{j|p\in C_j^+} \frac{\partial \boldsymbol{N}_j^+}{\partial \eta}\bigg|_p \hat{\boldsymbol{u}}_j - \sum_{j|p\in C_j^-} \frac{\partial \boldsymbol{N}_j^-}{\partial \eta}\bigg|_p \hat{\boldsymbol{u}}_j \tag{50}$$

These equations are analogous to Eq. 19, except that the shape functions are now replaced by their derivatives. Thus the basic rules of TFEM post-processing apply.

5. Numerical Experiments

In this section, we demonstrate the proposed method using numerical experiments. We consider a Poisson problem over a domain Ω bounded by $\partial\Omega$:

$$-\nabla^2 u(x,y) = f(x,y) \text{ in } \Omega \tag{51a}$$

$$u = g(x,y) \text{ on } \partial\Omega_d \tag{51b}$$

$$\nabla u \cdot \bar{\boldsymbol{n}} = h(x,y) \text{ on } \partial\Omega_n \tag{51c}$$

The problem is solved over various implicitly tangled meshes and boundary conditions. Numerical experiments are conducted under the following conditions:

- The implementation is in MATLAB R2020b, on a standard Windows 10 desktop with Intel(R) Core(TM) i9-9820X CPU running at 3.3 GHz with 16 GB memory.
- The number of quadrature points for convex quadrilateral elements is 16.
- Triangulation of the folded region is performed by employing MATLAB's inbuilt mesher - `generateMesh`. The number of quadrature points for triangles is 4.

The questions being investigated through the experiments are:

- **Accuracy**: How does the accuracy of TFEM compare against that of the FEM? Does the accuracy of TFEM and FEM depend on the extent of tangling? To measure accuracy, we consider the L_2 and/or energy errors; the L_2 error is defined as:

$$||u - u^h||_{L_2(\Omega)} = \sqrt{\int_\Omega (u - u^h)^2 \, d\Omega} \tag{52}$$

and the energy norm is defined as:

$$||e_h||_{E(\Omega)} = ||u - u^h||_{E(\Omega)} = \sqrt{\int_\Omega (\nabla u - \nabla u^h)^\top (\nabla u - \nabla u^h) \, d\Omega} \tag{53}$$

where u and u^h are the exact and computed solutions respectively.

- **Condition Number**: The condition number is a measure of how close a matrix is to being singular [28]; it is desirable to have a condition number close to unity. Here, we investigate the condition numbers of FEM and TFEM linear systems. We also investigate if the condition numbers depend on the extent of tangling. We employ MATLAB's built-in function `condest` to compute the 1-norm condition number.

- **Convergence**: Here we investigate if TFEM converges as the element size is decreased.

- **Computational Cost**: Finally, we investigate the computational overhead of TFEM for a sample problem.

5.1. Patch Tests

5.1.1. Two element mesh

In the first experiment, we consider a square domain $\Omega \in (0, 1)^2$ with the field $u = x$ as the exact solution; the corresponding Dirichlet boundary condition is applied on the left edge, and Neumann conditions on the remaining three edges. The domain is divided into two quads, one of which is concave as in Fig. 16. In this case, the folded region is shared by a single neighboring element. To vary the extent of tangling, the position of vertex 5 is defined as in Fig. 16, where $d \in (0, 0.5)$. Larger the value of d, greater is the extent of tangling.

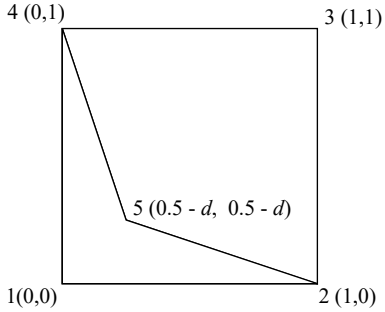


Figure 16: Two element mesh with implicit tangling.

Fig. 17 illustrates the L_2 errors in FEM and TFEM, for varying extent of tangling. As one can observe, TFEM leads to machine precision accuracy for all the values of d, while FEM is erroneous. In TFEM, due to the equality condition, there is one additional degree of freedom compared to FEM. This leads to slightly higher condition number in TFEM, but is well within acceptable range.

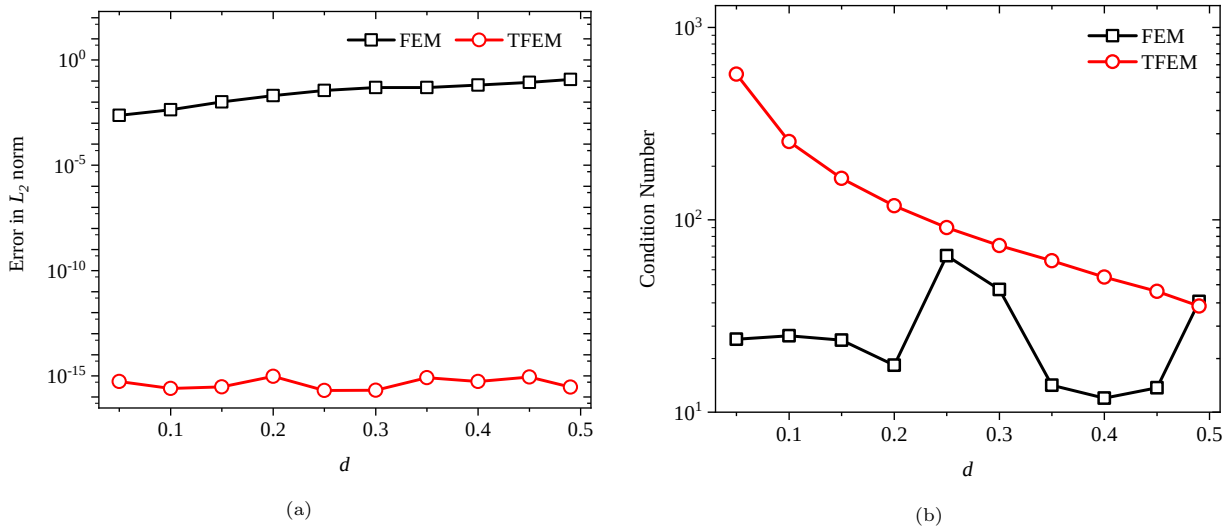


Figure 17: Comparison of TFEM and FEM for two-element mesh: (a) L_2 error vs. d, (b) condition number vs. d.

Fig. 18 illustrates the post-processed results for $d = 0.4$ for FEM and TFEM, where the domain is triangulated and field at the nodes of all the triangles is computed by the procedure discussed in Section 4.7; Visit 3.0.2 [29] was used for visualizing the results. As can one observe in Fig. 18, TFEM leads to a smooth linear field, as opposed to FEM.

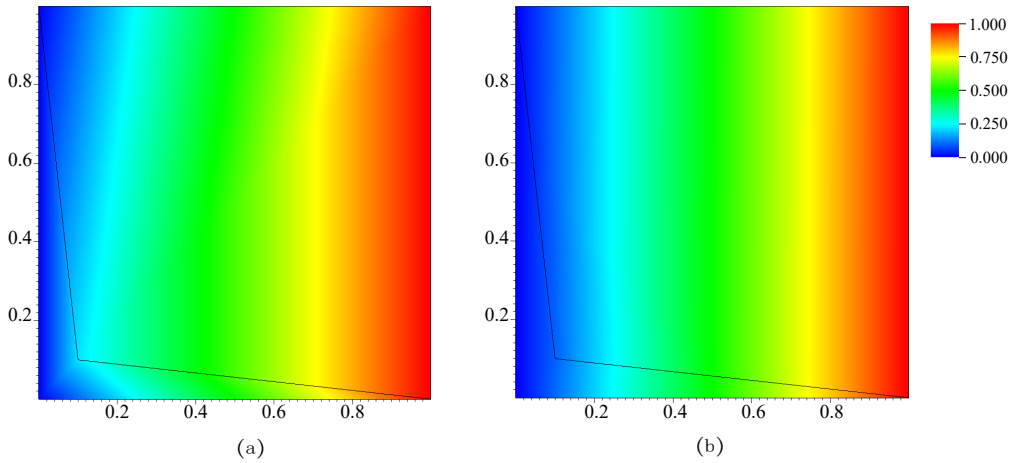


Figure 18: Post-processed solution obtained by (a) FEM (b) TFEM.

5.1.2. Four element mesh

Next, we consider a square domain $\Omega \in (0,1)^2$ with a linear field $u = 0.579x + 0.246y - 0.374$ as the exact solution. Dirichlet condition is imposed on the left edge, while Neumann conditions are imposed on the remaining three edges. The domain is discretized into 4 quadrilateral elements, one of which is concave as in Fig. 19. The folded region is shared by three neighboring convex elements. To introduce asymmetry, we move vertex 9 along an arc of a circle as illustrated, where α varies from 15° to 75°, and radius $r = 0.125\sqrt{2}$.

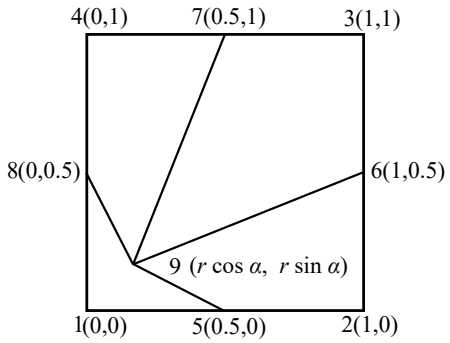


Figure 19: Implicitly tangled mesh with four elements.

Fig. 20a illustrates the L_2 errors in FEM and TFEM, while Fig. 20b compares the condition numbers. It is reassuring to note that the TFEM leads to machine precision accuracy, without a significant increase in condition number.

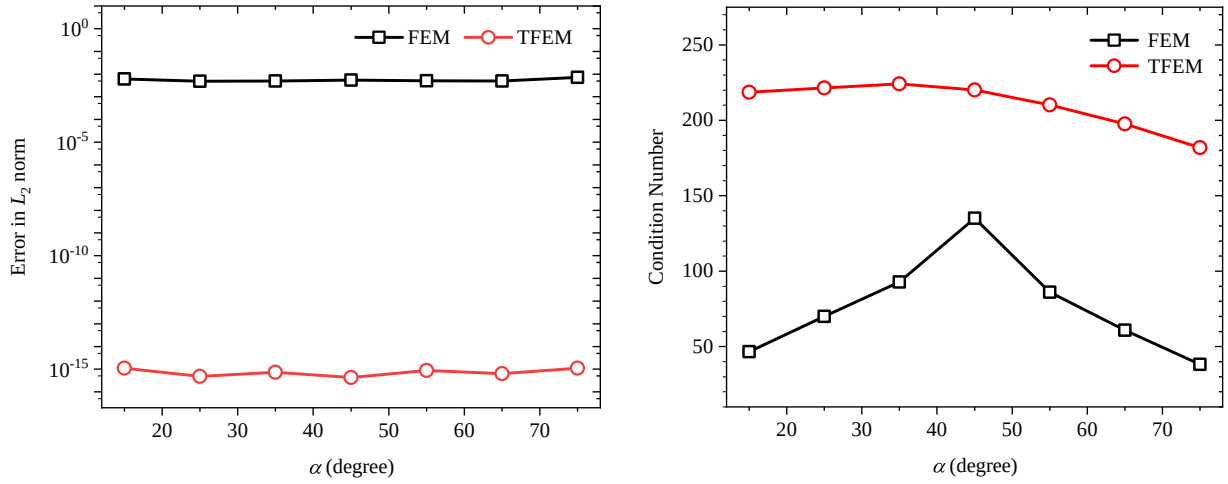


Figure 20: Comparison of TFEM and FEM for four-element mesh: (a) Error vs. α (b) Condition number vs. α

5.1.3. L-shaped domain

Next consider the mesh in Fig. 21 for the L-shaped domain. The concave element E_2 is on the boundary, and the folded region is not shared by any neighboring element. With the exact solution as $u = 0.579x + 0.246y - 0.374$, we apply Dirichlet condition on the bottom edge and Neumann on the remaining 5 edges.

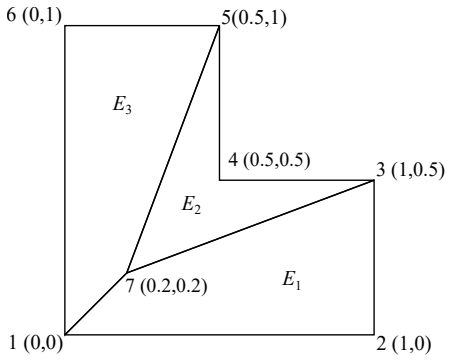


Figure 21: L-shaped domain discretized by implicitly tangled mesh.

Table 1 compares the performance of FEM and TFEM for the mesh. As in the previous experiments, we observe that TFEM captures the field exactly with a comparable condition number.

Table 1: Comparing FEM and TFEM for the mesh in Fig. 21.

Method	FEM	TFEM
L_2 error	5.6×10^{-3}	1.4×10^{-16}
Condition number	38.91	30.39

5.2. Convergence Study

While the previous experiments confirmed that TFEM satisfies various patch tests, we now investigate the convergence of TFEM when the exact solution lies outside the span of the finite element space. We consider an example from [30] where boundary conditions and heat source are such that the exact solution is $u = (1-x^2)(1-y^2)$. The domain is a unit square with Dirichlet boundary conditions. Various meshes are constructed by using the two-element mesh (see Fig. 16) with $d = 0.4$ as the basic repeating unit. Fig. 22 illustrates a few sample meshes used for the convergence study; the basic units are scaled to conform to the unit square domain.

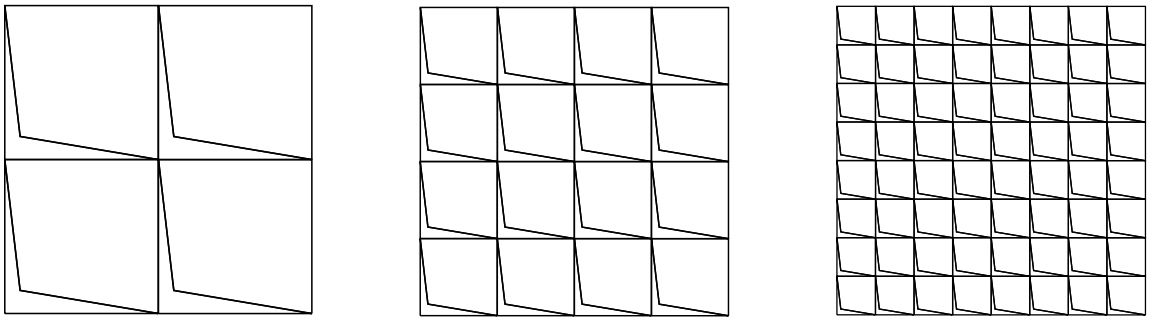


Figure 22: Meshes used for convergence study.

The L_2 and energy norm errors for FEM and TFEM, as a function of mesh size (h), are illustrated in

Fig. 23. The convergence rates for the TFEM L_2 and energy norm errors are 1 and 2 respectively which is consistent with expectations for second-order elliptic PDEs with linearly complete approximations [30].

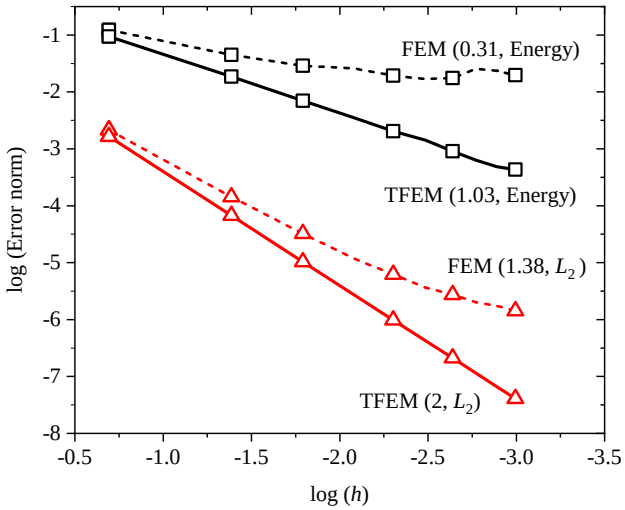


Figure 23: L_2 and energy norms errors as a function of mesh size h for the meshes in Fig. 22.

5.3. Applications

5.3.1. Mesh morphing

To illustrate potential applications, we consider mesh morphing with implicit tangling. Mesh morphing is essentially a mesh update where the topology is maintained while the mesh nodes are re-located according to a specified rule [14]. It enables rapid simulation of geometric configurations. As an example, consider a domain with a 'plus' shaped void at the center as in Fig. 24a, with the initial mesh as in Fig. 24b.

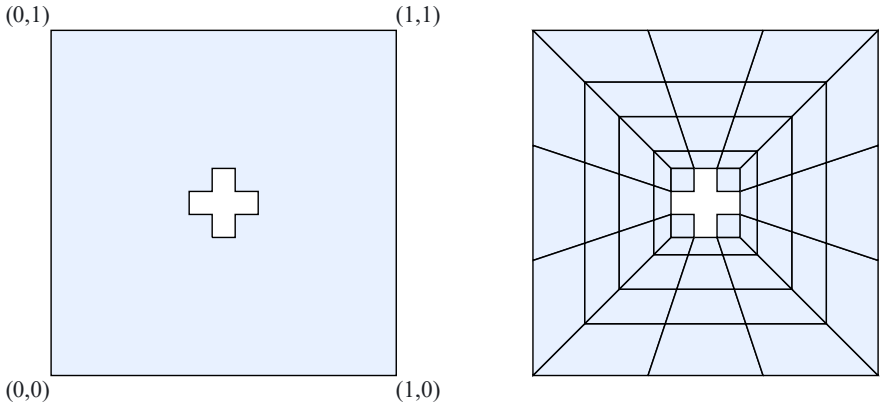


Figure 24: Mesh morphing application: (a) Domain with a void, and (b) initial quad mesh.

The inner void is now rotated in a counterclockwise direction by an angle β, and the mesh is morphed as follows. Nodes except the outer boundary nodes are rotated about the center such that the angle of rotation

exponentially increase from 0° to β as nodes get closer to the void. Fig. 25 illustrates the morphed mesh for $\beta = 70^\circ$. Observe that some of the quads (shown in red) are concave, i.e., the mesh is implicitly tangled.

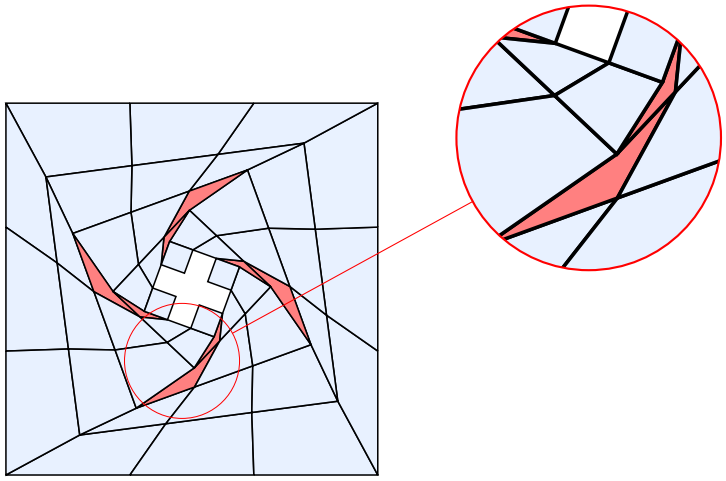


Figure 25: An implicitly tangled morphed mesh.

We carry out two experiments for this example. First, we apply Dirichlet boundary conditions corresponding to $u(x, y) = 0.579x + 0.246y - 0.374$ over the entire boundary, with $f(x, y) = 0$. The L_2 errors for various values of β are illustrated in Fig 26a. For $\beta < 60^\circ$, i.e., when there is no tangling, both FEM and TFEM are able to capture the field exactly. However, for $\beta > 50^\circ$, FEM error grows rapidly, while TFEM error remains close to machine precision. The two condition numbers are illustrated (see Fig. 26b).

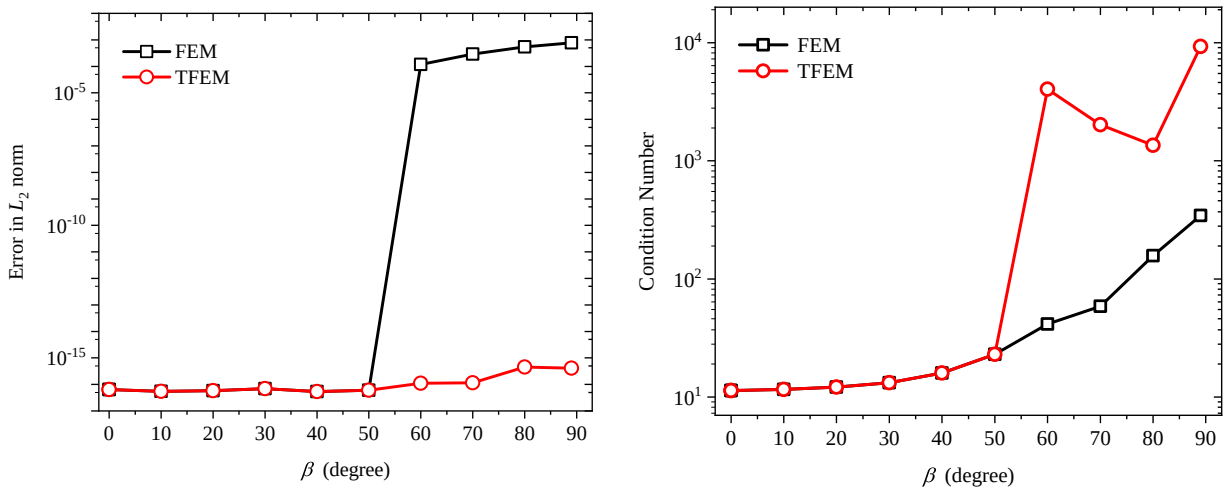


Figure 26: Comparison of TFEM and FEM for four-element mesh: (a) L_2 error vs. β, (b) condition number vs. β.

Next, we set $f(x, y) = 1$ and zero Dirichlet condition is applied on the entire boundary. Fig. 27 illustrates the post-processed results for $\beta = 0^\circ,\ 40^\circ$, and 70°.

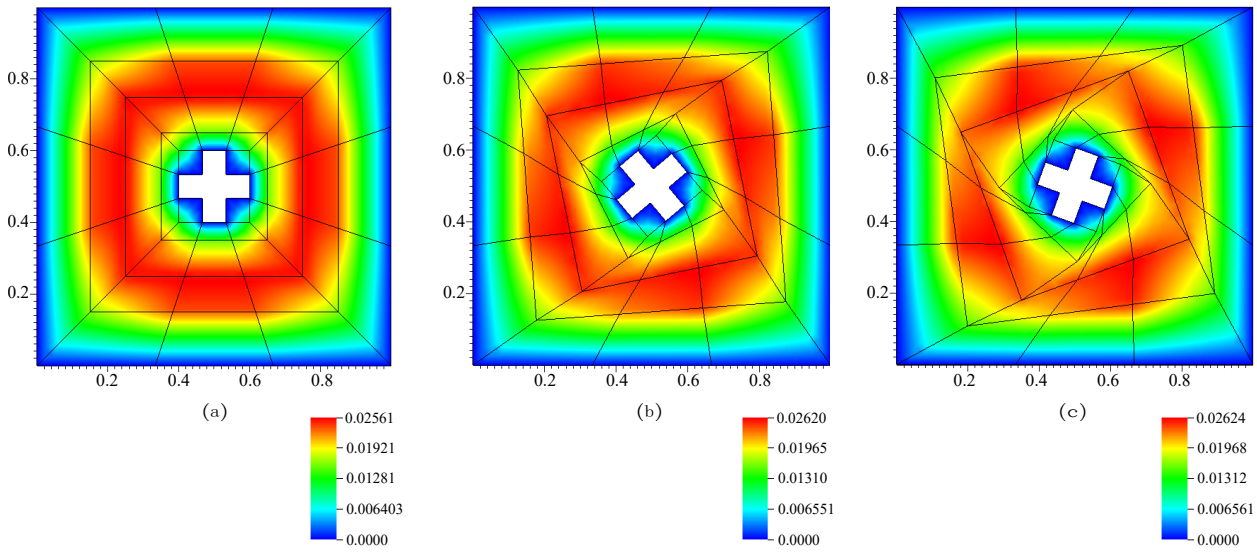


Figure 27: Post-processed solution using TFEM for $\beta =$ (a) 0°, (b) 40° and (c) 70°.

5.3.2. Aircraft model

We now consider an example where tangling can occur during meshing in practice. The mesh in Fig. 28a was generated using the quad mesher proposed in [31]; it can be observed that one quad element (among a total of around 600 elements) is concave, and the folded region is shared by two neighboring elements. Here we solve a Poisson problem with $f(x, y) = 10$, with zero Dirichlet conditions on all edges. The TFEM post-processed solution is illustrated in Fig. 28b.

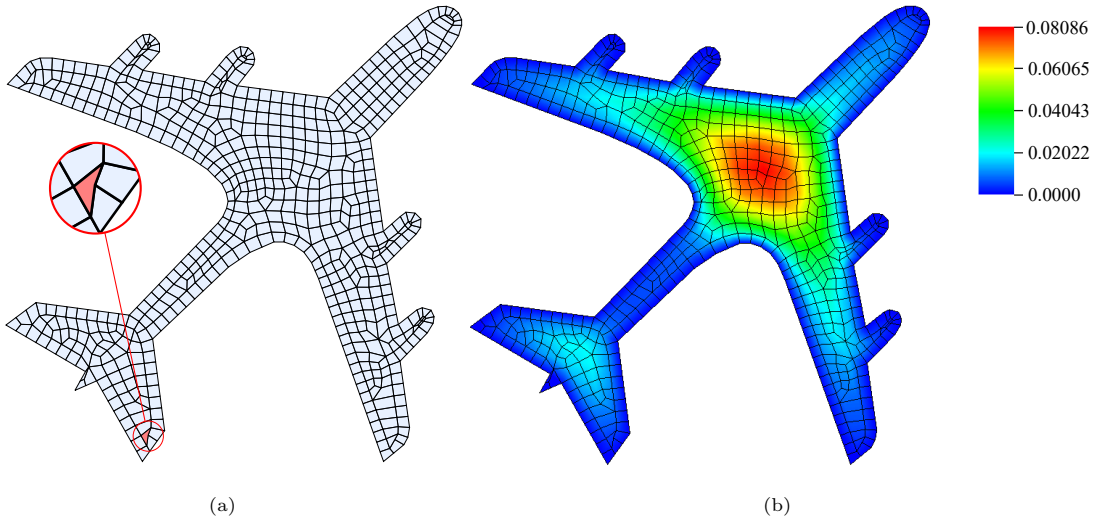


Figure 28: (a) Mesh for an aircraft model, with one concave element.(b) TFEM solution to a Poisson problem.

FEM took 0.37 seconds to solve the problem (albeit incorrectly), while TFEM requires around 1.4 seconds (post-processing was not included in either case). Fig. 29 illustrates the time distribution among various TFEM steps (time taken to solve the linear system was negligible). The TFEM time can potentially be

reduced by: (1) using a compiled language such as C, rather than MATLAB, (2) by developing more efficient integration techniques, (3) reducing/optimizing the number of triangles in the folded region, and (4) parallelization.

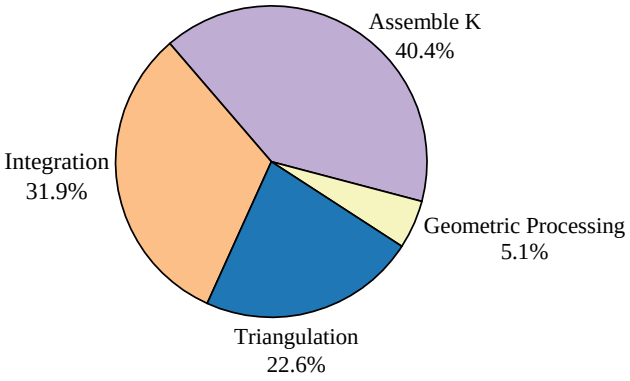


Figure 29: Time distribution for various TFEM steps for the aircraft model.

6. Conclusions

In FEM, a high quality mesh without tangles is always preferable over a tangled mesh. However, in scenarios where tangling is unavoidable, the proposed tangled finite element method (TFEM) not only leads to accurate results, but is also theoretically justifiable. Further, the proposed TFEM method only entails adding certain correction terms to standard FEM, and can therefore be implemented in commercial FEM packages.

Certain drawbacks and challenges in TFEM are to be noted. First, it was emphasized that in TFEM, singularities in the Jacobian can arise if proper care is not taken during numerical integration. Second, book-keeping can be challenging if there are a large number of tangled elements; fortunately, this problem rarely arises in practice. Finally, as one can expect, there is a computational overhead in handling the tangled elements in TFEM; efficient implementations can minimize this overhead.

Several extensions of TFEM are possible. The most significant is the 3D extension to implicitly tangled hexahedral meshes [32, 33]. Another possible extension is handling of curved elements where implicit tangling is fairly common [34, 35, 36, 37]. It is also of significant theoretical and practical interest to extend the ideas behind TFEM to iso-geometric analysis [38, 39] where, once again, implicit tangling can occur [40, 41, 42, 43], and lead to erroneous results.

Replication of Results

The MATLAB code for TFEM is available at www.ersl.wisc.edu/software/TFEM2DQuad.zip. The authors have incorporated MATLAB functions [44], [45] in this work.

Acknowledgments

The authors would like to thank the support of National Science Foundation through grant 1715970. Prof. Krishnan Suresh also serves as a consulting Chief Scientific Officer for SciArt, Corp.

References

[1] T. J. Hughes, The finite element method: linear static and dynamic finite element analysis, Courier Corporation, 2012.

[2] O. C. Zienkiewicz, R. L. Taylor, J. Z. Zhu, The finite element method: its basis and fundamentals, Elsevier, 2005.

[3] O. Axelsson, V. A. Barker, Finite element solution of boundary value problems: theory and computation, SIAM, 2001.

[4] J. Danczyk, K. Suresh, Finite element analysis over tangled simplicial meshes: Theory and implementation, Finite Elements in Analysis and Design 70 (2013) 57–67.

[5] J. Danczyk, K. Suresh, Finite element analysis over tangled meshes, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Vol. 45011, Citeseer, 2012, pp. 89–95.

[6] C. S. Verma, K. Suresh, Towards fea over tangled quads, Procedia Engineering 82 (2014) 187–199.

[7] M. Livesu, A. Sheffer, N. Vining, M. Tarini, Practical hex-mesh optimization via edge-cone rectification, ACM Transactions on Graphics (TOG) 34 (4) (2015) 1–11.

[8] P. M. Knupp, A method for hexahedral mesh shape optimization, International journal for numerical methods in engineering 58 (2) (2003) 319–332.

[9] L. A. Freitag, P. Plassmann, Local optimization-based simplicial mesh untangling and improvement, International Journal for Numerical Methods in Engineering 49 (1-2) (2000) 109–125.

[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, International Journal for numerical methods in engineering 48 (8) (2000) 1165–1185.

[11] J. M. Escobar, E. Rodrıguez, R. Montenegro, G. Montero, J. M. González-Yuste, Simultaneous untangling and smoothing of tetrahedral meshes, Computer Methods in Applied Mechanics and Engineering 192 (25) (2003) 2775–2787.

[12] P. M. Knupp, Hexahedral and tetrahedral mesh untangling, Engineering with Computers 17 (3) (2001) 261–268.

[13] P. Vachal, R. V. Garimella, M. J. Shashkov, Untangling of 2d meshes in ale simulations, Journal of Computational Physics 196 (2) (2004) 627–644.

[14] M. L. Staten, S. J. Owen, S. M. Shontz, A. G. Salinger, T. S. Coffey, A comparison of mesh morphing methods for 3d shape optimization, in: Proceedings of the 20th international meshing roundtable, Springer, 2011, pp. 293–311.

[15] G. Irving, J. Teran, R. Fedkiw, Invertible finite elements for robust simulation of large deformation, in: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation, 2004, pp. 131–140.

[16] A. Stomakhin, R. Howes, C. Schroeder, J. M. Teran, Energetically consistent invertible elasticity, in: Proceedings of the 11th ACM SIGGRAPH/Eurographics conference on Computer Animation, 2012, pp. 25–32.

[17] G. Liu, K. Dai, T. T. Nguyen, A smoothed finite element method for mechanics problems, Computational Mechanics 39 (6) (2007) 859–877.

[18] G. Manzini, A. Russo, N. Sukumar, New perspectives on polygonal and polyhedral finite element methods, Mathematical Models and Methods in Applied Sciences 24 (08) (2014) 1665–1699.

[19] H. Chi, C. Talischi, O. Lopez-Pamies, G. H. Paulino, Polygonal finite elements for finite elasticity, International Journal for Numerical Methods in Engineering 101 (4) (2015) 305–328.

[20] S. Martin, P. Kaufmann, M. Botsch, M. Wicke, M. Gross, Polyhedral finite elements using harmonic basis functions, in: Computer Graphics Forum, Vol. 27, Wiley Online Library, 2008, pp. 1521–1529.

[21] N. Sukumar, Quadratic maximum-entropy serendipity shape functions for arbitrary planar polygons, Computer Methods in Applied Mechanics and Engineering 263 (2013) 27–41.

[22] L. Beirão da Veiga, F. Brezzi, A. Cangiani, G. Manzini, L. D. Marini, A. Russo, Basic principles of virtual element methods, Mathematical Models and Methods in Applied Sciences 23 (01) (2013) 199–214.

[23] K.-Y. Dai, G.-R. Liu, T.-T. Nguyen, An n-sided polygonal smoothed finite element method (nsfem) for solid mechanics, Finite elements in analysis and design 43 (11-12) (2007) 847–860.

[24] L. A. Freitag, P. E. Plassmann, Local optimization-based untangling algorithms for quadrilateral meshes., in: IMR, Citeseer, 2001.

[25] E. B. Chin, J. B. Lasserre, N. Sukumar, Numerical integration of homogeneous functions on convex and nonconvex polygons and polyhedra, Computational Mechanics 56 (6) (2015) 967–981.

[26] E. B. Chin, N. Sukumar, An efficient method to integrate polynomials over polytopes and curved solids, Computer Aided Geometric Design 82 (2020) 101914.

[27] S. Natarajan, S. Bordas, D. Roy Mahapatra, Numerical integration over arbitrary polygonal domains based on schwarz–christoffel conformal mapping, International Journal for Numerical Methods in Engineering 80 (1) (2009) 103–134.

[28] M. T. Heath, Scientific Computing: An Introductory Survey, Revised Second Edition, SIAM, 2018.

[29] H. Childs, Visit: an end-user tool for visualizing and analyzing very large data (2012).

[30] N. Sukumar, A. Tabarraei, Conforming polygonal finite elements, International Journal for Numerical Methods in Engineering 61 (12) (2004) 2045–2066.

[31] J. Sarrate, A. Huerta, Efficient unstructured quadrilateral mesh generation, International journal for numerical methods in engineering 49 (10) (2000) 1327–1350.

[32] N. A. Calvo, S. R. Idelsohn, All-hexahedral element meshing: Generation of the dual mesh by recurrent subdivision, Computer Methods in Applied Mechanics and Engineering 182 (3-4) (2000) 371–378.

[33] K. Verhetsel, J. Pellerin, J.-F. Remacle, A 44-element mesh of schneiders' pyramid: Bounding the difficulty of hex-meshing problems, Computer-Aided Design 116 (2019) 102735.

[34] M. Stees, S. M. Shontz, An angular approach to untangling high-order curvilinear triangular meshes, in: International Meshing Roundtable, Springer, 2018, pp. 327–342.

[35] J.-F. Remacle, T. Toulorge, J. Lambrechts, Robust untangling of curvilinear meshes, in: Proceedings of the 21st International meshing roundtable, Springer, 2013, pp. 71–83.

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

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

[38] T. J. Hughes, J. A. Cottrell, Y. Bazilevs, Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement, Computer methods in applied mechanics and engineering 194 (39-41) (2005) 4135–4195.

[39] Y. Bazilevs, V. M. Calo, J. A. Cottrell, J. A. Evans, T. J. R. Hughes, S. Lipton, M. A. Scott, T. W. Sederberg, Isogeometric analysis using t-splines, Computer Methods in Applied Mechanics and Engineering 199 (5-8) (2010) 229–263.

[40] S. Xia, X. Qian, Generating high-quality high-order parameterization for isogeometric analysis on triangulations, Computer Methods in Applied Mechanics and Engineering 338 (2018) 1–26.

[41] E. Cohen, T. Martin, R. Kirby, T. Lyche, R. Riesenfeld, Analysis-aware modeling: Understanding quality considerations in modeling for isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 199 (5-8) (2010) 334–356.

[42] G. Xu, B. Mourrain, R. Duvigneau, A. Galligo, Parameterization of computational domain in isogeometric analysis: methods and comparison, Computer Methods in Applied Mechanics and Engineering 200 (23-24) (2011) 2021–2031.

[43] D. Fußeder, B. Simeon, A.-V. Vuong, Fundamental aspects of shape optimization in the context of isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 286 (2015) 313–331.

[44] G. von Winckel, Legendre-gauss quadrature weights and nodes (lgwt.m), at: Matlab file exchange, `https://www.mathworks.com/matlabcentral/fileexchange/4540-legendre-gauss-quadrature-weights-and-nodes?s_tid=srchtitle` (2004).

[45] C. Ott, Non-linearly spaced vector generator (nonlinspace.m), at: Matlab file exchange, `https://www.mathworks.com/matlabcentral/fileexchange/64831-non-linearly-spaced-vector-generator?s_tid=srchtitle` (2017).