Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 342 (2018) 32–45
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Local refinement for analysis-suitable++ T-splines

Jingjing Zhang[a,b], Xin Li[a,*]

[a] University of Science and Technology of China, Hefei, Anhui, PR China
[b] Hefei Technological University, Hefei, Anhui, PR China

Received 24 January 2018; received in revised form 15 July 2018; accepted 18 July 2018
Available online 30 July 2018

Abstract

We develop an optimized local refinement algorithm for analysis-suitable++ T-splines (AS++ T-splines) which produces less propagation of control points. We then demonstrate its use as an adaptive framework for isogeometric analysis. AS++ T-splines are a class of T-splines which include analysis-suitable T-spline as a special case, are linearly independent, form a weighted partition of unity and are optimal convergent. These properties, coupled with the local refinement algorithm, make the AS++ T-spline appealing as a better basis both for design and analysis.

Keywords: T-splines; Linear independence; Isogeometric analysis; Analysis-suitable T-splines; Analysis-suitable++ T-splines

1. Introduction

T-splines [1,2] have proved to be an important technology for geometric modeling because of their supporting several very significant operations, such as local refinement [2,3], watertightness and trimmed NURBS conversion [1,4,5]. T-splines are attractive not only in geometric modeling but also in iso-geometric analysis (IGA), which uses the smooth spline basis that defines the geometry as a basis for analysis [6,7]. Although the whole class of T-splines are not suitable as a basis for IGA because of possible linear dependence [8], a mildly topological restricted subset of T-splines, analysis-suitable T-splines (AS T-splines), are optimized to meet the needs both for design and analysis [9,3,10–13]. Thus, the use of T-splines as the basis for IGA has gained widespread attention [14–24].

Analysis-suitable++ T-splines (for short, AS++ T-splines) are T-splines which define on a class of T-meshes with less restriction than AS T-splines, i.e., AS++ T-splines include AS T-splines as a special case. And meanwhile, AS++ T-splines maintain all the good properties as AS T-splines: they are linear independent [25], are NURBS compatible, obey the convex hull property, provide watertight models, and are weighted partition of unity (the constant function belongs to the AS++ T-spline space)—all the important properties for isogeometric analysis.

This paper contains two main contributions. First, we prove that the AS++ T-spline space is closed under all the existing local refinement algorithms, i.e., applying any existing local refinement algorithms in [2,3] on an AS++

* Corresponding author.
E-mail address: lixustc@ustc.edu.cn (X. Li).

https://doi.org/10.1016/j.cma.2018.07.024
0045-7825/

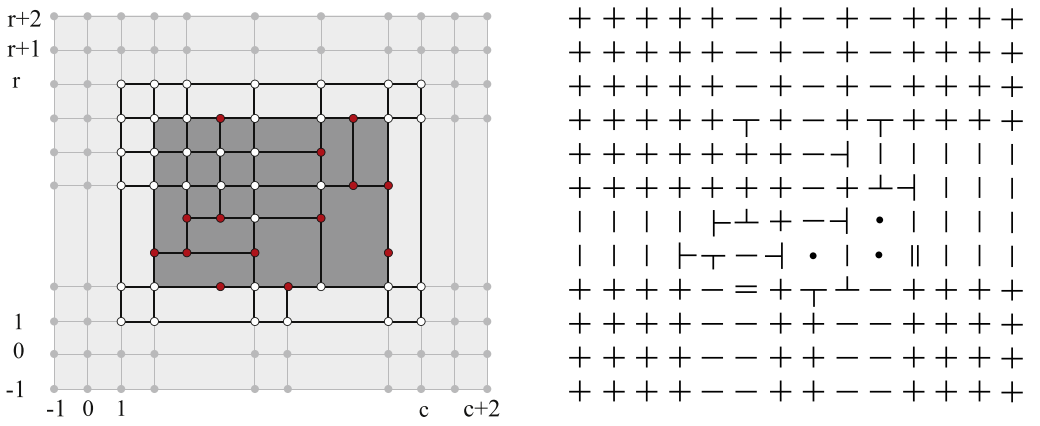


Fig. 1. An example T-mesh and the associated symbolic T-mesh.

T-spline will produce another AS++ T-spline. And second, we develop a highly localized refinement algorithm for AS++ T-splines which tries to minimize the number of additional control points. As the algorithm in [2] is a recursive procedure to refine all the possible influence blending functions without any optimization and the AS++ T-spline spaces are closed under the algorithm in [2], so our new algorithm is an improvement of the algorithm in [2] because our algorithm is an optimized algorithm for all the possible local refinement results in the AS++ T-spline space. The numerical examples and comparisons also indicate that our new algorithm has less propagation than that in [2]. On the other hand, AS++ T-splines have less restrictions than AS T-splines. So the new algorithm is also an improvement of the algorithm in [3]. And the numerical examples show that the new algorithm produces less propagation in most cases.

The rest of the paper is structured as follows. In Section 2, we recall some basic notations for bi-cubic T-splines. In Section 3, we provide the definition of the AS++ T-spline space and prove that the space is closed under the local refinement algorithm in [2]. In Section 4, we give the details of the new optimized local refinement algorithm. The comparisons and the application in the IGA will be discussed in Section 5. Section 6 is the conclusion and future work.

2. T-meshes and T-splines

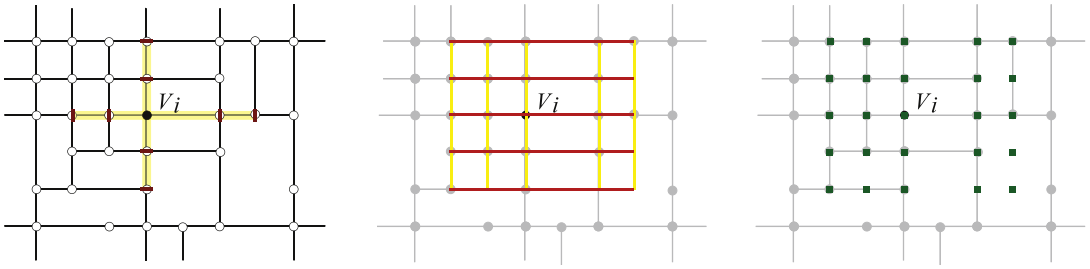
This section reviews the basic concept for bi-cubic T-splines.

2.1. Index T-mesh

An index T-mesh [14] T for a bi-cubic T-spline is a collection of all the elements of a rectangular partition of the index domain $[-1, c+2] \times [-1, r+2]$, where all rectangle corners (or vertices) have integer coordinates. In the following, (σ_i, τ_i) or $\{\sigma_i\} \times \{\tau_i\}$ denotes a vertex in T, and $[\sigma_j, \sigma_k] \times \{\tau_i\}$ ($\{\sigma_i\} \times [\tau_j, \tau_k]$) denotes a horizontal (vertical) edge or a set of connected horizontal (vertical) edges. Denote $[\sigma_i, \sigma_j] \times [\tau_k, \tau_l]$ ($(\sigma_i, \sigma_j) \times (\tau_k, \tau_l)$) to be a closed (open) face.

A symbolic T-mesh [9] is created by assigning a symbol to each element in a 2-D array formed from the index T-mesh T, see Fig. 1 as an example. The symbol is chosen to match the mesh topology of T. We adopt the symbol '+' to indicate valence four vertex, corner vertex, or valence three boundary vertex, symbols '⊢', '⊣', '⊥' and '⊤' to indicate the four possible orientations for the T-junctions, symbols '‖' and '=' to indicate two possible orientations of the I-junctions, symbols '|' and '−' to indicate a vertical and a horizontal edge, and symbol '.' to indicate no vertices and edges. In the following, the interior vertices can only be I-junctions, T-junctions and valence four vertices.

For the ith vertex $\mathbf{V}_i = (\sigma_i, \tau_i)$ in the rectangle $[1, c] \times [1, r]$, we define a local index vector $\vec{\sigma}_i \times \vec{\tau}_i$, where $\vec{\sigma}_i = [\sigma_i^0, \ldots, \sigma_i^4]$ and $\vec{\tau}_i = [\tau_i^0, \ldots, \tau_i^4]$. From the vertex, we shoot a ray in both directions traversing the T-mesh and collect a set of knot indices $\{\sigma_i^j\}$ and $\{\tau_i^k\}$ in both directions such that $\sigma_i^2 = \sigma_i$ and $\tau_i^2 = \tau_i$, as shown in Fig. 2.



a. Local knot vector $\vec{\sigma_i}$, $\vec{\tau_i}$ b. $SK(\mathbf{V}_i)$ c. $VK(\mathbf{V}_i)$

Fig. 2. Define the local index vector $\vec{\sigma_i}$, $\vec{\tau_i}$, the skeleton $SK(\mathbf{V}_i)$ and the vertex of the skeleton $VK(\mathbf{V}_i)$ for a vertex $\mathbf{V}_i$ in a T-mesh.

For the vertex $\mathbf{V}_i$, let $hSK(\mathbf{V}_i)$ be the union of all the edge segments $[\sigma_i^0, \sigma_i^4] \times \{\tau_i^j\}$, $j = 0, 1, \ldots, 4$ and $vSK(\mathbf{V}_i)$ is the union of all the edge segments $\{\sigma_i^j\} \times [\tau_i^0, \tau_i^4]$, $j = 0, 1, \ldots, 4$. Denote $SK(\mathbf{V}_i) = hSK(\mathbf{V}_i) \cup vSK(\mathbf{V}_i)$ and $VK(\mathbf{V}_i) = \{(\sigma_i^j, \tau_i^k), j, k = 0, 1, \ldots, 4\}$.

2.2. T-splines

Let T be an index T-mesh, $\vec{\mathbf{s}} = [s_{-1}, s_0, \ldots, s_{c+2}]$ and $\vec{\mathbf{t}} = [t_{-1}, t_0, \ldots, t_{r+2}]$ be two non-decreasing global knot vectors in $[0, 1]$, where $s_{-1} = \cdots = s_2 = t_{-1} = \cdots = t_2 = 0$, $s_{c-1} = \cdots = s_{c+2} = t_{r-1} = \cdots = t_{r+2} = 1$ and $0 < s_i, t_j < 1$ for $2 < i < c-1$, $2 < j < r-1$. Then for the ith vertex in the rectangle $[1, c] \times [1, r]$, we associated it with a blending function $T_i(s, t) = B[s_{\vec{\sigma_i}}](s)B[t_{\vec{\tau_i}}](t)$, where $B[s_{\vec{\sigma_i}}](s)$, $B[t_{\vec{\tau_i}}](t)$ are cubic B-spline basis functions defined in terms of knot vector $s_{\vec{\sigma_i}} = [s_{\sigma_i^0}, s_{\sigma_i^1}, \ldots, s_{\sigma_i^4}]$ and $t_{\vec{\tau_i}} = [t_{\tau_i^0}, t_{\tau_i^1}, \ldots, t_{\tau_i^4}]$.

A T-spline space $\mathbf{S}(\mathsf{T}, \vec{\mathbf{s}}, \vec{\mathbf{t}})$ defined on a T-mesh T with the knot vectors $\vec{\mathbf{s}}$ and $\vec{\mathbf{t}}$ is finally given as the linear space spanned by all these blending functions and a T-spline surface is defined as

$$\mathbf{T}(s, t) = \sum_{i=1}^{n_A} \mathbf{C}_i T_i(s, t) \tag{1}$$

where $\mathbf{C}_i = (\omega_i x_i, \omega_i y_i, \omega_i z_i, \omega_i) \in \mathbb{P}^3$ are homogeneous control points, $\omega_i \in \mathbb{R}$ are weights, $T_i(s, t)$ are blending functions, and n_A is the number of the T-mesh vertices in the rectangle $[1, c] \times [1, r]$.

2.3. Extensions and analysis-suitable T-splines

Analysis-suitable T-splines are defined in terms of T-junction extension, which is a line segment associated with each T-junction. For example, for a T-junction $\mathbf{T}_i$ of type $\vdash$, the edge extension $ext^e(\mathbf{T}_i)$ is a line segment $[\sigma_i^2, \sigma_i^3] \times \{\tau_i^2\}$ and the face extension $ext_a^f(\mathbf{T}_i)$ with an integer a is a line segment $[\sigma_i^{2-a}, \sigma_i^2] \times \{\tau_i^2\}$. If $\mathbf{T}_i$ is type of $\dashv$, then $ext^e(\mathbf{T}_i) = [\sigma_i^1, \sigma_i^2] \times \{\tau_i^2\}$ and $ext_a^f(\mathbf{T}_i) = [\sigma_i^2, \sigma_i^{2+a}] \times \{\tau_i^2\}$. And if $\mathbf{T}_i$ is type of $\|$, then $ext_a^f(\mathbf{T}_i) = [\sigma_i^{2-a}, \sigma_i^{2+a}] \times \{\tau_i^2\}$. Similarly, we can define the face extension for T-junctions of type $\perp$ or $\top$ and $=$. The extension of a T-junction $\mathbf{T}_i$ is $ext(\mathbf{T}_i) = ext^e(\mathbf{T}_i) \bigcup ext_2^f(\mathbf{T}_i)$.

Definition 2.1. A T-mesh is called an analysis-suitable T-mesh (for short, AS T-mesh) if and only if for any two T-junctions whose extensions are not parallel, the extensions of the two T-junctions do not intersect. An AS T-spline is a T-spline defined on an AS T-mesh.

2.4. Local refinement algorithm for T-splines

As we know, the local refinement algorithm is one of the most important algorithms for the geometric modeling and IGA with T-splines. There are two main local refinement algorithms for T-splines, the local refinement algorithm

Algorithm 1: T-spline local refinement algorithm

Require: A T-spline space defined on T-mesh T^1 and the insertion control points $\{N_j\}$;
Ensure: A T-spline defined on T^2 such that $\mathsf{T}^1 \bigcup \{N_j\} \subseteq \mathsf{T}^2$ and $\mathbf{S}(\mathsf{T}^1) \subseteq \mathbf{S}(\mathsf{T}^2)$;

```
Insert all the insertion control points into T^1
Put all the blending functions into a queue L
while L is not empty do
    Pop a blending function from L
    if The blending function has missing knots then
        Insert the missing knots into the blending function and put all the refined B-splines into L
    end if
    if The blending function has a knot that is not dictated then
        Add an appropriate control point associated with the knot
    end if
end while
```

in [2] and the AS T-spline local refinement algorithm in [3]. The algorithm in [2] is a recursive procedure to refine all the possible influence blending functions. Particularly, the algorithm in [2] is summarized in Algorithm 1.

[3] develops an optimized local refinement algorithm to enforce the T-spline after knot insertion to be an analysis suitable T-spline. The basic steps of the algorithm are described as follows Algorithm 2, for more details, please refer to [3].

Algorithm 2: AS T-spline local refinement algorithm

Require: A T-spline space defined on an AS T-mesh T^1 and the insertion control points $\{N_j\}$;
Ensure: A T-spline defined on an **AS T-mesh** T^2 such that $\mathsf{T}^1 \bigcup \{N_j\} \subseteq \mathsf{T}^2$ and $\mathbf{S}(\mathsf{T}^1) \subseteq \mathbf{S}(\mathsf{T}^2)$;

```
Insert all the control points {N_j} into T^1
Convert the T-mesh after insertion into an AS T-mesh by minimizing the number of
    new insertion control points using a greedy strategy;
Compute the positions for the control points.
```

3. AS++ T-splines

Before giving the definition of AS++ T-splines, we first introduce some notations. An extended T-mesh for a T-mesh T is a new T-mesh from the *extended T-mesh set* $ext(\mathsf{T})$, where

$$ext(\mathsf{T}) = \{\mathsf{T}_1 \parallel \mathsf{T}_1 = \bigcup_{\mathbf{T}_i \in \mathsf{T}} ext_{a_i}^f(\mathbf{T}_i) \bigcup \mathsf{T}, a_i \geq 0 \text{ is an integer associated with the T-junction } \mathbf{T}_i\}.$$

The edges in the extended T-mesh but not in the original T-mesh are called the *extended edges*. Two T-junction extensions maybe overlapping. Thus, the *multiplicity* of an extended edge is the number of T-junction face extensions that contain the extended edge, which is two if the edge belongs to two overlapping extensions and is one for the other cases. For two extended T-meshes $\mathsf{T}_1, \mathsf{T}_2 \in ext(\mathsf{T})$, we said $\mathsf{T}_1 = \mathsf{T}_2$ if and only if all the extended edges are same and the multiplicities for the corresponding extended edges are also the same. Denote

$$\mathsf{T}_{ext} = \bigcup_{\mathbf{T}_i \in \mathsf{T}} ext_2^f(\mathbf{T}_i) \bigcup \mathsf{T}, \mathsf{T}_{elem} = \bigcup_{\mathbf{V}_i \in \mathsf{T}} SK(\mathbf{V}_i) \bigcup \mathsf{T}.$$

The T-mesh T_{ext} is a T-mesh that contains the origin T-mesh and all the face extensions, while the T-mesh T_{elem} is a new T-mesh formed by all the local Bézier meshes for each blending function. Figs. 3 and 4 show two example T-meshes and the associated T_{ext} and T_{elem}. In the first example, the T-meshes T_{ext} and T_{elem} are shown in Fig. 3b and c. We can see that the two T-meshes are different because there is an edge that belongs to T_{elem} but does not belong to T_{ext}. In the second example, the T-mesh T_{ext} (Fig. 4b) is also different from the T-mesh T_{elem} (Fig. 4c) because

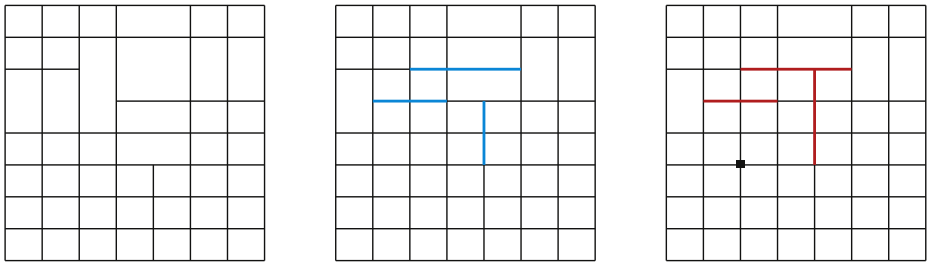


Fig. 3. The extended T-mesh T_{ext} (b) is different from the element T-mesh T_{elem} (c).

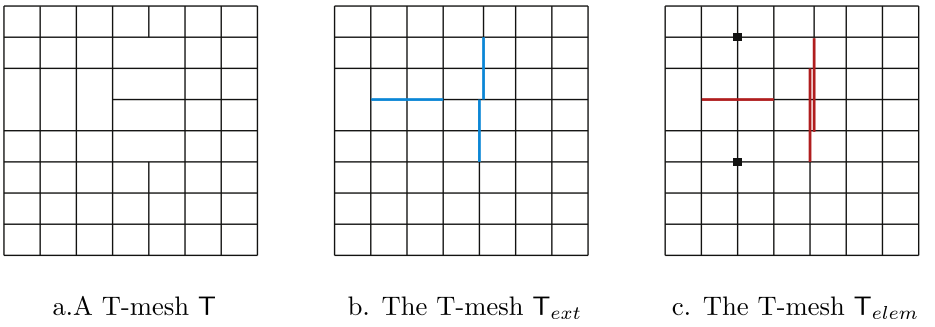


Fig. 4. The extended T-mesh T_{ext} (b) is different from the element T-mesh T_{elem} (c) because the multiplicities for some extended edges in T_{ext} and T_{elem} are different. The multiplicity for all the extended edges in T_{ext} is one but the multiplicity for two extended edges in T_{elem} is two.

the multiplicities for some extended edges in T_{ext} and T_{elem} are different. In order to make the figure to be easily understood, we move one of the T-junctions a little bit such that we can count the multiplicities easier.

Definition 3.1. A T-mesh is called an analysis-suitable++ T-mesh (for short, AS++ T-mesh) if and only if:

1. For any two T-junctions $\mathbf{T}_i, \mathbf{T}_j$ whose extensions are not parallel, denote $V = ext_2^f(\mathbf{T}_i) \cap ext_2^f(\mathbf{T}_j)$, then either $ext_2^f(\mathbf{T}_i) \cap ext_2^f(\mathbf{T}_j) = \varnothing$ (no V exists) or for any $\mathbf{V}_k$, $V \notin VK(\mathbf{V}_k)$;
2. $\mathsf{T}_{ext} = \mathsf{T}_{elem}$.

An AS++ T-spline is a T-spline defined on an AS++ T-mesh.

Lemma 3.2 (a) and (b) in [12] state that AS T-meshes satisfy the requirement of AS++ T-meshes, i.e., AS T-meshes are always AS++ T-meshes. On the contrary, AS++ T-meshes are not always AS T-meshes. For example, the T-mesh in Fig. 5a. is an AS++ T-mesh but is not an AS T-mesh because the extensions of two red T-junctions intersect. We can check that any two non-parallel face extensions do not intersect and $\mathsf{T}_{ext} = \mathsf{T}_{elem}$. The T-mesh in Fig. 5b. is also an AS++ T-mesh but is not an AS T-mesh. In this example, there are many T-junction intersections including two face extensions intersection at the vertex V. But we can check that for any vertices $\mathbf{V}_i$, $V \notin VK(\mathbf{V}_i)$, which states that the T-mesh is an AS++ T-mesh. The cases of the red T-junctions in Fig. 5c are very popular in the real applications.

3.1. Properties for AS++ T-splines

The following properties have been discovered for AS++ T-splines according to the requirement from IGA.

- The blending functions for the AS++ T-splines are linearly independent for all the knots [25] while the blending functions for AS T-splines are *locally* linear independent.

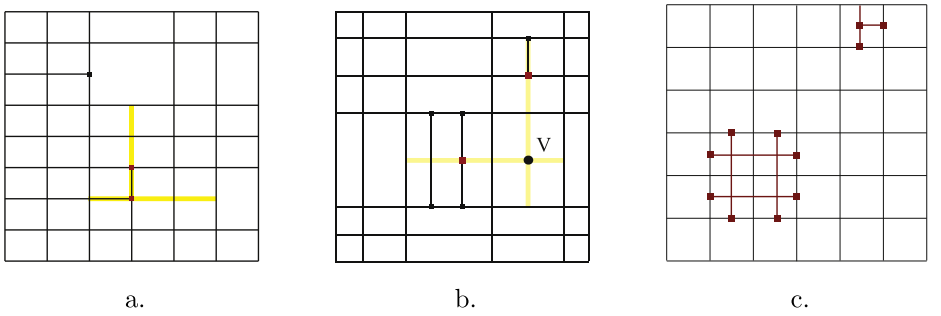


Fig. 5. Three example AS++ T-meshes which are all not AS T-meshes.

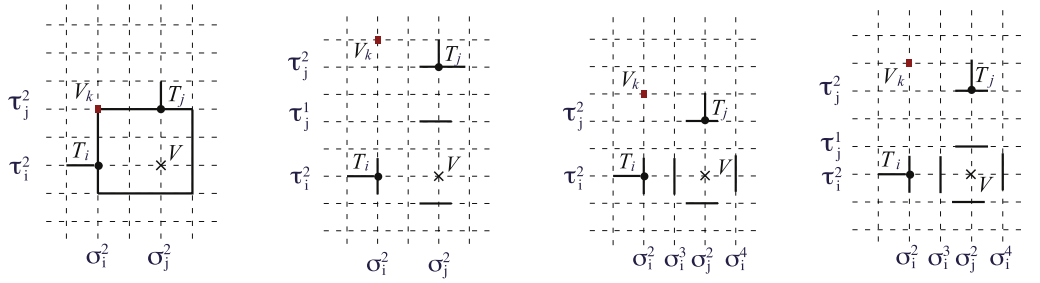


Fig. 6. The four possible cases for the face extension intersection.

- The basis functions are weighted partition of unity [25], i.e., there exist some weights ω_i which are function of knot intervals and the T-mesh, such that

$$\sum_{i=1}^{n_A} \omega_i T_i(s,t) = 1.$$

  And the weights can be very easily computed through dual basis.
- AS++ T-splines obey the convex hull property.
- A dual basis can be constructed [25].
- The AS++ T-Spline space can be characterized in terms of piecewise polynomials [26].

3.2. AS++ T-spline space under local refinement algorithms

As AS T-splines are always AS++ T-splines, we can guarantee that the AS++ T-spline space is closed under the local refinement algorithm for AS T-splines in [3]. In the following, we prove that the AS++ T-spline space is also closed under the local refinement algorithm in [2].

Theorem 3.2. *Given a T-spline defined on an AS++ T-mesh* T^1*, we insert a set of control points into* T^1 *via local refinement Algorithm 1 and produce a T-spline defined on a T-mesh* T^2*, then* T^2 *is an AS++ T-mesh.*

Proof. Denote $N = \{N_i, i = 1, \ldots, n\}$ to be the set of vertices which are in T-mesh T^2 but not in the T-mesh T^1. In order to prove that the T-mesh T^2 is an AS++ T-mesh, we only need to prove that T^2 satisfies the two conditions for AS++ T-splines one by one.

First, we prove that the T-mesh T^2 satisfies the first condition. Otherwise, referring to Fig. 6 there at least exist two T-junctions $\mathbf{T}_i = (\sigma_i, \tau_i)$ and $\mathbf{T}_j = (\sigma_j, \tau_j)$ such that $V = ext_2^f(\mathbf{T}_i) \cap ext_2^f(\mathbf{T}_j)$ and $V \in VK(\mathbf{V}_k)$ for some vertex $\mathbf{V}_k$. Without loss of generalization, we assume that $\sigma_i = \sigma_i^2 < \sigma_j^2$ and $\tau_i = \tau_i^2 < \tau_j^2$, then we only need to prove that the following four cases are impossible.

- $\sigma_j^2 \in [\sigma_i^2, \sigma_i^3]$ and $\tau_i^2 \in [\tau_j^1, \tau_j^2]$: In this case, we consider the blending function for $\mathbf{V}_k$, since T^1 is an AS++ T-mesh, so there is at least one of $\mathbf{T}_i$ or $\mathbf{T}_j$ that belongs to the set N. Without loss of generalization, we assume that $\mathbf{T}_j \in N$. Then the blending function for $\mathbf{V}_k$ has a missing knot σ_j^2 during refinement, so we need to refine the blending function at the missing knots σ_j^2, which will cause that $V \in \mathsf{T}^2$. However, this contradicts the assumption.
- $\sigma_j^2 \in [\sigma_i^2, \sigma_i^3]$ and $\tau_i^2 \in [\tau_j^0, \tau_j^1)$: In this case, we can prove the theorem similarly as the first case. We still consider the blending function for $\mathbf{V}_k$ and we assume $\mathbf{T}_j \in N$. Then the blending function for $\mathbf{V}_k$ has a missing knot σ_j^2 during refinement, so we need to refine the blending function at the missing knot σ_j^2, which will cause that $V \in \mathsf{T}^2$. However, this contradicts the assumption.
- $\sigma_j^2 \in (\sigma_i^3, \sigma_i^4]$ and $\tau_i^2 \in [\tau_j^1, \tau_j^2]$: There are two possible cases to prove this one. If $\sigma_i^3 \in \{\sigma_k^p, p = 0, \ldots, 4\}$, then we can similarly prove that $(\sigma_i^3, \tau_i^2) \in \mathsf{T}^2$. And then we replace vertex $\mathbf{T}_i$ with vertex (σ_i^3, τ_i^2), which is converted into the first case. Otherwise, if $(\sigma_i^3, \tau_i^2) \notin \mathsf{T}^2$, then this is exactly same as the first case. So in both cases, we can conclude it is impossible.
- $\sigma_j \in (\sigma_i^3, \sigma_i^4]$ and $\tau_i \in [\tau_j^0, \tau_j^1)$: This is similar as third case which can be reduced to second case by considering the vertex (σ_i^3, τ_i^2).

Now, we prove that the T-mesh T^2 satisfies the second condition, i.e., $\mathsf{T}^2_{ext} = \mathsf{T}^2_{elem}$. Let $\mathbf{V}_i^k = (\sigma_i^2, \tau_i^2)$, $k = 1,2$ be the ith vertex in T-mesh T^k. Then for any blending function for T-spline defined on the T-mesh T^1, it can be written into the linear combinations of the blending functions for T-spline defined on the T-mesh T^2 through B-spline basis functions refinement. Suppose that a blending function associated with the vertex $\mathbf{V}_i^1$ is refined into two blending functions associated with vertices $\mathbf{V}_i^2$ and $\mathbf{V}_j^2$, then it is obvious that $SK(\mathbf{V}_i^2) \cup SK(\mathbf{V}_j^2) \subseteq SK(\mathbf{V}_i^1) \bigcup_{\mathbf{T}_k \in \mathsf{T}^2} ext^f(\mathbf{T}_k)$. Thus we can conclude that $\bigcup_{\mathbf{V}_i^2 \in \mathsf{T}^2} SK(\mathbf{V}_i^2) = \mathsf{T}^1_{elem} \bigcup_{\mathbf{T}_k \in \mathsf{T}^2} ext^f(\mathbf{T}_k)$. According to the assumption, $\mathsf{T}^1_{elem} = \mathsf{T}^1_{ext}$. So $\mathsf{T}^2_{elem} = \bigcup_{\mathbf{V}_i^2 \in \mathsf{T}^2} SK(\mathbf{V}_i^2) = \mathsf{T}^1_{ext} \bigcup_{\mathbf{T}_k \in \mathsf{T}^2} ext^f(\mathbf{T}_k) = \mathsf{T}^2_{ext}$. In the summary, we can conclude that T^2 is an AS++ T-mesh. □

4. Local refinement for AS++ T-splines

In this section, we first provide a concise numerical framework for refinement and then develop a specific local refinement algorithm. By restricting the T-spline being an AS++ T-spline, we can develop a simple local refinement algorithm which introduces a minimal number of superfluous control points and preserves the mathematical properties of the AS++ T-spline space.

Similar as AS T-spline space, the nesting behavior between a T-spline space and an AS++ T-spline space can be characterized in the following lemma whose proof is given in the forthcoming paper [26].

Lemma 4.1. *For a T-spline space* $\mathbf{S}(\mathsf{T}^1)$ *and an AS++ T-spline space* $\mathbf{S}(\mathsf{T}^2)$*, if* $\mathsf{T}^1_{elem} \subseteq \mathsf{T}^2_{elem}$ *and the multiplicity of any extended edges in* T^1_{elem} *is less than or equal to that of the corresponding extended edges in* T^2_{elem}*, then* $\mathbf{S}(\mathsf{T}^1) \subseteq \mathbf{S}(\mathsf{T}^2)$.

The main objective is to devise an algorithm for finding the fewest additional control points automatically. However, finding the minimal number of additional control points is an NP-hard problem. So for efficiency, our algorithm uses the following greedy strategy that only provides an approximate minimum.

The first step in generating an algorithm to approximate the minimal T-mesh is to quantify all violations of the two AS++ T-spline conditions. In the following, we call it a violation of intersection if two T-junctions face extensions intersect and the intersection belongs to some $VK(\mathbf{V}_k)$. And we call it a violation of equivalence if $\mathsf{T}_{ext} \neq \mathsf{T}_{elem}$. The main steps for AS++ local refinement are Algorithm 3.

4.1. Remove the intersection violations

Similar as the approach in [3], we use the extension graph to guide the removal of the intersection violations. The difference between our algorithm and [3] is that the extension graph we are using only contains the face extensions while the AS T-spline local refinement algorithm requires all the T-junctions extensions. The extension graph for a T-mesh T is denoted by $E(\mathsf{T})$. Each node in $E(\mathsf{T})$ corresponds to a single T-junction in T. If two face extensions in

Algorithm 3: AS++ T-spline local refinement algorithm

Require: A T-spline space defined on T-mesh T^1 and a set of control points $\{N_i\}$;
Ensure: A T-spline defined on an AS++ T-mesh T such that $\mathsf{T}^1 \bigcup \{N_i\} \subseteq \mathsf{T}$ and $\mathbf{S}(\mathsf{T}^1) \subseteq \mathbf{S}(\mathsf{T})$;
1: Insert all the $\{N_i\}$ into T^1 to get T^2
2: **while** T^k is not AS++ T-spline **do**
3: Remove the intersection violations for T^k;
4: Remove the equivalence violations for the T-mesh created from the last step;
5: Set T^{k+1} to be the T-mesh in the last step;
6: **end while**
7: **return** T^{k+1}

T intersect and the intersection belongs to some $VK(\mathbf{V}_i)$, then an edge is drawn between the corresponding nodes in $E(\mathsf{T})$. For example, for the T-mesh in Fig. 7a, the corresponding extension graph is drawn in Fig. 7b. In the T-mesh, there are seven T-junctions, which means the number of nodes in the extension graph is seven. Each edge in the extension graph corresponds to one pair of violation of intersections. We can see that there exists one T-junction whose face extension has no intersection with the other face extensions. So there are no edges connecting the node in $E(\mathsf{T})$.

Our basic idea to remove the intersection violations is using a greedy strategy. Our basic objective is to extend as less as possible T-junctions such that the new T-mesh has no intersection violations. Suppose the current T-mesh is T and let k be the number of nodes in the extension graph which has at least one edge connecting the node. Denote the T-mesh T_i to be the T-mesh by extension the T-junction corresponding to the ith node one bay in the original T-mesh. We check the number of edges in the corresponding extension graph of all the possible T-meshes T_i and pick the one that has smallest number of edges. If there are more than one such T-mesh, then we pick any of them. And then we perform the same process for the picking T-mesh until there are no intersection violations in the current mesh. It should be noted that the algorithm will always be terminated, because in the worst case, if all T-junctions are extended all the way to a boundary edge, a NURBS is created.

Now we demonstrate the algorithm on the following simple example in Fig. 8. Suppose we insert the red edges in the T-mesh in Fig. 8a, which cause two face extension intersections. In the T-mesh, there are three T-junctions. If we extend the first T-junction and get the new T-mesh as shown in Fig. 8b, there are still two edges in the extension graph. If we extend the second T-junction, then there is only one edge in the corresponding extension graph as illustrated in Fig. 8c. And if we extend the third T-junction, then there are no edges in the extension graph. So we will pick the last T-mesh which removes the intersection violations.

4.2. Remove the equivalence violations

The next step is to remove the equivalence violations. Because $\mathsf{T}_{ext} \subseteq \mathsf{T}_{elem}$ and according to the assumption, $\mathsf{T}_{ext} \neq \mathsf{T}_{elem}$, so there are existing some vertices $\mathbf{V}_{i_j}$, $j = 1, 2, \ldots, k$ such that $SK(\mathbf{V}_{i_j}) \nsubseteq \mathsf{T}_{ext}$. Now we will remove all the equivalence violations by considering all the T-junctions on the $SK(\mathbf{V}_{i_j})$, $j = 1, 2, \ldots, k$. For each equivalence violation, we check all the possible ways to extend the T-junctions on the $SK(\mathbf{V}_{i_j})$ such that it removes the equivalence violations. For the ith possible way, denote the number of the new control points to be n_i^n and the number of edges in the extension graph after extension to be n_i^e, we will pick the one in which the number of $n_i^n + n_i^e$ is to be minimal. Same as the last step, the algorithm will also be terminated.

For example, for the T-mesh created by Fig. 8, the skeleton of the star vertex violates the equivalence violations. And there are two T-junctions (the two blue rectangle control points) on the skeleton of the star vertex. For the first T-junction, if we extend the T-junction as shown in Fig. 9b and it will remove the equivalence violations but leads one edge in the extension graph. And if we do not extend the first T-junction, and in order to remove the equivalence violations, we need to extend the second T-junction one way as shown in Fig. 9c. And in this case, we need to add one additional control point and it has no edges in the extension graph. So we will the pick the T-mesh in Fig. 9c. as the final T-mesh.

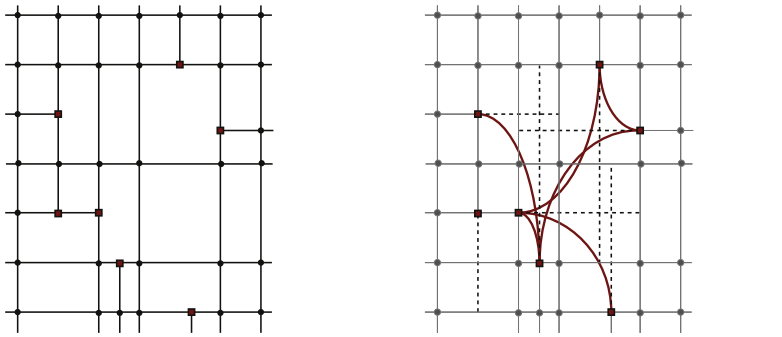


Fig. 7. A T-mesh and the extension graph.

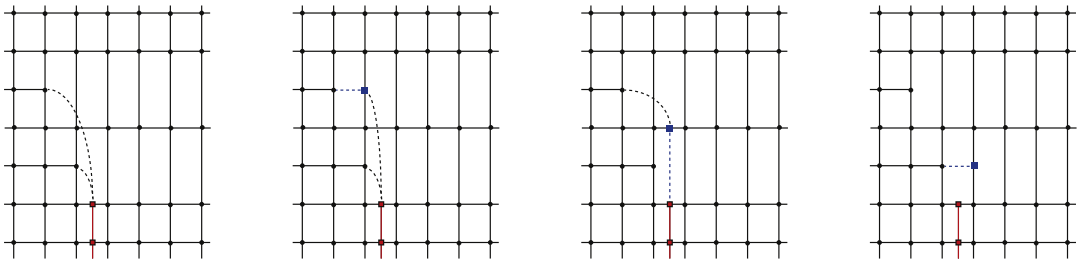


Fig. 8. Remove the intersection violations. a is the T-mesh after insertion of the red edges whose extension graph has two edges. b, c, and d are the T-meshes which extended three possible T-junctions whose extension graph has 2, 1, and 0 edges.

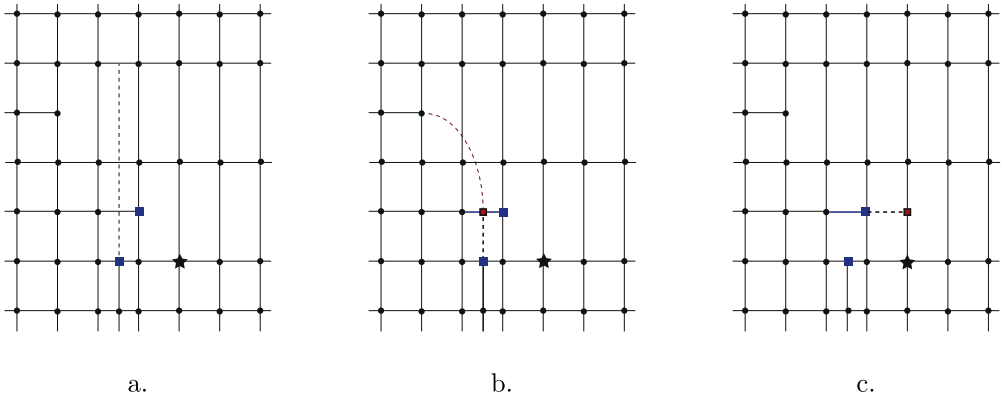


Fig. 9. Remove the equivalence violations. Figure b and c are two possible ways to remove the equivalence violation for the pentagram control point. And Figure b will lead one intersection violation and Figure c will not lead any intersection violation. So the algorithm will choose the way to extend as Figure c.

Remark 4.2. Although the local refinement algorithm for AS T-splines produces less additional control points than the algorithm in [2], but we cannot regard the algorithm in [3] as an improvement of that in [2] because the T-splines created from the algorithm in [2] are generally not AS T-splines. There are many examples for which the algorithm in [2] is better than that in [3], for example the T-meshes in Fig. 5. Theorem 3.2 states that the new optimized local refinement algorithm can be regarded as an improvement of that in [2].

Table 1
Comparisons of AS++ refinement and AS refinement.

	$n_{as++} < n_{as}$	$n_{as++} = n_{as}$	$n_{as++} > n_{as}$
$\frac{N^*}{N}$	86.7%	4.2%	9.1%
$\frac{diff}{n_c}$	13.2%		1.4%

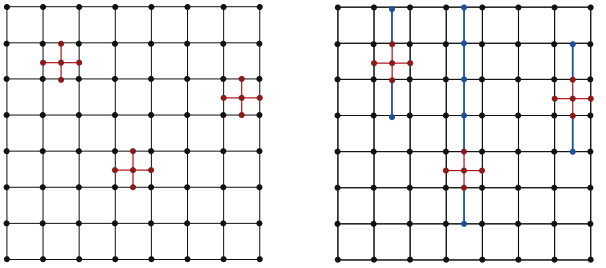


Fig. 10. Comparisons of AS++ refinement and AS refinement.

5. Applying AS++ T-spline local refinement algorithm

In this section, we demonstrate the superiority of the algorithm over the original T-spline local refinement algorithm in [2] and the AS T-spline local refinement algorithm in [3].

5.1. Comparisons of the propagation

We first compare our algorithm with the local refinement algorithm in [2] and the AS T-spline local refinement algorithm in [3] on the propagations of the control points. The basic framework for the comparisons is from a $m \times m$ (m = 10, 20, ..., 50) tensor-product mesh, randomly select n faces and split each face into four small faces by bisection the face. Suppose the number of control points in this process is n_c. And then, we run the three algorithms and count the number of control points after insertion respectively. Suppose the number of control points for the algorithm in [2], the AS refinement in [3] and the algorithm in the present paper are n_{old}, n_{as} and n_{as++} respectively. Let N be the number of tests which is set to be 10^4. Now, we first compute the number of tests, denoted as N^+ where $n_{as++} > n_{as}$. Let N^0 and N^- be the number of tests where $n_{as++} = n_{as}$ and $n_{as++} < n_{as}$ respectively. Then, we measure how many of the tests which the AS++ refinement algorithm is prior than the AS refinement algorithm. We also measure the difference of the number of control points compared with the number of control points we wish to insert. The statistics are connected in Table 1. For the comparisons with the original algorithm in [2], we find the new algorithm has less or equal number of control points in all the testing examples.

Fig. 10 shows a simple example of above tests. We try to insert some edges (red) into a tensor-product mesh. By applying the local refinement algorithm in the present paper, we do not need to insert any additional control points and if we apply the AS T-spline local refinement algorithm, the resulting T-mesh is shown on the right. All the blue control points are the propagation control points. Fig. 11 shows another example for the comparisons which inserts only one single control point into a given T-spline. As shown in Fig. 11, given a T-mesh (the black one), we insert one single control point (the red one) into the T-mesh. If we apply the AS T-spline refinement algorithm, we need to insert all the blue control points, which are shown on the right of Fig. 11. However, if we apply the AS++ T-spline refinement algorithm, then we only need to insert one more control point, as shown on the left of Fig. 11.

5.2. Advection dominated advection–diffusion problem

We now explore the application and behavior in IGA of the new AS++ T-spline space and the corresponding local refinement algorithm. We solve the advection–diffusion equation

$$\kappa \Sigma u + \mathbf{a} \nabla u = 0 \tag{2}$$

in the unit square with discontinuous Dirichlet boundary conditions as shown in Fig. 12.

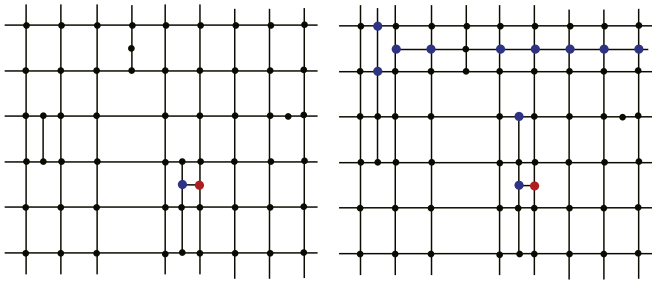


Fig. 11. Comparisons of AS++ refinement and AS refinement.

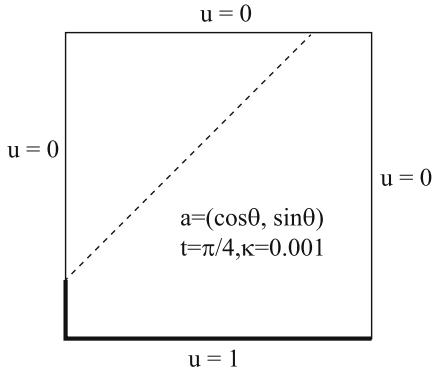


Fig. 12. Setup for the advection dominated advection–diffusion problem.

Table 2
Comparisons of the elements.

	Refinement 1	Refinement 2	Refinement 3
AS++ refinement	222	637	1744
Refinement in [2]	256	1017	3757

In this example, the Peclet number, which characterizes competition between advection and diffusion, is set to be 10^{-3}. This makes the problem strongly advection dominated and gives rise to a sharp interior layer and two sharp boundary layers at the outflow. This is an ideal benchmark problem to exercise the capabilities of our new refinement algorithm. This is due to the anisotropic orientation of the interior layer with respect to the linear parameterization of the geometry. It was noted in [27,14] that employing the original T-spline refinement scheme for this problem (when the degree is greater than 1) produced nearly global propagation of refinement.

We begin with a cubic uniform B-spline mesh with 8×8 elements (10×10 control grid). Note that in all cases the interior element edges are C^2 while boundary edges are C^0. We employ an automatic refinement scheme that makes use of a simple gradient-based error indicator. At each step, the Bézier elements are identified for refinement based on the magnitude of the indicator. Each of these Bézier elements is then bisected by inserting knots into the governing T-mesh at the appropriate locations. In all cases, the standard SUPG formulation [28] is used with stabilization parameter $\tau = \frac{h_a}{2a}$, where h_a is the Bézier element length in the direction of the flow velocity. For the problem considered, $a = |\mathbf{a}| = 1$ and $h_a = \sqrt{2}h$, where h is the element edge length. The number of elements for the algorithm in [2] and the algorithm in the present paper are listed in Table 2.

The Bézier meshes and corresponding solutions for each refinement step are shown in Fig. 13. The Bézier meshes and solutions generated using the new algorithm are on the right. The Bézier meshes generated using the new algorithm show a sharp change in mesh density in the areas corresponding to the interior and boundary layers. The meshes generated using the old algorithm suffer from nearly global refinement. We find that using either refinement scheme

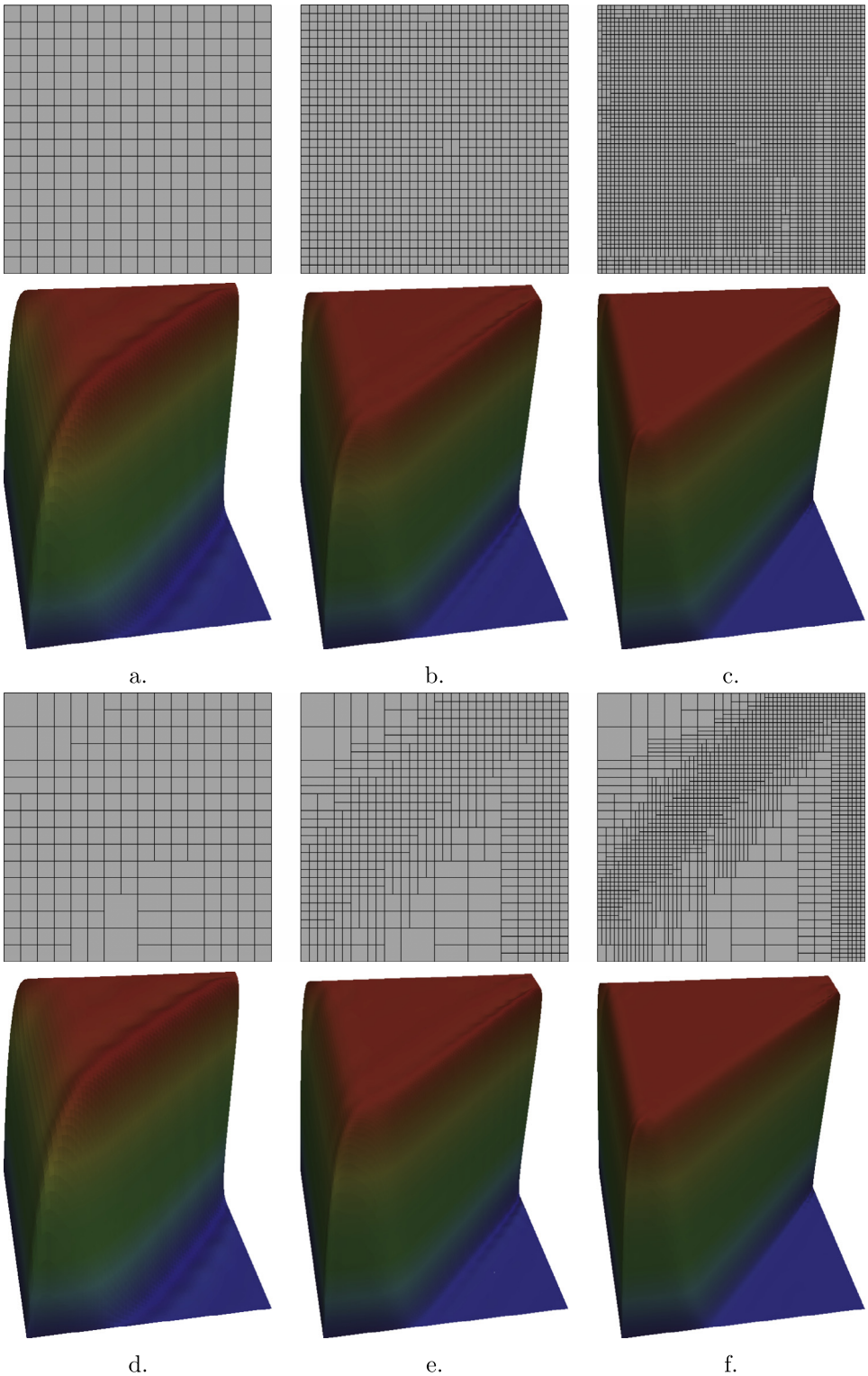


Fig. 13. The Bézier meshes and corresponding solutions for refinements one, two and three. The first row is the Bézier meshes generated from the old local refinement algorithm in [2], the second row is the corresponding solutions, the third row is generated from the new algorithm in the present paper and the fourth row is the corresponding solutions.

the layers become sharper, and the overshoots and undershoots about the layers are attenuated in a nearly identical manner. However, the solution in the third level is obtained with less than half the number of basis functions as that in Fig. 13. This discrepancy widens with more refinement.

We should mention that in IGA application example, our new algorithm produces exactly the same result as those for the AS T-spline local refinement algorithm. The reason for this is that all the extension intersections in this example are face extension intersections. If we consider the conditions for the AS T-splines and AS++ T-splines, we can find that although the AS++ T-splines allow face extension intersection (see the second T-mesh in Figure 5 in [25]), the main advantage for AS++ T-splines is that they allow edge extension intersections while the AS T-splines do not allow. Thus, in the random insertion tests, the AS++ T-splines have less propagations in most cases than AS T-splines.

6. Conclusion

We have developed an efficient adaptive framework to preform local refinement for AS++ T-splines. The sequence of refined spaces is nested and exactly preserves the initial geometry. The basis of the refined T-spline spaces maintains the smoothness of the initial basis. We have demonstrated its effectiveness on a randomly selected example and the benchmark problem in IGA. The procedures described provide a powerful methodology for instantiating the vision of isogeometric analysis. In future work, we plan to describe the approaches to handle arbitrary degrees and the treatment for extraordinary points.

Acknowledgments

The authors are supported by the NSF of China (Nos. 11031007, 60903148, 11371341), a NKBRPC (2011CB302400), the Fundamental Research Funds for the Central Universities, SRF for ROCS SE, and the Youth Innovation Promotion Association CAS.

References

[1] T.W. Sederberg, J. Zheng, A. Bakenov, A. Nasri, T-splines and T-NURCCSs, ACM Trans. Graph. 22 (3) (2003) 477–484.
[2] T.W. Sederberg, D.L. Cardon, G.T. Finnigan, N.S. North, J. Zheng, T. Lyche, T-spline simplification and local refinement, ACM Trans. Graph. 23 (3) (2004) 276–283.
[3] M.A. Scott, X. Li, T.W. Sederberg, T.J.R. Hughes, Local refinement of analysis-suitable T-splines, Comput. Methods Appl. Mech. Engrg. 213–216 (2012) 206–222.
[4] H. Ipson, T-spline Merging (Master's thesis), Brigham Young University, 2005.
[5] T.W. Sederberg, G.T. Finnigan, X. Li, H. Lin, H. Ipson, Watertight trimmed NURBS, ACM Trans. Graph. 27 (3) (2008) 79.
[6] T.J.R. Hughes, J.A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry, and mesh refinement, Comput. Methods Appl. Mech. Engrg. 194 (2005) 4135–4195.
[7] J.A. Cottrell, T.J.R. Hughes, Y. Bazilevs, Isogeometric Analysis: Toward Integration of CAD and FEA, Wiley, Chichester, 2009.
[8] A. Buffa, D. Cho, G. Sangalli, Linear independence of the T-spline blending functions associated with some particular T-meshes, Comput. Methods Appl. Mech. Engrg. 199 (23–24) (2010) 1437–1445.
[9] X. Li, J. Zheng, T.W. Sederberg, T.J.R. Hughes, M.A. Scott, On the linear independence of T-splines blending functions, Comput. Aided Geom. Design 29 (2012) 63–76.
[10] L.B. Veiga, A. Buffa, D.C.G. Sangalli, Analysis-suitable T-splines are dual-compatible, Comput. Methods Appl. Mech. Engrg. 249–252 (2012) 42–51.
[11] J. Zhang, X. Li, On the linear independence and partition of unity of arbitrary degree analysis-suitable T-splines, Commun. Math. Stat. 3 (3) (2015) 353–364.
[12] L.B. Veiga, A. Buffa, G. Sangalli, R. Vazquez, Analysis-suitable T-splines of arbitrary degree: definition and properties, Math. Models Methods Appl. Sci. 23 (2013) 1979–2003.
[13] X. Li, M.A. Scott, Analysis-suitable T-splines: characterization, refinablility and approximation, Math. Models Methods Appl. Sci. 24(06) (2014) 1141–1164.
[14] 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, Comput. Methods Appl. Mech. Engrg. 199 (5–8) (2010) 229–263.
[15] M.J. Borden, C.V. Verhoosel, M.A. Scott, T.J.R. Hughes, C.M. Landis, A phase-field description of dynamic brittle fracture, Comput. Methods Appl. Mech. Engrg. 217–220 (2012) 77–95.
[16] D.J. Benson, Y. Bazilevs, E. De Luycker, M.C. Hsu, M.A. Scott, T.J.R. Hughes, T. Belytschko, A generalized finite element formulation for arbitrary basis functions: from isogeometric analysis to XFEM, Internat. J. Numer. Methods Engrg. 83 (2010) 765–785.
[17] L.B. Veiga, A. Buffa, D.C.G. Sangalli, Isogeometric analysis using T-splines on two-patch geometries, Comput. Methods Appl. Mech. Engrg. 200 (2011) 1787–1803.
[18] A. Buffa, D. Cho, M. Kumar, Characterization of T-splines with reduced continuity order on T-meshes, Comput. Methods Appl. Mech. Engrg. 201–204 (2012) 112–126.

[19] M.A. Scott, R.N. Simpson, J.A. Evans, S. Lipton, S.P.A. Bordas, T.J.R. Hughes, T.W. Sederberg, Isogeometric boundary element analysis using unstructured T-splines, Comput. Methods Appl. Mech. Engrg. 254 (2013) 197–221.
[20] R. Dimitri, L.D. Lorenzis, M.A. Scott, P. Wriggers, R.L. Taylor, G. Zavarise, Isogeometric large deformation frictionless contact using T-splines, Comput. Methods Appl. Mech. Engrg. 269 (2014) 394–414.
[21] L. Liu, Y. Zhang, T.J.R. Hughes, M.A. Scott, T.W. Sederberg, Volumetric T-spline construction using boolean operations, Eng. Comput. 30 (2014) 425–439.
[22] D. Schillinger, L. Dede, M.A. Scott, J.A. Evans, M.J. Borden, E. Rank, T.J.R. Hughes, An isogeometric design-through-analysis methodology based on adaptive hierarchical refinement of NURBS, immersed boundary methods, and T-spline CAD surfaces, Comput. Methods Appl. Mech. Engrg. 249–252 (2014) 116–150.
[23] X. Wei, Y. Zhanga, L. Liu, T.J. Hughes, Truncated T-splines: Fundamentals and methods, Comput. Methods Appl. Mech. Engrg. 316 (2017) 349–372.
[24] L. Liu, Y.J. Zhang, X. Wei, Weighted T-splines with application in reparameterizing trimmed NURBS surfaces, Comput. Methods Appl. Mech. Engrg. 295 (2015) 108–126.
[25] X. Li, J. Zhang, Analysis-Suitable++ T-splines: linear independence and approximation, Comput. Methods Appl. Mech. Engrg. 333 (2018) 462–474.
[26] X. Li, Characterization and approximation for analysis-suitable++ T-splines, Math. Models Methods Appl. Sci. (2018) in preparation.
[27] M. Dörfel, B. Jüttler, B. Simeon, Adaptive isogeometric analysis by local *h*-refinement with T-splines, Comput. Methods Appl. Mech. Engrg. 199 (5–8) (2009) 264–275.
[28] A.N. Brooks, T.J.R. Hughes, Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations, Comput. Methods Appl. Mech. Engrg. 32 (1982) 199–259.