Graphical Models 106 (2019) 101042

Contents lists available at ScienceDirect

Graphical Models

journal homepage: www.elsevier.com/locate/gmod

de Boor-like evaluation algorithm for Analysis-suitable T-splines

Hongmei Kang [a], Xin Li [b,*]

[a] Soochow University, No.1 Road Shizi, Suzhou, Jiangsu, China
[b] University of Science and Technology of China, No.96 Road Jinzhai, Hefei, Anhui, China

ARTICLE INFO

Keywords:
Analysis-suitable
T-splines
de Boor Evaluation
Control polygons
Bézier extraction

ABSTRACT

Analysis-suitable T-splines form a practically useful subset of T-splines. They maintain the design flexibility of T-splines with an efficient and highly localized refinement capability, while preserving the important analysis-suitable mathematical properties of the NURBS basis. The present paper proposes a new evaluation algorithm for analysis-suitable T-splines. The algorithm is based on the control polygon directly and it reduces both time and storage cost comparing with Bézier extraction.

1. Introduction

T-splines [1,2] have been used to overcome many limitations inherent in the industry standard NURBS representation, such as watertightness [1,3], trimmed NURBS conversion [4] and local refinement [2,5]. These capabilities make T-splines attractive both for CAD and for applications in iso-geometric analysis (for short, IGA), which uses the smooth spline basis that defines the geometry as the basis for analysis. IGA is introduced in [6] and described in detail in [7]. However, all T-splines are not suitable as a basis for IGA since they are not always linear independent [8]. Thus an important development in the evolution of IGA was the advent of analysis-suitable T-splines(for short, AS T-splines), a mild topologically restricted subset of T-splines. AS T-splines are optimized to meet the needs both for design and analysis [5,9]. Such T-splines inherit all the good properties from T-splines, such as watertightness, NURBS compatible, convex hull, and affine invariant. Unlike the general T-splines, such T-splines are guaranteed to be locally linearly independent [9], the polynomial blending functions for such T-splines sum identically to one for an admissible T-mesh [10] and the T-spline space can be characterized in terms of piecewise polynomials [11]. Furthermore, algorithms have been devised whereby local refinement of such T-splines is well contained [5].

Among all the good properties of T-splines, there is one key property which is not well developed, i.e., the evaluation algorithm directly based on the T-spline control grid, owing to the topological complexity of the T-grid. The current evaluation of T-spline is based on Bézier extraction, which first represents the T-spline with a set of Bézier patches and use de Casteljau algorithm [12]. Recently, Yang Zhang [13] proposed a subset of AS T-splines, called de Boor-suitable T-splines, which can be applied de Boor algorithm directly on T-spline control grid. Their proposed de Boor-suitable T-spline requires the control points on each unit element of the underlying AS T-mesh span exactly four columns (or rows) and the control points on the same column (or row) should have the same horizontal (or vertical) local knot vector. In this paper, based on an important observation, we develop a de Boor-like evaluation algorithm for analysis-suitable T-splines. In order to evaluate a given element, we first find the control points associated with the nonvanishing T-spline blending functions on the element, then we modify the control points such that they can be applied de Boor algorithm. When the control points span four columns or rows, the control points are only scaled and we do not need auxiliary points. Otherwise, several control points are modified such that the control points span four columns or rows by knot insertion. Comparing to [13], the conditions imposed on the control points are relaxed and the number of modified control points is far fewer than that of traditional Bézier extraction.

2. Preliminaries

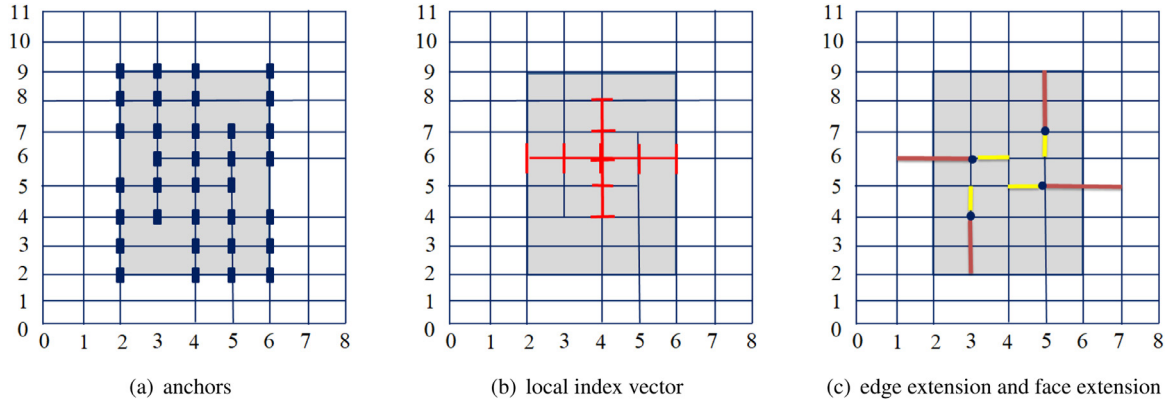
In the section, we will introduce some basic notations and preliminary results for bicubic analysis suitable T-splines [11,14,15] and give a brief introduction of de Boor algorithm of bicubic B-splines.

2.1. Index T-mesh

Similar to the approach of Bazilevs et al. [15], we define T-splines based on the T-meshes in the index domain which are referred as *index T-meshes*. A T-mesh $\mathcal{T}$ for a bicubic T-spline is a connection of all the elements of a rectangular partition of the index domain $[0, c+3] \times [0, r+3]$, where all rectangle corners (or vertices) have integer coordinates.

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

https://doi.org/10.1016/j.gmod.2019.101042
Received 29 September 2018; Received in revised form 1 April 2019; Accepted 9 September 2019
Available online 11 September 2019



(a) anchors (b) local index vector (c) edge extension and face extension

Fig. 1. (a) Anchors for a bicubic T-spline, (b) the local index vector for one bicubic T-spline blending function and (c) extensions of T-junctions.

Three types of elements are

- Vertex: Vertex of a rectangle, denoted as (ξ_i, η_i) or $\xi_i \times \eta_i$.
- Edge: A line segment connecting two vertices in the T-mesh and no other vertices lying in the interior, denoted as $[\xi_j, \xi_k] \times \{\eta_i\}$ or $\{\xi_i\} \times [\eta_j, \eta_k]$ for a horizonal or vertical edge.
- Face: A rectangle where no other edges and vertices in the interior, denoted as $[\xi_i, \xi_j] \times [\eta_i, \eta_j]$ or $(\xi_i, \xi_j) \times (\eta_i, \eta_j)$. The second notation is for an open face.

The valence of a vertex is the number of edges connecting to this vertex. For the interior vertices, we only allow valence three (called *T-junctions*) or four vertices (called *cross vertices*). We adopt the notation $\vdash$, $\dashv$, $\perp$, and $\top$ to indicate the four possible orientations for the T-junctions. The T-junctions of type $\vdash$ and $\dashv$ are called *horizontal T-junctions*, and T-junctions of type $\perp$ and $\top$ are called *vertical T-junctions*.

Denote the active region as a rectangle region $[2, c+1] \times [2, r+1]$. An *anchor* is a point in the index T-mesh which corresponds one blending function. For bicubic T-splines, an anchor is exactly a vertex in the active region of the T-mesh. The active region carries the anchors that will be associated the blending functions while the other indices will be needed for the definition of the blending functions when the anchors close to the boundary. Fig. 1(a) shows an index T-mesh of degree (3,3), where the small solid rectangles are used to label anchors and the grey domain is the active region in a T-mesh.

For the i-th anchor A_i, we define a local index vector $\vec{\xi}_i \times \vec{\eta}_i$ which is used to define the blending function $T_i(s, t)$. The values of $\vec{\xi}_i = [\xi_i^0, \cdots, \xi_i^4]$ and $\vec{\eta}_i = [\eta_i^0, \cdots, \eta_i^4]$ are determined as follows. Starting from A_i, we shoot a ray in the s and t direction traversing the T-mesh and collect a total of 5 and 5 knot indices to form $\vec{\delta}_i$ and $\vec{\eta}_i$, as shown in Fig. 1(b). Then the T-spline blending function $T_i(s, t)$ is defined as

$$T_i(s,t) = N[\xi_i^0, \xi_i^1, \xi_i^2, \xi_i^3, \xi_i^4](s) \cdot N[\eta_i^0, \eta_i^1, \eta_i^2, \eta_i^3, \eta_i^4](t),$$

where $N[\xi_i^0, \xi_i^1, \xi_i^2, \xi_i^3, \xi_i^4](s)$ and $N[\eta_i^0, \eta_i^1, \eta_i^2, \eta_i^3, \eta_i^4](t)$ are the cubic B-spline functions in $s-$direction and $t-$direction respectively.

2.2. Analysis-suitable T-splines

The *extension* of a T-junction is defined as a closed line segment associated with the T-junction. For a i-th T-junction (ξ_i, η_i) of type $\vdash$ or $\dashv$, the extension is the line segment $[\xi_{i_0}, \xi_{i_1}] \times \{\eta_i\}$. When the T-junction is of type $\vdash$, ξ_{i_0} and ξ_{i_1} are determined such that the edges $[\xi_{i_0}, \xi_i) \times \{\eta_i\}$ have 1 intersection with the T-mesh and the edges $(\xi_i, \xi_{i_1}] \times \{\eta_i\}$ have 2 intersections with the T-mesh. Here $[\xi_{i_0}, \xi_i) \times \{\eta_i\}$ is called the *edge extension* while $(\xi_i, \xi_{i_1}] \times \{\eta_i\}$ is called the *face extension*. For a T-junction of type $\dashv$, we can similarly define the extension except the number of intersections are exchanged. Also, we can define the extensions for the other kinds of T-junctions $\perp$, $\top$. The extension of a T-junction of type $\vdash$ or $\dashv$ is called the *horizontal extension* (HE for short) and that of a T-junction of type $\perp$ or $\top$ is called the *vertical extension* (VE for short). In Fig. 1(c), the yellow line segments are edge extensions and red line segments are face extensions.

Definition 2.1. For a bicubic T-spline, a T-mesh is called analysis-suitable (for short, AS T- mesh) if the extensions for all the T-junctions $\vdash$ and $\dashv$, don't intersect the extensions for all the T-junctions $\perp$ and $\top$. A T-spline defined on an analysis-suitable T-mesh is called analysis-suitable T-spline, for short AS T-spline [5,11].

AS T-splines are a subset of T-splines with the constraint that horizontal extensions do not intersect with vertical extensions. Such a constraint makes the T-vertices in an AS T-mesh be separated up to some extent. Thus AS T-splines possess some nice properties that T-splines do not have. The blending functions of AS T-splines are linearly independent for all the knots [10,11,14,16]. The basis constitutes a partition of unity [11]. AS T-splines obey the convex hull property and can be locally refined [5]. There are exactly 16 basis functions on a Bézier element [17]. In all, AS T-mesh significantly simplifies the complexity of arbitrary T-mesh and makes it possible to apply de Boor algorithm.

2.3. de Boor algorithm

A B-spline curve $c(s)$ of degree p is defined as

$$c(s) = \sum_{i=0}^{m} P_i N_i^p(s),$$

where $N_i^p(s)$ are the B-splines basis function defined over a knot vector $U = \{s_0, s_1, s_2, \cdots, s_{m+p+1}\}$ and $P_i \in \mathbb{R}^d$ are control points. Suppose $s \in [s_i, s_{i+1}]$, $i = p, p+1, \cdots, m$, then the curve point $c(s) = P_i^p$ is evaluated by de Boor algorithm as follows.

$$\begin{cases} P_j^0(s) = P_j, & j = i-p, i-p+1, \cdots, i, \\ P_j^r(s) = (1-\alpha_j^r) P_{j-1}^{r-1}(s) + \alpha_j^r P_j^{r-1}(s), & \alpha_j^r = \dfrac{s - s_j}{s_{j+p+1-r} - s_j}, \\ \qquad r = 1, 2, \cdots, p, & j = i-p+r, i-p+r+1, \cdots, i. \end{cases} \tag{1}$$

A tensor product B-spline surface of degree (p, q) is defined as

$$S(s,t) = \sum_{i=0}^{m} \sum_{j=0}^{n} P_{ij} N_i^p(s) N_j^q(t),$$

where $P_{ij} \in \mathbb{R}^d$ are control points, $N_i^p(s)$ and $N_i^q(t)$ are B-splines basis functions defined over knots vectors $U = \{s_0, s_1, s_2, \cdots, s_{m+p+1}\}$ and $V = \{t_0, t_1, t_2, \cdots, t_{n+q+1}\}$ respectively. Suppose $(s, t) \in [s_k, s_{k+1}] \times [t_l, t_{l+1}]$, the surface point $S(s, t)$ is evaluated by applying de Boor algorithm in one direction firstly, then in the other direction. Fig. 2 shows the evaluation framework of bicubic B-spline surfaces by de Boor algorithm, the surface point $S(s, t) = P_{k,l}^{3,3}$.

The de Boor algorithm provides a fast and numerically stable way for evaluating B-splines. Instead of evaluating each B-spline basis functions,

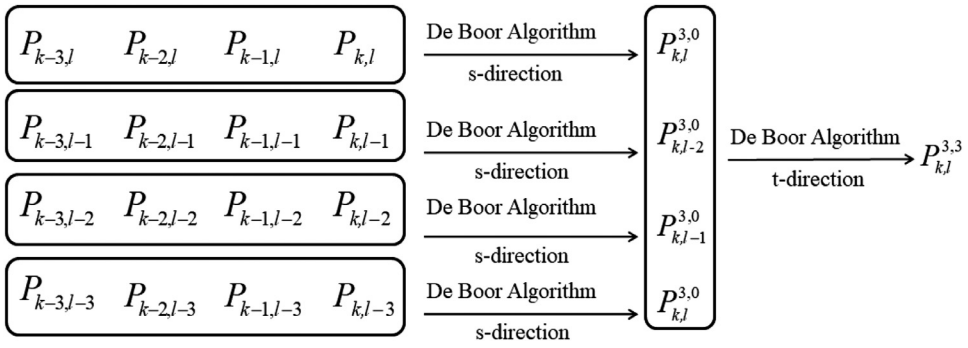


Fig. 2. de Boor algorithm of bicubic tensor product B-splines surfaces.

the curve/surface point is evaluated by calculating a linear combination of control points recursively. We are going to extend de Boor algorithm in AS T-splines.

3. de Boor-like evaluation algorithm

For T-splines, the non-vanishing basis functions on an element do not have a global tensor product structure owing to the T-junctions. Therefore de Boor algorithm can not be applied directly. In the current existing algorithm, a T-spline is evaluated by Bézier extraction [18,19]. That is, a T-spline is firstly represented as element-wise Bernstein functions and then is evaluated by de Casteljau's algorithm. However such a work-around is often time- and memory-consuming. In this paper, we propose an algorithm similar to de Boor's for evaluating AS T-splines by calculating control points instead of calculating the Bézier coordinates, we call such an algorithm as de Boor-like evaluation algorithm. In this paper, we focus on the de Boor-like evaluation of bicubic AS T-splines as bicubic AS T-splines are the most common ones used in application.

3.1. An observation

We firstly introduce several concepts which are necessary for stating the observation.

Definition 3.1. A basis function is called a non-vanishing basis function on an element if it does not vanish on this element. The corresponding control point and anchor associated with this non-vanishing basis function is called a non-vanishing control point and a non-vanishing index, respectively.

For bicubic AS T-splines, there is a one-to-one correspondence among control points, basis functions and anchors. Thus we do not distinguish these three concepts in the following context.

Definition 3.2. An extended T-mesh $\mathcal{T}_{ext}$ is formed by augmenting a T-mesh $\mathcal{T}$ with face extensions of all T-junctions in $\mathcal{T}$. For a vertex $v \in \mathcal{T}_{ext}$, but $v \notin \mathcal{T}$, it is called a *HE*-vertex if it is the intersection of a horizontal face extension and a vertical edge in $\mathcal{T}$; it is called a *VE*-vertex if it is the intersection of a vertical face extension and a horizontal edge in $\mathcal{T}$.

For tensor-product B-spline surfaces, there are 16 non-vanishing basis functions on an element. And the corresponding non-vanishing indices span exactly four rows and four columns. For bicubic AS T-splines, there are exactly 16 non-vanishing basis functions on a Bézier element [9]. But the non-vanishing indices do not span four rows and four columns. Fortunately, we observed that the non-vanishing indices must span two rows or columns. This observation inspires de Boor-like algorithm. We will conclude this in Theorem 3.1.

Theorem 3.1. *Given an AS T-spline, suppose $\mathcal{T}$ is the underlying index T-mesh. For an element $F = [s_k, s_{k+1}] \times [t_l, t_{l+1}]$ in the extended T-mesh $\mathcal{T}_{ext}$, among the 16 non-vanishing anchors on F, either*

(1) four anchors are covered by the line $s = s_k$ and four anchors are covered by $s = s_{k+1}$; or

(2) four anchors are covered by the line $t = t_l$ and four anchors are covered by $t = t_{l+1}$.

Proof. Suppose the four vertices of F are v_1, v_2, v_3, v_4 arranged in anti-clock order and v_2 is the left bottom vertex, see Fig. 3 for a reference. Denote $K_1 = \{v_1, v_2, v_3, v_4\}$ and v_i may not necessarily be in $\mathcal{T}$, for example v_3 in Fig. 3(a).

We introduce a case indicator I for the set K_1. If there exists one *HE*-vertex in K_1, then $I = 0$; if there exists one *VE*-vertex in K_1, then $I = 1$; otherwise $I = 2$. Such a case indicator is well-defined since the *HE*-vertex and the *VE*-vertex should not occur simultaneously in a element according to the definition of AS T-meshes.

If $v_{i_0} \in K_1$ is a *HE*-vertex, then among the vertical neighbors of v_{i_0}, the one in $\mathcal{T}$ but not in K_1 replaces v_{i_0}. If $v_{i_0} \in K_1$ is a *VE*-vertex, then among the horizontal neighbors of v_{i_0}, the one in $\mathcal{T}$ but not in K_1 replaces v_{i_0}. For example in Fig. 3(a), v_3 is replaced by its right neighbor in Fig. 3(b). In this way, the four vertices in K_1 are all in $\mathcal{T}$. For convenience, we still use v_1, v_2, v_3, v_4 to denote the vertices in K_1.

If $I = 0$, the vertical 1-neighboring vertices in $\mathcal{T}$ of v_1, v_2, v_3, v_4 exist, therefore denote these 1-neighboring vertices as $K_2 = \{v_5, v_6, v_7, v_8\}$. If $I = 1$, the horizontal 1-neighboring vertices in $\mathcal{T}$ of v_1, v_2, v_3, v_4 exist, denote these 1-neighboring vertices as $K_2 = \{v_5, v_6, v_7, v_8\}$. If $I = 2$, both the horizontal 1-neighboring vertices and vertical 1-neighboring vertices in $\mathcal{T}$ of v_1, v_2, v_3, v_4 exist, here we denote the horizontal neighboring vertices as $K_2 = \{v_5, v_6, v_7, v_8\}$. For example in Fig. 3(b), v_5 and v_6 are the right neighbors of v_1 and v_2 respectively and v_7 and v_8 are the right neighbors of v_3 and v_4 respectively.

It is easily proved that the corresponding AS T-spline blending function associated with v_1, v_2, $\cdots$, v_7, v_8 do not vanish on F. And

- when $I = 0$, v_5, v_1, v_2, v_6 are covered by the straight line $s = s_k$ and v_8, v_4, v_3, v_7 are covered by the straight line $s = s_{k+1}$.
- when $I = 1$, v_5, v_1, v_4, v_8 are covered by the straight line $t = t_{l+1}$ and v_6, v_2, v_3, v_7 are covered by the straight line $t = t_l$.
- when $I = 2$, v_5, v_1, v_2, v_6 are covered by the straight line $s = s_k$ and v_8, v_4, v_3, v_7 are covered by the straight line $s = s_{k+1}$. Meanwhile, v_5, v_1, v_4, v_8 are covered by the straight line $t = t_{l+1}$ and v_6, v_2, v_3, v_7 are covered by the straight line $t = t_l$. □

3.2. Finding non-vanishing basis functions

Based on Theorem 3.1, we propose the following algorithm for finding non-vanishing basis functions on a given element F.

Input $\mathcal{T}$ is the underlying T-mesh of a given AS T-spline. F is a given element in $\mathcal{T}_{ext}$ and has four vertices v_1, v_2, v_3, v_4 arranged in anti-clock order with the assumption that v_2 is the left bottom vertex.

Output 16 non-vanishing indices on F and the case indicator I.

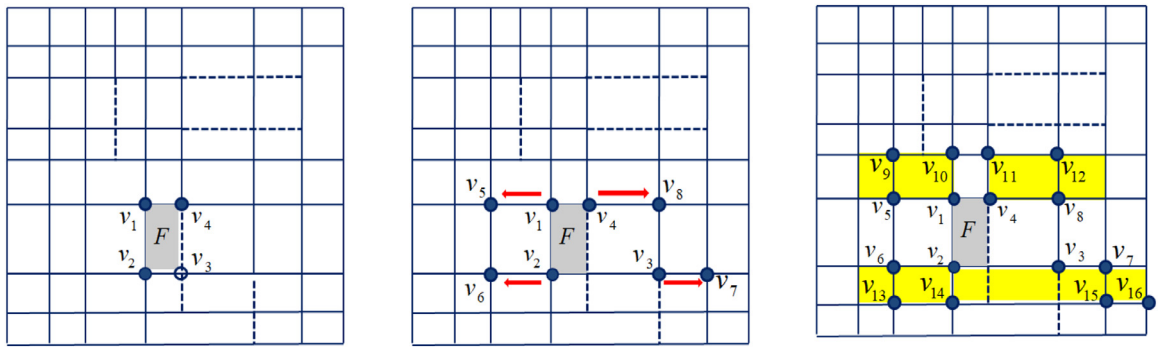


Fig. 3. Find non-vanishing basis functions on element F.

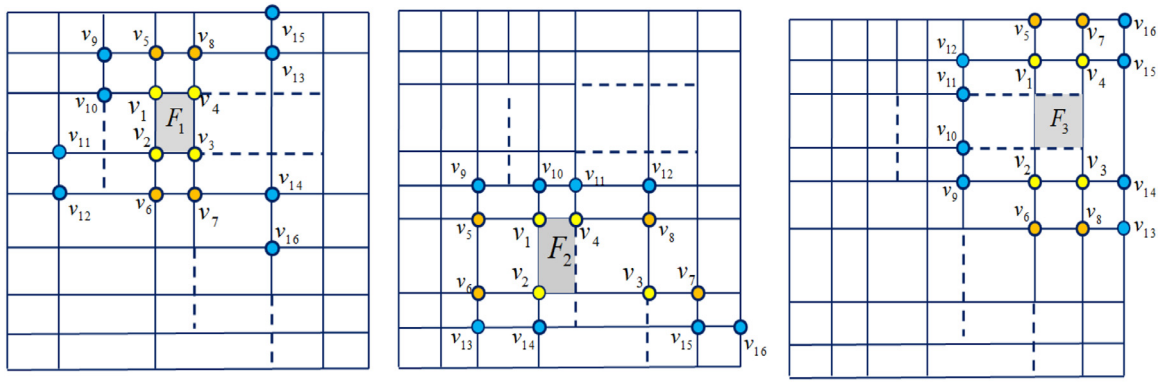


Fig. 4. Find non-vanishing basis functions on element F_1, F_2 and F_3.

1. update $K_1 = \{v_1, v_2, v_3, v_4\}$. If there is a *HE*-vertex in K_1, then set $I = 0$. If there is a *VE*-vertex in K_1, then set $I = 1$. Otherwise $I = 2$.
   If $v_{i_0} \in K_1$ is a *HE*-vertex, then search the 1-neighboring vertices of v_{i_0} in the vertical direction pointing to the outside of F until it is a vertex in $\mathcal{T}$ and replace v_{i_0}. If $v_{i_0} \in K_1$ is a *VE*-vertex, then search the 1-neighboring vertices of v_{i_0} in the horizontal direction pointing to the outside of F until it is a vertex in $\mathcal{T}$ and replace v_{i_0}.
2. find $K_2 = \{v_5, v_6, v_7, v_8\}$. If $I = 0$, the closest vertical neighboring vertices (outside of F) of v_i, $i = 1, 2, 3, 4$ are denoted as $K_2 = \{v_5, v_6, v_7, v_8\}$. If $I = 1$ or $I = 2$, change 'vertical' as 'horizontal'.
3. find $K_3 = \{v_9, v_{10}, v_{11}, v_{12}, v_{13}, v_{14}, v_{15}, v_{16}\}$.When $I = 0$, find the left adjacent elements of v_5 and v_6 and collect them in a set K_f. If elements in K_f are less than 4, the left elements of v_1 and v_2 are included in K_f. Collect the endpoints on the left edges of the elements in K_f without repetition. Denote the endpoints associated with non-vanishing T-spline blending functions on F as $\{v_9, v_{10}, v_{11}, v_{12}\}$. Similarly search the right adjacent elements of v_8, v_4, v_3, v_7 and find $\{v_{13}, v_{14}, v_{15}, v_{16}\}$. When $I = 1$ or $I = 2$, find the top adjacent elements of v_5, v_1, v_4, v_8 and find the down adjacent elements of v_6, v_2, v_3, v_7 similarly as $I = 0$. In Fig. 3(c), the elements in K_f are colored by yellow.
4. $K = K_1 \cup K_2 \cup K_3 = \{v_1, v_2, \cdots, v_{16}\}$ is the output vertices on F.

Fig. 4 shows the results of finding the non-vanishing anchors on three elements, where the anchors in K_1 are marked by yellow solid circles, the anchors in K_2 are marked by orange solid circles, and the anchors in K_3 are marked by blue solid vertices.

3.3. Preprocessing control points

In order to evaluate an AS T-spline on an element by de Boor algorithm, the non-vanishing anchors should span four columns or four rows and the non-vanishing control points on each column or row should have the same local knot vector in s-direction or t-direction. Thus the non-vanishing control points have to be modified by knot insertion. With the help of Theorem 3.1, this process can be simplified significantly. We present this process as follows.

Denote by P_{ij}, $i, j = 1, 2, 3, 4$ as the non-vanishing control points on an element $F = [s_3, s_4] \times [t_3, t_4]$. Assume the associated anchors of P_{12}, P_{22}, P_{32}, P_{42} are covered by the vertical line $s = s_3$, and the associated anchors of P_{13}, P_{23}, P_{33}, P_{43} are covered by the vertical line $s = s_4$. Let $U = \{s_0, s_1, s_2, \cdots, s_7\}$ be the referring knot vector in $s-$direction with the assumption that there is no other knot between s_k and s_{k+1}, $k = 0, 1, \cdots, 6$. For the control points on the jth row $P_{j1}, P_{j2}, P_{j3}, P_{j4}$, the associated local knot vectors in s-direction are

$$(x_0, x_1, x_2, s_3, s_4), \quad (x_1, x_2, s_3, s_4, x_5), \quad (x_2, s_3, s_4, x_5, x_6), \quad (s_3, s_4, x_5, x_6, x_7),$$

respectively. Here $x_1 \le s_1 \le x_2 \le s_2 < s_3 < s_4 < s_5 \le x_5 \le x_6$. We are going to see how the control points P_{ij}, $i, j = 1, 2, 3, 4$ are updated in order to match the referring knot vector. Referring to U, there are four cases needed to be considered:

$$(a)\ x_2 = s_2,\ x_5 = s_5; (b)\ x_2 = s_2,\ x_5 \neq s_5; (c)\ x_2 \neq s_2,\ x_5 = s_5;$$
$$(d)\ x_2 \neq s_2,\ x_5 \neq s_5.$$

Case (b) and (c) can be integrated into case (d). And case (d) can be transformed into case (a) by inserting knots s_2 and s_5. Thus we only need to consider case (a) and (d). For case (a), it means the non-vanishing

control points span four columns. By knot insertion, it has

$$\begin{aligned}
N[x_0,x_1,s_2,s_3,s_4](s) &= N[x_0,x_1,s_1,s_2,s_3](s)\frac{s_1-x_0}{s_3-x_0}\\
&\quad + N[x_1,s_1,s_2,s_3,s_4](s)\frac{s_4-s_1}{s_4-x_1}, \quad x_1<s_1,\\
N[x_1,s_2,s_3,s_4,s_5](s) &= N[x_1,s_1,s_2,s_3,s_4](s)\frac{s_1-x_1}{s_4-x_1}\\
&\quad + N[s_1,s_2,s_3,s_4,s_5](s), \quad x_1<s_1,\\
N[s_2,s_3,s_4,s_5,x_6](s) &= N[s_2,s_3,s_4,s_5,s_6](s)\\
&\quad + N[s_3,s_4,s_5,s_6,x_6](s)\frac{x_6-s_6}{x_6-s_3}, \quad x_6>s_6,\\
N[s_3,s_4,s_5,x_6,x_7](s) &= N[s_3,s_4,s_5,s_6,x_6](s)\frac{s_6-s_3}{x_6-s_3}\\
&\quad + N[s_4,s_5,s_6,x_6,x_7](s)\frac{x_7-s_6}{x_7-s_4}, \quad x_6>s_6.
\end{aligned}$$

Then for this case the control points are updated as follows

$$\begin{aligned}
&P_{j1}\leftarrow \alpha_1 P_{j1}+(1-\alpha_1)P_{j2}, \quad \alpha_1=\frac{s_4-s_1}{s_4-x_1},\\
&P_{j4}\leftarrow \alpha_4 P_{j3}+(1-\alpha_4)P_{j4}, \quad \alpha_4=\frac{x_6-s_6}{x_6-s_3}.
\end{aligned} \tag{2}$$

For case (d), it means the non-vanishing control points only span two columns. So we insert the knot s_2 and s_5 and it has

$$\begin{aligned}
N[x_0,x_1,x_2,s_3,s_4](s) &= N[x_0,x_1,x_2,s_2,s_3](s)\frac{s_2-s_0}{s_3-s_0}\\
&\quad + N[x_1,x_2,s_2,s_3,s_4](s)\frac{s_4-s_2}{s_4-x_1},\\
N[x_1,x_2,s_3,s_4,x_5](s) &= N[x_1,x_2,s_2,s_3,s_4](s)\frac{s_2-x_1}{s_4-x_1}\\
&\quad + N[x_2,s_2,s_3,s_4,s_5](s)\frac{x_5-s_2}{x_5-x_2}\\
&\quad + N[s_2,s_3,s_4,s_5,x_5](s)\frac{x_5-s_5}{x_5-x_2},\\
N[x_2,s_3,s_4,x_5,x_6](s) &= N[x_2,s_2,s_3,s_4,s_5](s)\frac{s_2-x_2}{x_5-x_2}\\
&\quad + N[s_2,s_3,s_4,s_5,x_5](s)\frac{s_5-s_2}{x_5-x_2}\\
&\quad + N[s_2,s_3,s_4,s_5,x_5](s)\frac{x_6-s_5}{x_6-s_3},\\
N[s_3,s_4,x_5,x_6,s_7](s) &= N[s_3,s_4,s_5,x_5,x_6](s)\frac{s_5-s_3}{x_6-s_3}\\
&\quad + N[s_4,s_5,x_5,s_7](s)\frac{s_7-s_5}{s_7-s_4}.
\end{aligned}$$

Then for this case the control points are updated as follows

$$\begin{aligned}
&P_{j1}\leftarrow \beta_1 P_{j1}+(1-\beta_1)P_{j2}, \quad P_{j2}\leftarrow \beta_2 P_{j2}+(1-\beta_2)P_{j3},\\
&P_{j3}\leftarrow \beta_3 P_{j2}+(1-\beta_3)P_{j3}, \quad P_{j4}\leftarrow \beta_4 P_{j3}+(1-\beta_4)P_{j4}.
\end{aligned} \tag{3}$$

with

$$\beta_1=\begin{cases}\frac{s_4-s_2}{s_4-x_1}, & x_2\neq s_2\\ 1, & x_2=s_2,\end{cases} \quad \beta_2=\frac{s_5-s_2}{s_5-x_2}, \quad \beta_3=\frac{x_5-s_5}{x_5-x_2},$$

$$\beta_4=\begin{cases}\frac{x_6-s_5}{x_6-s_3}, & x_5\neq s_5\\ 0, & x_5=s_5\end{cases} \tag{4}$$

According to the above analysis, we present the way of preprocessing control points. We do the preprocessing column by column. For a control point P_{j1} on the first column assumed with the $s-$direction knot vector $(x_0, x_1, x_2, s_3, s_4)$, we have to consider three cases in order to match the referring knot vector $(s_0, s_1, s_2, s_3, s_4)$: (1) $x_2 = s_2$; (2) $x_2 = s_1$; and (3) $x_2 = s_0$. According to (2) and (3), the updates associated with these three cases can be integrated into the following form:

$$\begin{aligned}
P_{j1}\leftarrow \lambda_1 P_{j1}+\sigma_1 P_{j2}, \quad \lambda_1&=\frac{s_4-s_2}{s_4-x_1}\frac{s_4-s_1}{s_4-x_2}, \quad \sigma_1=\frac{s_4-s_1}{s_4-x_2}\frac{s_2-x_1}{s_4-x_1}\\
&\quad +\frac{s_5-s_2}{s_5-x_2}\frac{s_1-x_2}{s_4-x_2}.
\end{aligned} \tag{5}$$

Similarly, for the control points P_{j2}, P_{j3} and P_{j4} associated with the $s-$direction knot vector $(x_1, x_2, s_3, s_4, s_5)$, $(x_1, x_2, s_3, s_4, s_5)$ and $(x_1, x_2, s_3, s_4, s_5)$ respectively, they are updated as follows,

$$P_{j2}\leftarrow \lambda_2 P_{j2}+\sigma_2 P_{j3}, \quad \lambda_2=\frac{s_5-s_2}{s_5-x_2}, \quad \sigma_2=1-\lambda_2, \tag{6}$$

$$P_{j3}\leftarrow \lambda_3 P_{j2}+\sigma_3 P_{j3}, \quad \lambda_3=\frac{x_5-s_5}{x_5-x_2}, \quad \sigma_3=1-\lambda_3, \tag{7}$$

$$\begin{aligned}
P_{j4}\leftarrow \lambda_4 P_{j3}+\sigma_4 P_{j4}, \quad \lambda_4&=\frac{s_5-x_2}{x_5-x_2}\frac{x_5-s_6}{x_5-s_3}+\frac{x_6-s_5}{x_6-s_3}\frac{s_6-s_3}{x_5-s_3},\\
\sigma_4&=\frac{s_5-s_3}{x_6-s_3}\frac{s_6-s_3}{x_5-s_3}.
\end{aligned} \tag{8}$$

Notice that we do not need to update all the non-vanishing control points. The control points on a column are updated only when the control points on this column do not have the same knot vector in $s-$direction. Actually there are at most 6 control points needed to be updated. This can be explained as follows. When the non-vanishing anchors span four columns or four rows exactly, the control points on the first and fourth column (row) may be updated according to (2). Notice that we can choose the knot vector of one control point on a column (row) as a referring knot vector, thus there are at most 6 control points needed to be updated for this case. When the non-vanishing anchors neither span four columns nor four rows, we discuss this the update of control points according to the case indicator. If $I = 0$, then either the control points on the first and second column are needed to be updated or the control points on the third column and fourth column are needed to be updated. Otherwise there exists an horizontal T-vertex whose extension intersects with a vertical extension crossing this element, which conflicts the definition of AS T-meshes. Similarly, if $I = 1$, then either the control points on the first row and second row are needed to be updated or the control points on the third and fourth row are needed to be updated. When $I = 2$, if the control points on four columns are needed to be updated, then there exists two rows on which the control points do not need an update. This is guaranteed by the definition of AS T-meshes.

Theorem 3.2. *For bicubic AS T-splines, it needs to update 6 control points at most for an element in the preprocessing.*

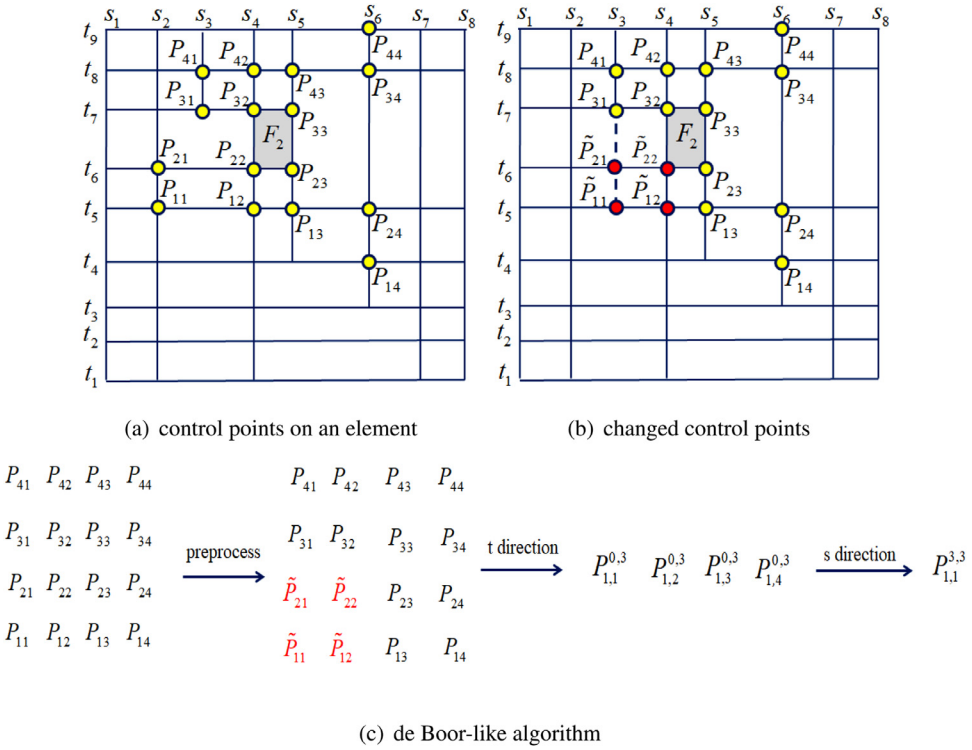
3.4. de Boor-like evaluation algorithm

Now we are ready to present the algorithm for evaluating AS T-splines. There are three steps:

1. Find non-vanishing basis functions and the case indicator I by the algorithm presented in Section 3.2.
2. Preprocess the non-vanishing control points on each column or row by one of (5), (6), (7) and (8) as presented in 3.3.
3. Apply de Boor algorithm firstly in one direction then in another direction according to the case indicator I.

This evaluation algorithm is very similar to de Boor algorithm but with a preprocessing of the control points. We call such an evaluation algorithm as de Boor-like algorithm. In the following, two examples are demonstrated to explain the work-flow of this algorithm.

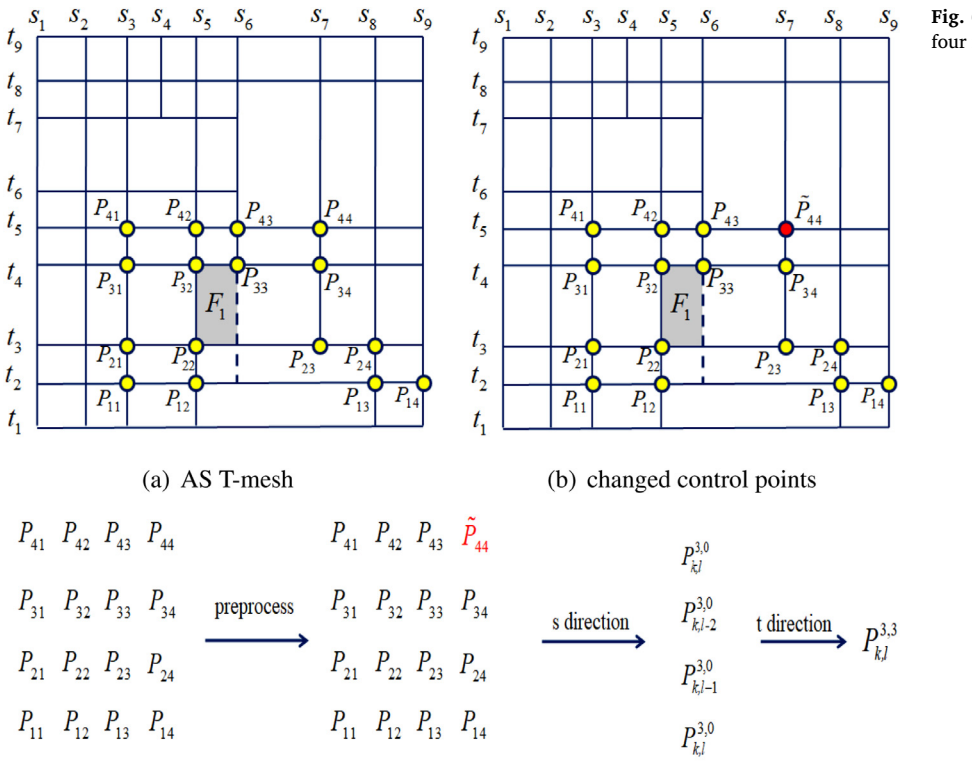
For the element shown in Fig. 5(a), the control points span three columns, thus we have to update the control points on the first column and second column. By the formulas (5) and (6), P_{11} and P_{21} are updated with the weights $\lambda_1=\frac{s_5-s_3}{s_5-s_1}, \sigma_1=\frac{s_3-s_1}{s_5-s_1}$, P_{12} and P_{22} are updated with the weights $\lambda_2=\frac{s_6-s_3}{s_6-s_2}, \sigma_2=1-\lambda_2$. For this AS T-mesh, there are four control points updated. Fig. 5(c) shows the process of de Boor-like algorithm.



(a) control points on an element (b) changed control points

(c) de Boor-like algorithm

Fig. 5. de Boor-like algorithm.



(a) AS T-mesh (b) changed control points

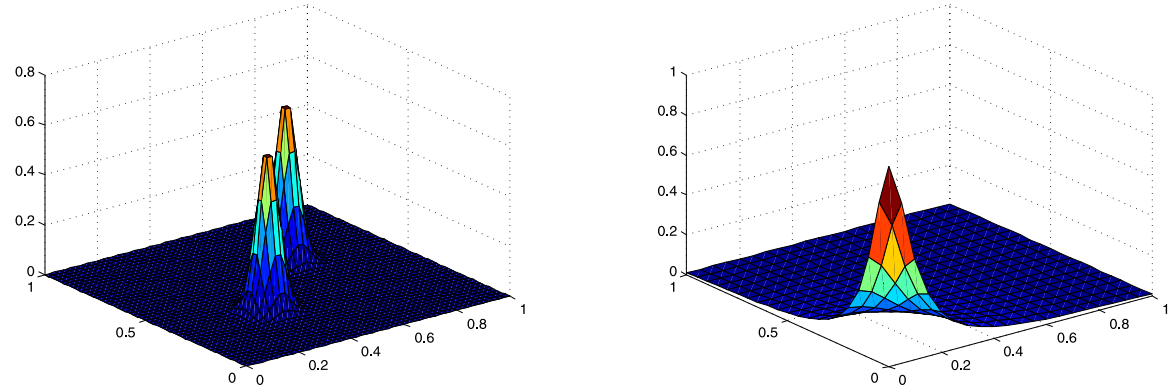
(c) de Boor-like algorithm

Fig. 6. de Boor-like algorithm for control points span four rows.

For the element shown in Fig. 6(a), the control points span four columns, but the control points on the fourth column do not have the same knot vector, thus the control points on the fourth column are updated. By the formulas (8), P_{44} is updated with the weights $\lambda_4 = \frac{t_6 - t_3}{t_8 - t_3}, \sigma_4 = \frac{t_8 - t_6}{t_8 - t_3}$. For this AS T-mesh, there is only one control point updated. Fig. 5(c) shows the process of de Boor-like algorithm.

4. Numerical experiments

In this section, we are going to compare de Boor-like evaluation algorithm with the Bézier extraction evaluation and the one proposed in [13] on several AS T-meshes produced in solving PDEs by AS T-splines. We compare the number of the control points needed to be updated of



(a) Example 1 (b) Example 2

Fig. 7. The exact solutions shown in Example 4.1 and 4.2.

de Boor-like algorithm and Bézier extraction. And we give a statistic of those elements which do not satisfy the conditions proposed in [13] in a given AS T-mesh. In [13], the proposed de Boor-suitable T-splines (DS T-splines for short) have to satisfy two conditions: the non-vanishing control points on each unit element have to span exactly four columns (or rows) and the control points on the same column (or row) should have the same horizontal (or vertical) local knot vector. For convenience, we call those elements which do not satisfy these two conditions as violated elements.

4.1. Solving PDEs by AS T-splines

The AS T-meshes in the experiments are produced from solving the Poisson equation by AS T-splines. We briefly review the framework of isogeometric analysis based on AS T-splines, more details can be found in [15].

The Poisson equation is defined as

$$\begin{aligned} -\Delta u &= f \quad \text{in} \quad \Omega, \\ u &= g \quad \text{on} \quad \Gamma, \end{aligned} \tag{9}$$

where $\Omega \subseteq \mathbb{R}^2$ is a connected, bounded domain with a Lipschitz-continuous boundary Γ, f and h are square-integrable on Ω and Γ_N, respectively.

The physical domain Ω is parameterized by a global geometry function $G : (s,t) \in \Omega_0 = [0,1]^2 \to (x,y) \in \Omega$, defined as

$$G(s,t) = \sum_{i=1}^{m} P_i \frac{w_i T_i(s,t)}{\sum_{i=1}^{m} w_i T_i(s,t)}, \quad (s,t) \in \Omega_0,$$

where $P_i \in \mathbb{R}^2$, $T_i(s,t)$ is an AS T-splines basis function, $w_i \in \mathbb{R}, w_i > 0$ is a weight, m is the number of basis functions and Ω_0 is the parameter domain. A finite dimensional subspace V^h is defined as

$$V^h = span\{\hat{T}_i(x,y) |\ \hat{T}_i(x,y) = T_i \circ G^{-1}, \ \ \hat{T}_i(x,y)|_\Gamma = 0, i = 1, \cdots, n.\}.$$

The weak form solution of problem (9) is to seek u_h such that

$$\int_\Omega \nabla u_h \cdot \nabla v_h \, d\Omega = \int_\Omega f v_h \, d\Omega, \quad \forall v_h \in V_h.$$

The approximate solution u^h is written as $u^h(x,y) = \sum_{i=1}^{n} c_i \hat{G}_i(x,y) = \sum_{i=1}^{n} c_i T_i \circ G^{-1}$, with unknown coefficients $c_i, i = 1, 2, \cdots, n$. Thus the weak form solution is converted into solving a linear system with the unknown coefficients.

We use the local refinement of AS T-splines introduced in [5]. The local refinement of the numerical solution is refined based on posterior error estimation.

4.2. Comparison

We choose two examples to demonstrate the comparison: Example 4.1 and 4.2. In the first example, the exact solution has two peaks at $(\frac{13}{20}, \frac{13}{20})$ and $(\frac{7}{20}, \frac{7}{20})$. Thus the underling T-mesh should be refined heavily around these two peaks to capture the feature. The second exact solution has a sharp gradient around (0,0) and the underling T-mesh should be refined heavily around the origin. Fig. 7(a) and (b) shows the plots of these two exact solutions.

Example 4.1. The exact solution in (9) is chosen as

$$u(x,y) = \frac{2}{3e^{(20x-13)^2+(20y-13)^2}} + \frac{2}{3e^{(20x-7)^2+(20y-7)^2}},$$

and the right hand can be derived from (9). We solve this problem on an unit square $\Omega = [0,1] \times [0,1]$.

Example 4.2. The exact solution in (9) is chosen as $u(x,y) = \frac{0.01}{x^2+y^2+.01}$, and the right hand can be derived from (9). We solve this problem on an unit square $\Omega = [0,1] \times [0,1]$.

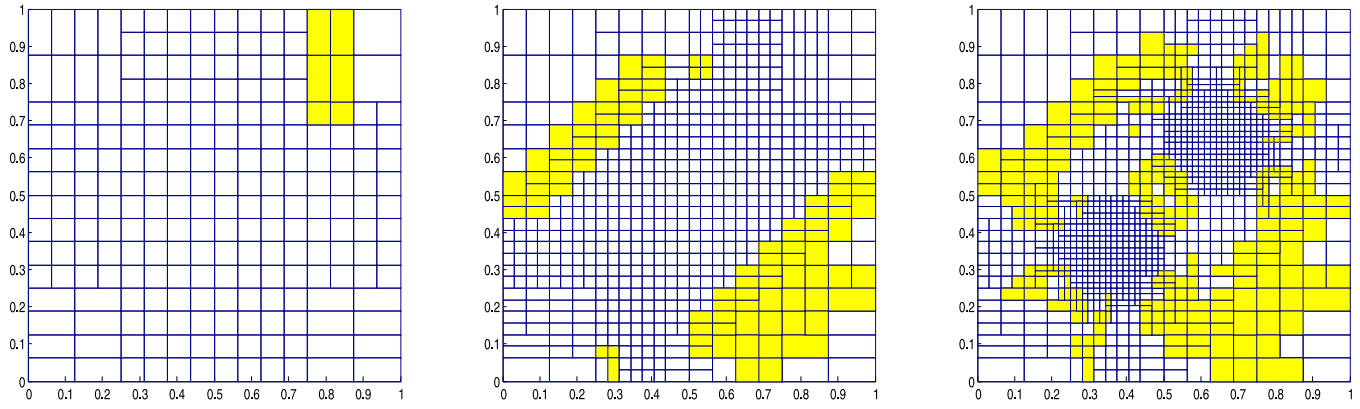
We start from a 9×9 tensor-product mesh for Example 4.1 and a 3×3 tensor-product mesh for Example 4.2. The corresponding refined T-meshes on the first three levels are shown in Figs. 8 and 9 respectively, where the elements shaded by yellow color are the violated elements.

Denote by N_{be} and N_{ve} as the number of Bézier elements and violated elements in an AS T-mesh. The number of the control points needed to be updated is denoted by N_c. In our algorithm, we need to update at most 6 control points for each violated element, thus we use $N_c = 6N_{ve}$ to give a upper bound of the updated control points for comparison. For Bézier extraction, it has exactly $N_c = 16N_e$. We summarize the numbers of Bézier elements, violated elements, updated control points of Example 4.1 and 4.2 in Tables 1 and 2 respectively.

From the statistics in Tables 1 and 2, it can be seen the number of updated control points in our algorithm is far fewer than that in Bézier extraction and the violated elements increase as the refinement level increases. Furthermore, we see the AS T-splines are not always DS T-splines in practice and our algorithm provides an efficient and economic way of dealing with violated elements in a sense.

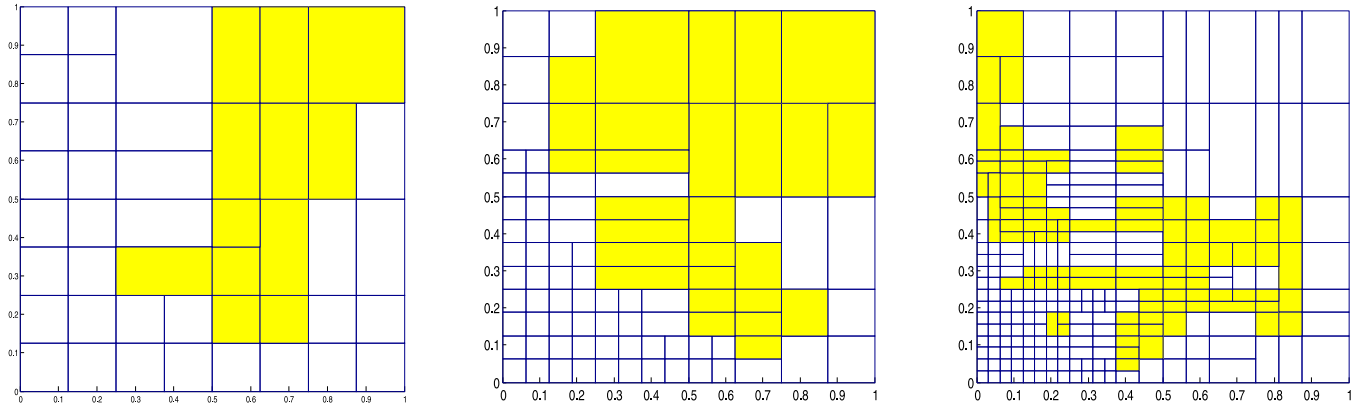
Table 1
Statistics of comparison with Bézier extraction and DS T-splines for Example 4.1.

	Our algorithm	DS T-splines	Our algorithm	Bézier extraction
Level	N_e	N_{ve}	N_c	N_c
1	240	6	36	3840
2	739	90	540	11824
3	1405	302	1812	22480



(a) level 1 (b) level 2 (c) level 3

Fig. 8. Three refined AS T-meshes for Example 4.1.



(a) level 1 (b) level 2 (c) level 3

Fig. 9. Three refined AS T-meshes for Example 4.2.

Table 2
Statistics of comparison with Bézier extraction and DS T-splines for Example 4.2.

	Our algorithm	DS T-splines	Our algorithm	Bézier extraction
Level	N_e	N_{ve}	N_c	N_c
1	54	12	72	864
2	121	27	162	1936
3	344	96	576	5504
4	806	163	978	12896

5. Conclusion

It is not a trivial work to extend de Boor algorithm on T-splines because of the existing of T-junctions in the underlying T-mesh. For AS T-splines, the vertical T-junctions and horizontal T-junctions can not be too close, simplifying the complexity of the T-mesh. Thus it is possible to explore an efficient evaluation algorithm similar to de Boor algorithm for AS T-splines. In this paper, we propose a de Boor-like evaluation algorithm for AS T-splines. There are mainly three steps, the non-vanishing control points are found and the case indicator is determined in the first step, then the control points are updated one column by one column or one row by one row depending on the case indicator, and finally the de Boor algorithm is applied in one direction then in the other direction. The control points are updated as a weighted average of itself and its neighbors. With the help of Theorem 3.1, the weights for update are easily computed and there are at most 6 control points needed to be updated. In summary, the proposed algorithm is an evaluation algorithm for bicubic AS T-splines based on calculating control points and is an improvement over Bézier extraction.

The proposed de Boor-like algorithm in this paper mainly focus on bicubic AS T-splines since bicubic AS T-splines are most commonly used in applications. The de Boor-like algorithm can of course be extended to AS T-splines of arbitrary degree, which means the non-vanishing control points can be updated by a similar way as the bicubic AS T-splines before applying de Boor algorithm. But the complexity of the preprocessing has not been explored in details. Is there a more efficient way of applying de Boor algorithm for AS T-splines of arbitrary degree? We leave this question as a future work.

Declaration of Competing Interest

None.

References

[1] T.W. Sederberg, J. Zheng, A. Bakenov, A. Nasri, T-splines and T-NURCCSs, ACM Trans. Graphics 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. Graphics 23 (3) (2004) 276–283.
[3] H. Ipson, T-spline merging, Brigham Young University, April 2005 Masters thesis.
[4] T.W. Sederberg, G.T. Finnigan, X. Li, H. Lin, H. Ipson, Watertight trimmed NURBS, ACM Trans. Graphics 27 (2008). Article no. 79
[5] M.A. Scott, X. Li, T.W. Sederberg, T.J.R. Hughes, Local refinement of analysis-suitable T-splines, Comput. Methods Appl. Mech.Eng. 213–216 (2012) 206–222.
[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.Eng. 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.Eng. 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 Geometric Des. 29 (2012) 63–76.
[10] Jingjing, Z. Li, On the linear independence and partition of unity of arbitrary degree analysis-suitable T-splines, Commun. Math. Stat. 3 (2015) 353–364.
[11] X. Li, M.A. Scott, Analysis-suitable T-splines: characterization, refinablility and approximation, Math. Models Methods Appl.Sci. 24 (2014).
[12] G.E. Farin, Curves and Surfaces for CAGD, A Practical Guide, Fifth Edition, Morgan Kaufmann Publishers, San Francisco, 1999.
[13] R.G. Yang Zhang, Visit Pataranutaporn, de Boor-suitable (DS) T-splines, Graph. Models (2017).
[14] 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.
[15] Y. Bazilevs, V.M. Calo, J.A. Cottrell, J.A. Evans, T.J.R. Hughes, S. Lipton, M.A.S. andT. W. Sederberg, Isogeometric analysis using T-splines, Comput. Methods Appl. Mech.Eng. 199 (2010) 229–263.
[16] L.B. Veiga, A. Buffa, D.C.G. Sangalli, Analysis-suitable T-splines are dual-compatible, Comput. Methods Appl. Mech. Eng 249–252 (2012) 42–51.
[17] X. Li, Some properties for analysis-suitable T-splines, J. Comput. Math. 33 (2015) 428–442.
[18] M.J. Borden, M.A. Scott, J.A. Evans, T.J. Hughes, Isogeometric finite element data structures based on Bézier extraction of NURBS, Int. J. Numer. MethodsEng. 87 (2011) 15–47.
[19] R. Borst, L. Chen, The role of Bézier extraction in adaptive isogeometric analysis: local refinement and hierarchical refinement, Int. J. Numer. MethodsEng. 113 (2017) 999–1019.