Journal of Computational Mathematics
Vol.xx, No.x, 2025, 1–14.

http://www.global-sci.org/jcm
doi:10.4208/jcm.2502-m2024-0179

GREEDY LOCAL REFINEMENT FOR ANALYSIS-SUITABLE T-SPLINES WITH LINEAR COMPLEXITY*

Liangwei Hong and Xin Li[1)]
School of Mathematical Science, USTC, Hefei 230026, China
Emails: bodhihlw@mail.ustc.edu.cn, lixustc@ustc.edu.cn

Abstract

Achieving linear complexity is crucial for demonstrating optimal convergence rates in adaptive refinement. It has been shown that the existing linear complexity local refinement algorithm for T-splines generally produces more degrees of freedom than the existing greedy refinement, which lacks linear complexity. This paper introduces a novel greedy local refinement algorithm for analysis-suitable T-splines, which achieves linear complexity and requires fewer control points than existing algorithms with linear complexity. Our approach is based on the observation that confining refinements around each T-junction to a pre-established feasible region ensures the algorithm's linear complexity. Building on this constraint, we propose a greedy optimization local refinement algorithm that upholds linear complexity while significantly reducing the degrees of freedom relative to previous linear complexity local refinement methods.

Mathematics subject classification: 65D07.
Key words: T-splines, Analysis-suitable T-splines, Linear complexity, Isogeometric analysis.

1. Introduction

Isogeometric analysis (IGA) is a potent numerical method for solving partial differential equations (PDEs) [6, 16, 26, 27, 29]. The basic idea of IGA uses the spline functions from computer aided design (CAD) for numerical simulation, eliminating the necessity for model conversion. In the early stages of IGA research, the most commonly used representation is non-uniform rational B-splines (NURBS). However, the tensor-product structure makes knot insertion having global impacts. In order to introduce adaptive refinement into NURBS representation, several relevant constructions are receiving particular attention, such as hierarchical B-splines [10, 14], T-splines [30, 32], PHT-splines [7, 19] and LR splines [8].

Among these technologies, hierarchical B-splines and T-splines are the two most used ones. Hierarchical B-splines [10] represent a classic local refinement method, ensuring the nested spaces and the linear independence of the basis functions [17]. Truncated basis for hierarchical splines (THB-splines) has been introduced [12, 13], which leads to more localized and unstructured meshes. Adaptive isogeometric methods with hierarchical splines [3–5] have linear complexity and optimal convergence rates.

T-splines [30, 32] are an important technology in industrial design [31] and isogeometric analysis [15, 23, 28, 29, 34]. Although the whole class of T-splines is not suitable as a basis for

* Received August 19, 2024 / Revised version received November 25, 2024 / Accepted February 17, 2025 / Published online April 8, 2025 /
[1)] Corresponding author

IGA because of possible linear dependence [2], analysis-suitable T-splines (AST-splines), are defined [20–22, 28, 33] to satisfy all the desirable properties for IGA.

The achievement of linear complexity plays a pivotal role in establishing the optimal convergence rates of adaptive refinement algorithms [9]. Firstly, Scott *et al.* [28] introduced a local refinement algorithm for AST-splines based on a greedy strategy. However, its theoretical proof of possessing linear complexity remains elusive. In contrast, the refinement algorithms developed by [11, 25] guarantee linear complexity and optimal convergence rates. Nevertheless, this approach leads to a substantial increase in the degrees of freedom associated with AST-splines.

Our objective is to synergize the advantages of two distinct algorithms, thereby reducing the degrees of freedom while maintaining linear complexity. The main idea is to select only some of the additional edges of the algorithm from [25] using an iterative greedy-algorithm based on the assignment of a weight to each feasible edge insertion. Notably, we transform the nested neighborhoods, originally discussed by [25], into distance constraints between edges. This modification facilitates direct input of complex initial meshes without the need for repeated iterative computations. By integrating these new distance constraints with the greedy strategy, we achieve a significant reduction in the degrees of freedom within the resulting AST-mesh. This enhancement enhances the practical applicability of our local refinement algorithm.

This paper is organized as follows. T-splines and AST-splines are reviewed in Section 2. Section 3 details the greedy local refinement algorithm. Section 4 demonstrates the superiority of the algorithm through numerical experiments. The last section presents the conclusion and future work.

2. T-splines

In this section, we review the basic concepts of T-meshes and T-splines [1].

2.1. Index T-meshes

Following the approach proposed by [1], we define T-splines based on T-meshes within the index domain, referred to as index T-meshes in this paper. A T-mesh $\mathcal{T}$ for a bi-degree (d_1, d_2) T-spline consists of faces, edges, and vertices that form a rectangular partition of the index domain $[0, c + d_1] \times [0, r + d_2]$. Each corner (or vertex) of the rectangles is located at integer coordinates. Let $p = \lfloor (d_1 + 1)/2 \rfloor$ and $q = \lfloor (d_2 + 1)/2 \rfloor$, which represent the maximal integers less than or equal to $(d_1 + 1)/2$ and $(d_2 + 1)/2$, respectively. Then the active region is denoted as the rectangular region $[p, c + d_1 - p] \times [q, r + d_2 - q]$. As elucidated below, the active region encompasses the anchors pertinent to the blending functions, while other indices are required for defining the blending function when the anchor is close to the boundary.

A T-mesh consists of three fundamental elements: vertices, edges, and faces. A vertex, representing the corner of a rectangle within the T-mesh, can be denoted as (σ_i, τ_i) or $\{\sigma_i\} \times \{\tau_i\}$. An edge, characterized as a line segment connecting two vertices without any intermediate vertices, is represented by $[\sigma_j, \sigma_k] \times \{\tau_i\}$ or $\{\sigma_i\} \times [\tau_j, \tau_k]$ for horizontal or vertical edges, respectively. A face, defined as a rectangle devoid of any internal edges or vertices, is represented by $[\sigma_i, \sigma_j] \times [\tau_i, \tau_j]$ or $(\sigma_i, \sigma_j) \times (\tau_i, \tau_j)$. The valence of a vertex is the number of edges that have the vertex as an endpoint. For interior vertices, only I-junctions, valence three (called T-junctions), or valence four vertices are allowed. The symbols $\vdash$, $\dashv$, $\perp$, and $\top$ denote the four potential orientations of T-junctions. A T-mesh $\mathcal{T}$ for a bi-degree (d_1, d_2) T-spline is admissible if for any vertex (i, j), if $0 \le i \le p$ or $c + d_1 - p \le i \le c + d_1$, the vertex cannot be $\perp, \top$ and if

$0 \le j \le q$ or $r + d_2 - q \le j \le r + d_2$, the vertex cannot be $\vdash$ and $\dashv$. In the following, we always assume the T-mesh is admissible.

An anchor in the index T-mesh corresponds to a blending function and is associated with vertices (both d_1 and d_2 are odd), edges (one of d_1 or d_2 is even), or faces (both d_1 and d_2 are even). For the i-th anchor $\mathbf{A}_i$, a local index vector $\vec{\sigma}_i \times \vec{\tau}_i$ is defined, which is instrumental in defining the blending function $\mathcal{T}_i(s,t)$. The values of $\vec{\sigma}_i = [\sigma_i^0, \ldots, \sigma_i^{d_1+1}]$ and $\vec{\tau}_i = [\tau_i^0, \ldots, \tau_i^{d_2+1}]$ are determined by projecting rays in both the s and t directions from the i-th anchor across the T-mesh. This process collects a total of $d_1 + 2$ and $d_2 + 2$ knot indices, respectively, to construct $\vec{\sigma}_i$ and $\vec{\tau}_i$ such that the anchor is in the middle of the $\vec{\sigma}_i \times \vec{\tau}_i$.

2.2. T-splines

Given two global knot vectors, $\vec{\mathbf{s}} = [s_0, s_1, \ldots, s_{c+d_1}]$ and $\vec{\mathbf{t}} = [t_0, t_1, \ldots, t_{r+d_2}]$, we define the blending function $\mathcal{T}_i(s,t)$ for the i-th anchor as $\mathcal{T}_i(s,t) = B[\vec{\mathbf{s}}_i](s)B[\vec{\mathbf{t}}_i](t)$, where $B[\vec{\mathbf{s}}_i](s)$ and $B[\vec{\mathbf{t}}_i](t)$ are B-spline functions of degrees d_1 and d_2, respectively. These functions are constructed using local knot vectors $\vec{\mathbf{s}}_i = [s_{\sigma_i^0}, s_{\sigma_i^1}, \ldots, s_{\sigma_i^{d_1+1}}]$ and $\vec{\mathbf{t}}_i = [t_{\tau_i^0}, t_{\tau_i^1}, \ldots, t_{\tau_i^{d_2+1}}]$, derived from the global knot vectors.

Here, σ_i^k and τ_i^k are indexing functions that map the global knot vectors to the local knot vectors. Specifically, σ_i^k selects the k-th knot from the global knot vector $\vec{\mathbf{s}}$ to form the local knot vector $\vec{\mathbf{s}}_i$ for the i-th blending function, and similarly, τ_i^k selects the k-th knot from the global knot vector $\vec{\mathbf{t}}$ to form the local knot vector $\vec{\mathbf{t}}_i$. These indexing functions ensure that each blending function $\mathcal{T}_i(s,t)$ is constructed from the appropriate segment of the global knot vectors.

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

$$\mathbf{T}(s,t) = \sum_{i=1}^{n_A} \mathbf{C}_i \mathcal{T}_i(s,t), \tag{2.1}$$

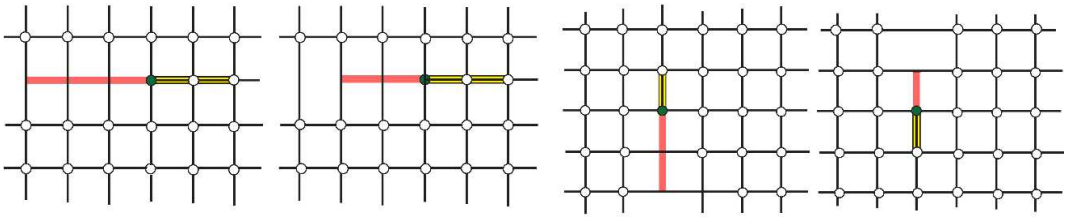
where $\mathbf{C}_i = (\omega_i x_i, \omega_i y_i, \omega_i z_i, \omega_i)$ are homogeneous control points, $\omega_i \in \mathbb{R}$ are weights, $\mathcal{T}_i(s,t)$ are blending functions, and $n_A(\mathcal{T})$ is the number of control points or anchors.

2.3. Analysis-suitable T-splines

Analysis-suitable T-splines are a subclass of T-splines with the restrictions based on T-junction extensions [18]. For a T-junction $\mathbf{T}_i : (\sigma_i, \tau_i)$ of type $\vdash$, the extension $ext^f_{a_i}(\mathbf{T}_i)$ indicates the line segments $[\underline{\beta}, \sigma_i) \times \{\tau_i\}$ such that it has a_i intersections with the T-mesh. And the edge extension $ext^e_{b_i}(\mathbf{T}_i)$ indicates the line segments $[\sigma_i, \overline{\beta}] \times \{\tau_i\}$ such that it has b_i intersections with the T-mesh. The face and edge extensions for the other kinds of T-junctions $\dashv$, $\perp$, $\top$ can be defined similarly.

For the T-junction $\mathbf{T}_i : (\sigma_i, \tau_i)$ of type $\vdash$ or $\dashv$, we define the face extension $ext^f(\mathbf{T}_i)$ as $ext^f_p(\mathbf{T}_i)$ and the edge extension $ext^e(\mathbf{T}_i)$ as $ext^e_p(\mathbf{T}_i)$. For the T-junction $\mathbf{T}_i : (\sigma_i, \tau_i)$ of type $\perp$ or $\top$, we define the face extension $ext^f(\mathbf{T}_i)$ as $ext^f_q(\mathbf{T}_i)$ and the edge extension $ext^e(\mathbf{T}_i)$ as $ext^e_q(\mathbf{T}_i)$. And the extension $ext(\mathbf{T}_i) = ext^f(\mathbf{T}_i) \bigcup ext^e(\mathbf{T}_i)$. Fig. 2.1 shows several different face and edge extensions for different degrees.

Definition 2.1. *A bi-degree (d_1, d_2) T-mesh is called an AS T-mesh if and only if the extensions of any T-junctions of different directions have no intersections. An AS T-spline is a T-spline defined on an AS T-mesh.*



(a) bi-degree (5,*) (b) bi-degree (4,*) (c) bi-degree (*,3) (d) bi-degree(*,2)

Fig. 2.1. The face (red line segments) and edge (yellow line segments) extensions for four different kinds of T-junctions.

3. A New Local Refinement Algorithm with Linear Complexity

In this section, we present a new local refinement algorithm with linear complexity. Initially, we define the D distance, taking into account the levels and directions of each edge. We then introduce the concept of D_c distance between edges, which plays a critical role in establishing primary constraints. These constraints are instrumental in determining the feasibility of refining the T-mesh. We demonstrate that the new constraint conditions are congruent with the nested constraint conditions previously outlined in [11,25]. This alignment confirms that our algorithm retains linear complexity.

3.1. Linear complexity

This section provides a concise review of the concepts related to the linear complexity of local refinement algorithms, utilizing the notations from [11, 25].

Given two T-meshes $\mathcal{T}_1$ and $\mathcal{T}_2$, we say $\mathcal{T}_1 \leq \mathcal{T}_2$ if $\mathcal{T}_2$ can be obtained by inserting edges into $\mathcal{T}_1$. Denote

$$n_A(\mathcal{T}_2/\mathcal{T}_1) = |n_A(\mathcal{T}_2) - n_A(\mathcal{T}_1)|, \quad \text{if} \quad \mathcal{T}_1 \leq \mathcal{T}_2.$$

For a given admissible T-mesh $\mathcal{T}_j$, consider $\mathcal{M}_j$ as the set of marked refinement faces. The T-mesh resulting from the refinements applied to $\mathcal{M}_j$ is denoted as $\text{ref}^{\,d_1,d_2}(\mathcal{T}_j, \mathcal{M}_j)$, while the T-mesh resulting from executing any local refinement algorithm, such as those proposed by [25] or [28], is denoted as $\mathcal{T}_{j+1}$. It logically follows that

$$\mathcal{T}_j \leq \text{ref}^{\,d_1,d_2}(\mathcal{T}_j, \mathcal{M}_j) \leq \mathcal{T}_{j+1}.$$

Definition 3.1. *Let $\{\mathcal{T}_j\}_{j=0}^{J}$ denote the sequence of admissible meshes generated by a local refinement algorithm, where $\mathcal{M}_j$ is the set of input edges at the j-th refinement stage. The refinement algorithm is said to have linear complexity if and only if*

$$n_A(\mathcal{T}_J/\mathcal{T}_0) \leq C_{d_1,d_2} \sum_{j=0}^{J-1} |\mathcal{M}_j|, \tag{3.1}$$

where $|\cdot|$ denotes set cardinality, and C_{d_1,d_2} is a constant depending only on the degrees d_1 and d_2.

3.2. Primary constraints for the extensions

This section introduces the concept of a neighborhood constraint region and explores the definitions pertinent to this constraint. Subsequently, we demonstrate that this constraint is

synonymous with the linear complexity local refinement algorithm for T-splines as presented in [25]. Through the use of defined terms and methodical proof, we establish a direct correspondence between the proposed neighborhood constraint and the proven efficiency of the referenced local refinement strategy, underscored by its application to T-spline meshes.

For computational convenience, we assume that for each vertex v_i in the index T-mesh, its corresponding coordinates σ_i and τ_i can be expressed as finite-bit binary fractions. The direction of each edge e is denoted as $\mathrm{dir}(e) \in \{0, 1\}$. Without loss of generality, we set $\mathrm{dir}(e_H) = 0$ for horizontal edges e_H and $\mathrm{dir}(e_V) = 1$ for vertical edges e_V. Additionally, on e_H, the ordinate τ remains constant for any point, and we define $l(e_H)$ as the number of binary digits in τ. Similarly, on e_V, the abscissa σ remains constant for any point, and we define $l(e_V)$ as the number of binary digits in σ.

Remark 3.1. The assumption that the coordinates σ_i and τ_i of each vertex v_i are represented as finite-bit binary fractions is specific to the index T-mesh. In the actual T-mesh, vertices can be positioned arbitrarily without any impact. We introduce this definition solely to simplify the analysis of T-mesh topology. By discretizing the levels in this manner, we enable a more efficient computational approach while preserving the essential geometric properties of the mesh.

The key property of the refinement algorithm is that edge insertion is allowed only if the edge lies within the feasible region of some input edge. This feasible region, referred to as the (d_1, d_2)-region, is defined based on the level and direction of the input edge, as well as the polynomial bi-degree (d_1, d_2) of the T-spline blending functions.

Definition 3.2 (Vector-valued Distance). *Given $x \in \mathbb{R}^2$ and an edge e, we define their distance as the componentwise minimum absolute difference between x and each point of e,*

$$\mathrm{Dist}(e, x) := \min_{x_0 \in e} \mathrm{abs}(x_0 - x) \in \mathbb{R}^2. \tag{3.2}$$

Definition 3.3. *Given an edge e with level l_e and direction dir_e, and polynomial degrees d_1 and d_2, the (d_1, d_2)-region is defined as*

$$\mathcal{R}(e) := \big\{x \in \mathbb{R}^2 \mid \mathrm{Dist}(e, x) \le D(l_e, \mathrm{dir}_e)\big\}, \tag{3.3}$$

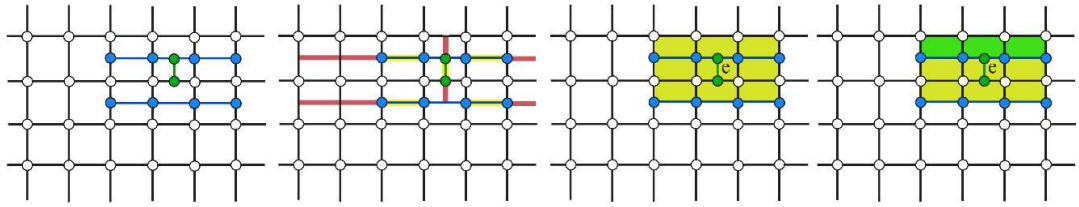
where $D(l, \mathrm{dir})$ and $\delta_D(l, \mathrm{dir})$ quantify the feasible and redundant distances, respectively, for a given level l and direction dir*,*

$$D(l, \mathrm{dir}) = \begin{cases} 2^{-l} \cdot (2p, 2d_2 - 2q + 1), & \text{if} \quad \mathrm{dir} = 0, \\ 2^{-l} \cdot (2d_1 - 2p + 1, q), & \text{if} \quad \mathrm{dir} = 1, \end{cases} \tag{3.4}$$

$$\delta_D(l, \mathrm{dir}) = \begin{cases} 2^{-l} \cdot (2, 0), & \text{if} \quad \mathrm{dir} = 0, \\ 2^{-l} \cdot (0, 1), & \text{if} \quad \mathrm{dir} = 1. \end{cases} \tag{3.5}$$

The definition of $D(l, \mathrm{dir})$ is consistent with the (p, q)-patch defined in Definition 2.4 of [25], with the determination point changed from the edge midpoint to the endpoint. Unlike the original paper, which could only compute elements obtained by bisection, the new definition can determine the neighborhood of any edge with a positive integer level.

Fig. 3.1 illustrates an example of a bi-degree (3,2) T-mesh, with face extensions and edge extensions indicated by red and yellow lines, respectively. In Fig. 3.1(c), the feasible region $\mathcal{R}(e)$ of the green edge e is highlighted in yellow. Blue edges are feasible edges within $\mathcal{R}(e)$. Fig. 3.1(d)



(a) bi-degree (3,2) AS T-mesh (b) T-extention of Fig. (a) (c) D neighborhood of e (d) δ_D neoghbirhood of e

Fig. 3.1. The D (yellow area) and δ_D (green area) neighborhood in bi-degree (3,2) AS T-mesh.

introduces δ_D as the redundant distance within the designated feasible region $\mathcal{R}(e)$. The green area represents the redundant region δ_D, defined as the excess part of the feasible region $\mathcal{R}(e)$ where the feasible edges introduced by edge e will not appear.

When computing the feasible region for an edge e, the redundant region induced by δ_D may introduce interference. In the following, we provide a definition for the feasible region of e across different combinations of levels and directions, and demonstrate that the accumulated redundant distance δ_D does not compromise the accuracy in determining the feasible edges.

Definition 3.4. *For any non-negative integers $l_1, \mathrm{dir}_1, l_2, \mathrm{dir}_2$ satisfying $l_1 \geq l_2, l_1 + \mathrm{dir}_1 \geq l_2 + \mathrm{dir}_2$, the cumulative of the distance function D between them, denoted as $D_c(l_1, \mathrm{dir}_1, l_2, \mathrm{dir}_2)$, is defined as*

$$D_c(l_1, \mathrm{dir}_1, l_2, \mathrm{dir}_2) = \begin{cases} \sum_{i=0}^{l_1-l_2-1} 2^i \cdot D(l_1, 0) + \sum_{i=1}^{l_1-l_2-\mathrm{dir}_2} 2^i \cdot D(l_1, 1), & \text{if} \quad \mathrm{dir}_1 = 0, \\ \sum_{i=0}^{l_1-l_2-1} 2^i \cdot D(l_1, 0), + \sum_{i=0}^{l_1-l_2-\mathrm{dir}_2} 2^i \cdot D(l_1, 1), & \text{if} \quad \mathrm{dir}_1 = 1 \end{cases}$$

$$= (2^{l_1-l_2} - 1) \cdot D(l_1, 0) + (2^{l_1+\mathrm{dir}_1-l_2-\mathrm{dir}_2} - 1) \cdot 2^{1-\mathrm{dir}_1} \cdot D(l_1, 1). \quad (3.6)$$

For an arbitrary input edge e with level l_e and direction dir_e, the feasible region $\mathcal{R}(e, l, \mathrm{dir})$ across different combinations of levels l and directions dir satisfying $l_e \geq l$ and $l_e + \mathrm{dir}_e \geq l + \mathrm{dir}$ is defined by

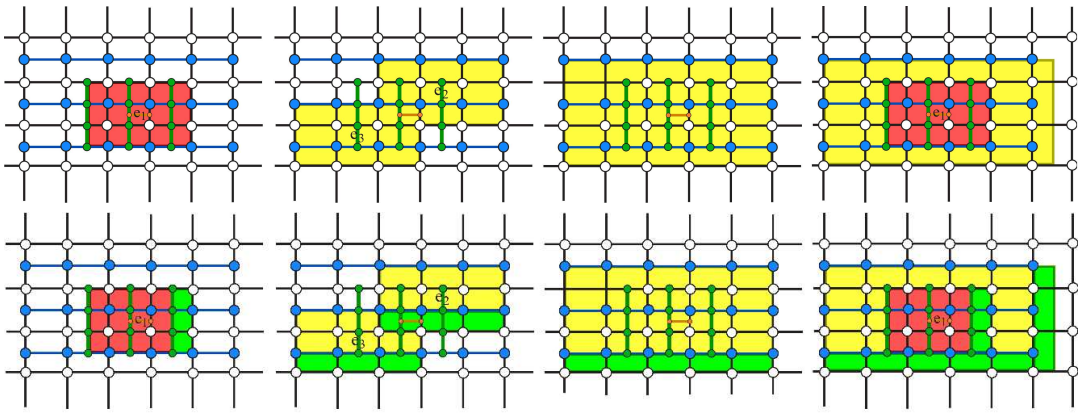
$$\mathcal{R}(e, l, \mathrm{dir}) := \{x \in \mathbb{R}^2 \mid \mathrm{Dist}(e, x) \leq D_c(l_e, \mathrm{dir}_e, l, \mathrm{dir})\}. \quad (3.7)$$

Theorem 3.1. *For any edge e_0 in a T-mesh $\mathcal{T}$, an edge e is considered a feasible edge of e_0 if and only if there exists a sequence of edges $\{e_1, e_2, \ldots, e_i\}$ such that for each $j \in \{1, 2, \ldots, i\}$, e_j is a feasible edge for e_{j-1}, and e is a feasible edge for e_i.*

Proof. As observed in Eq. (3.5) and Fig. 3.2, the redundant distance δ_D in each direction is halved as the level increases. Therefore, starting from the edge e_0 with level l_0 and direction dir_0, the sum of redundant distances to the edge e satisfies

$$\begin{aligned} \delta_{D_c}(l_0, \mathrm{dir}_0, l, \mathrm{dir}) &= (2^{l_0-l} - 1) \cdot \delta_D(l_0, 0) + (2^{l_0+\mathrm{dir}_0-l-\mathrm{dir}} - 1) \cdot 2^{1-\mathrm{dir}_0} \cdot \delta_D(l_0, 1) \\ &= \left((2^{l_0-l} - 1) \cdot 2^{1-l_0}, (2^{l_0+\mathrm{dir}_0-l-\mathrm{dir}} - 1) \cdot 2^{1-l_0-\mathrm{dir}_0}\right) \\ &< (2^{1-l}, 2^{1-l-\mathrm{dir}}). \end{aligned} \quad (3.8)$$

Notably, for any edge with level l and direction dir, the minimum non-zero distance between its endpoints is $(2^{1-l}, 2^{1-l-\mathrm{dir}})$, and the above proof shows that δ_{D_c} is strictly smaller than



(a) D neighborhood of e_1 (b) D neighborhood of e_2 (c) Union of D neighborhoods of the green edges (d) D_c neighborhood of e_1

Fig. 3.2. In the bi-degree (3,2) T-mesh, (a) The red region shows the D neighborhood of edge e_1, (b) The yellow region represents the D neighborhood of edges e_2, e_3, (c) Illustrates the union of the D neighborhoods of all feasible edges (green edges) in the D neighborhood of edge e_1, (d) Displays the D_c neighborhood of edge e_1. The green regions represent the redundant region within each neighborhood region.

this minimum non-zero distance. This demonstrates that the feasible edges defined by the feasible region $\mathcal{R}(e, l, \text{dir})$ are not affected by the cumulative redundant distance δ_{D_c} introduced in D_c. Therefore, our feasible region concept is equivalent to the nested neighborhood definition proposed in paper [25]. □

For simplification, we denote the union of all input edges as $\mathcal{M}$. The corresponding feasible region is represented as $R(\mathcal{M}, l, \text{dir})$. For a T-mesh $\mathcal{T}$ where each edge has a integer level, we define its feasible region for any level l and direction dir as

$$R(\mathcal{T}, l, \text{dir}) = \bigcup_{e \in \mathcal{T}} R(e, l, \text{dir}). \tag{3.9}$$

Theorem 3.1 proves the transitivity of edge feasibility. Therefore, inserting feasible edges into T-mesh $\mathcal{T}$ does not alter its feasible region. This is because for any $e \in \mathcal{T}$, any feasible edges of a feasible edge of e are also feasible edges of e. Therefore, only input edges can modify the feasible region of T-mesh $\mathcal{T}$

$$R(\mathcal{T}^{\text{ref}}, l, \text{dir}) = R(\mathcal{T}, l, \text{dir}), \tag{3.10}$$

where $\mathcal{T}^{\text{ref}}$ is the T-mesh obtained by inserting feasible edges into $\mathcal{T}$.

During the iterative refinement process, such as adaptive mesh refinement, $\mathcal{M}_j$ denotes the set of input edges inserted into the j-th layer T-mesh $\mathcal{T}_j$, which yields the T-mesh $\mathcal{T}_{j+1}$. The corresponding feasible region satisfies

$$R(\mathcal{T}_{i+1}, l, \text{dir}) = R(\mathcal{T}_i, l, \text{dir}) \cup R(\mathcal{M}_i, l, \text{dir}). \tag{3.11}$$

By inserting all feasible edges into T-mesh $\mathcal{T}$, we obtain the T-mesh output by [25], which has been proven to be an AS T-mesh. Therefore, we can always find a set of feasible edges that transforms any given T-mesh into an AST-mesh. This finding is crucial as it guarantees the existence of a solution within our defined constraints, thereby ensuring the robustness and reliability of our method.

3.3. Local refinement algorithm

Given a T-mesh $\mathcal{T}_j$ with input edges $\mathcal{M}_j$, the resulting T-mesh, denoted as $\text{ref}^{d_1,d_2}(\mathcal{T}_j, \mathcal{M}_j)$, does not qualify as an AST mesh. Therefore, it is imperative to develop a linear complexity algorithm for local refinement that can transform the T-mesh $\text{ref}^{d_1,d_2}(\mathcal{T}_j, \mathcal{M}_j)$ into an AST mesh $\mathcal{T}_{j+1}$.

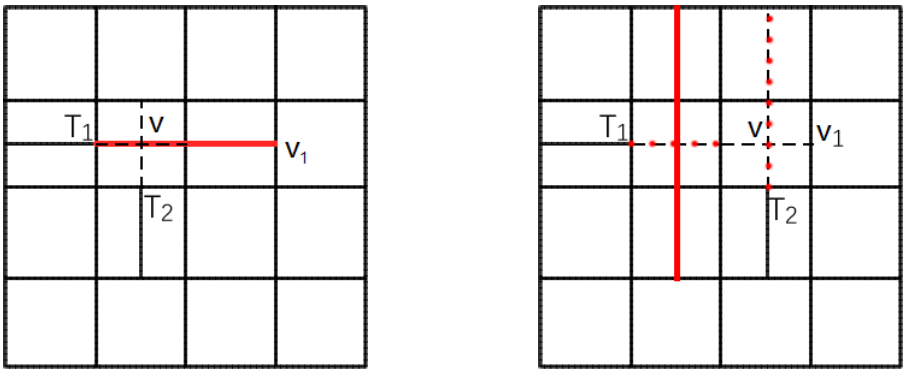
The algorithm described in [25] refines T-meshes by iteratively expanding marked and neighboring faces until no further unmarked faces meet the neighbourhood conditions. Despite maintaining certain mesh properties, the reliance on face-based neighbourhood definitions necessitates starting from a topology with specific characteristics. This complicates the derivation of conditions for greedy algorithms, potentially limiting efficiency in large-scale applications. To address these limitations, we utilize the edge-based feasible refinement region introduced in the previous section, simplifying the identification of refinement constraints in our algorithm. On the other hand, the greedy strategy in [28] only focuses on extending T-junctions, which does not encompass all refinement strategies needed in [25]. Therefore, we consider two primary strategies for refining the mesh, both aiming to maintain the analysis-suitable property.

Extending-method: This strategy extends edges from a T-junction. Fig. 3.3(a) illustrates the new edges of the T-junction $\mathcal{T}_1$ using red lines.

Truncating-method: This strategy involves the addition of new edges that intersect a T-extension perpendicularly, positioned between two corresponding T-junctions. This is demonstrated by the refinement with red lines in Fig. 3.3(b).

Definition 3.5. *Given a T-mesh $\mathcal{T}$, we define a graph $G(\mathcal{T})$, where each vertex in $G(\mathcal{T})$ corresponds to a T-junction in $\mathcal{T}$. An edge is formed between two vertices in $G(\mathcal{T})$ if the extensions of their respective T-junctions in $\mathcal{T}$ intersect. The set of all edges in $G(\mathcal{T})$ is denoted by $E(G(\mathcal{T}))$.*

For each edge in the graph $G(\mathcal{T})$, we evaluate the potential refinement strategies (extending or truncating). The efficiency of each strategy is quantified by the weight that reflects the trade-off between the number of edges in $E(G(\mathcal{T}))$ and the complexity introduced by additional control points.



(a) Extending-method (b) Truncating-method

Fig. 3.3. Types of edge insertion to convert a T-mesh into an AST mesh.

Specifically, the weight $w(e)$ for an edge $e \in E(G(\mathcal{T}))$ is calculated as

$$w(e) = \frac{\Delta E\big(G(\mathcal{T})\big)}{\Delta n_A}, \tag{3.12}$$

where $\Delta E(G(\mathcal{T}))$ represents the reduction in the number of edges in $G(\mathcal{T})$ after applying the strategy, and Δn_A is the increase in the number of control points or anchors in the T-mesh $\mathcal{T}$.

By adopting this greedy approach, the strategy with the highest weight is selected at each step, even if the weight is negative. This ensures that the refinement process minimizes the total number of additional vertices while effectively transforming the mesh into an AST mesh.

The local refinement algorithm proceeds as follows.

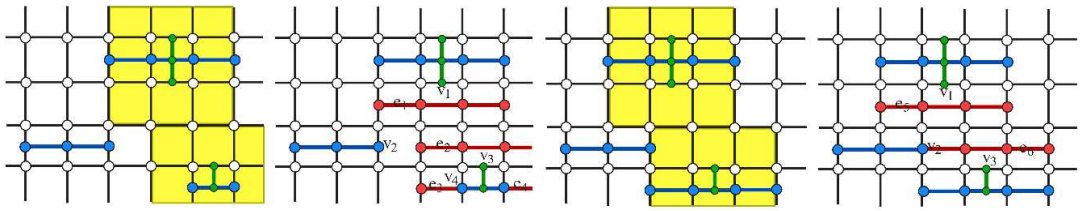
Algorithm 3.1: Refinement of a T-mesh to an AST-mesh.

Require: T-mesh $\mathcal{T}_i$ with input edges $\mathcal{M}_i$.
Ensure : Refined T-mesh $\mathcal{T}_{i+1}$.
1 Set $\mathcal{T}_{i+1}^{semi} = \mathrm{ref}^{d_1,d_2}(\mathcal{T}_j, \mathcal{M}_j)$, where the graph of $\mathcal{T}_{i+1}^{semi}$ is denoted as G.
2 Set the feasible refinement region of l_j and dir_j for $\mathcal{T}_{i+1}^{semi}$ to be $R(\mathcal{T}_{i+1}^{semi}, l_j, \mathrm{dir}_j) = R(\mathcal{T}_i, l_j, \mathrm{dir}_j) \cup R(\mathcal{M}_i, l_j, \mathrm{dir}_j)$.
3 **while** $E(G(\mathcal{T}_{i+1}^{semi}))$ is not empty **do**
4 For each edge $e \in E(G(\mathcal{T}_{i+1}^{semi}))$, calculate weights for all possible extending-method and truncating-method strategies.
5 Execute the refinement strategy corresponding to the highest weight using a greedy approach, subsequently updating $\mathcal{T}_{i+1}^{semi}$ and the graph G.
6 **end**

The following example illustrates the application of the local refinement algorithm on a bi-degree $(3, 3)$ T-mesh. In this context, the anchor points coincide with the vertices, so the weight of each refinement strategy is calculated based on the change in the number of vertices.

Figs. 3.4(a) and 3.4(c) depict the feasible regions $R(\mathcal{T}, 1)$ for a bi-degree (3,3) T-mesh $\mathcal{T}$, highlighted in yellow. In Fig. 3.4(b), we consider T-junctions v_1 and v_2 with intersecting T-extensions, where edge e_1 signifies the strategy for truncating the extension of T-junction v_1. Conversely, edges e_3 and e_4 represent the strategies for extending the T-junctions v_3, v_4 and v_3, v_5, respectively, while edge e_2 is unrelated to any T-extension intersections.

Fig. 3.4(d) presents a more complex scenario, where edges e_5 and e_6 are employed as refinement strategies for T-junctions v_1 and v_2, with e_5 serving as the extending-method and e_6 as the truncating-method. Additionally, e_6 is used as the extending-method for the T-junctions v_2, v_3.



(a) feasible region $R(\mathcal{T}, 1)$ (b) refinement strategies (c) feasible region $R(\mathcal{T}, 1)$ (d) refinement strategies

Fig. 3.4. In the bi-degree $(3, 3)$ T-mesh, (a)(c) The yellow region shows the union of the D neighborhoods. (b)(d) The red edges correspond to the feasible refinement strategies.

It has been determined that the insertion of edge e_5 results in the addition of four vertices while reducing the number of edge elements in the graph G by one. In contrast, e_6 contributes three vertices but decreases the edge count in $E(G(\mathcal{T}))$ by two. The efficiency of refinement strategies is quantified by the ratio of the reduction in $E(G(\mathcal{T}))$ to the increment in vertices. Employing a greedy approach, the most efficacious strategy is chosen at each juncture. For instance, in Fig. 3.4(d), edge e_6 is selected for insertion into the T-mesh $\mathcal{T}$.

Theorem 3.2. *The greedy local refinement algorithm (Algorithm* 3.1*) guarantees the existence of a solution and has linear complexity.*

Proof. In Section 3.1, we introduced the concept of linear complexity in the context of local refinement algorithms. Section 3.2 established that the novel algorithm operates as a subset of local refinements with linear complexity, as previously analyzed in [11, 25].

To clarify, the optimal admissible meshes $\mathcal{T}_0^{opt}, \mathcal{T}_1^{opt}, \ldots, \mathcal{T}_J^{opt}$ are generated using an existing linear complexity algorithm. This existing algorithm performs refinement by subdividing all face elements within the specified constraints. Our algorithm, on the other hand, has been shown to have constraints that are equivalent to those in the existing linear complexity algorithms. Within the same constraints, our algorithm employs a greedy strategy. In the worst-case scenario, if all possible strategies are executed, our algorithm would produce the same result as the existing linear complexity algorithm. Therefore, our algorithm always has a solution and maintains linear complexity.

Given the same set of input edges $\mathcal{M}_j$, the new algorithm produces a sequence of admissible meshes $\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_J$. Meanwhile, the existing algorithm generates a corresponding series of optimal admissible meshes $\mathcal{T}_0^{opt}, \mathcal{T}_1^{opt}, \ldots, \mathcal{T}_J^{opt}$, which satisfy the following inequalities:

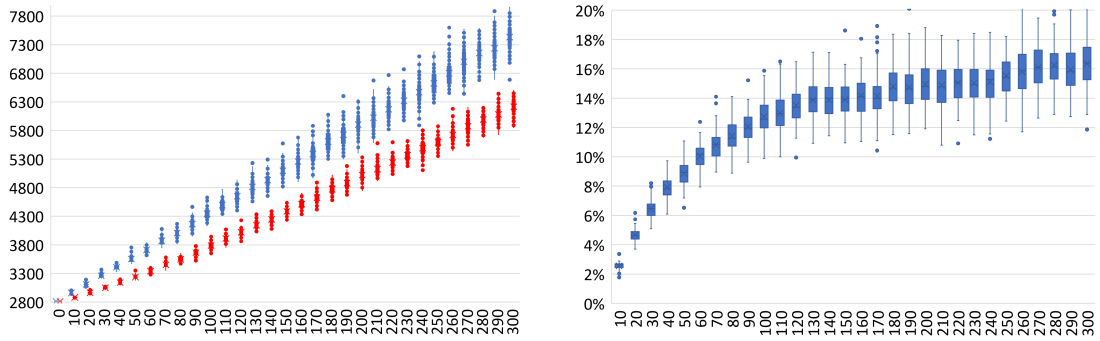
$$\mathcal{T}_j \leq \mathcal{T}_j^{opt}, \quad n_A(\mathcal{T}_J/\mathcal{T}_0) \leq n_A\big(\mathcal{T}_J^{opt}/\mathcal{T}_0\big) \leq C_{d_1,d_2} \sum_{j=0}^{J-1} |\mathcal{M}_j|. \tag{3.13}$$

Thus, the new algorithm preserves linear complexity. □

4. Numerical Experiment

The first experiment employs a random insertion strategy on a 50×50 tensor-product grid. Here, we randomly select n faces for $n = 10, 20, \ldots, 300$. For each n, faces are chosen randomly 100 times, and during each iteration, the selected faces are refined into four sub-faces. Let x_n represent the number of cross insertions, and y_n the quantity of control points generated by executing the local refinement algorithms. We assess y_n across various methods, presenting the findings in Fig. 4.1. Specifically, the outcomes from the algorithms reported by [28], and our current method are depicted in blue and red, respectively. Fig. 4.1(a) illustrates the number of cross insertions x_n on the x-axis against the number of control points y_n on the y-axis, demonstrating the linear complexity of our algorithm. Fig. 4.1(b) presents the optimization rate of the new algorithm, defined as $(y_n^{\text{orig}} - y_n^{\text{new}})/y_n^{\text{orig}}$, where y_n^{orig} and y_n^{new} are the number of control points generated by the original and new algorithms, respectively.

The new algorithm consistently yields a lower number of control points compared to the method delineated by [25]. Specifically, the optimization rate stabilizes at approximately 16% as the number of cross insertions increases, indicating the efficiency of the proposed method in reducing control points.



(a) Control points comparison

(b) Optimization rate of new algorithm

Fig. 4.1. Comparison of control points between two different algorithms (blue for [25] and red for the present paper). The x-axis represents the number of cross insertions, while the y-axis in Fig. 4.1(a) shows the number of control points generated after applying each algorithm.

The second experiment is associated with AST-spline based isogeometric analysis for a second-order elliptic partial differential equation. Suppose $\Omega \in \mathbb{R}^2$ is a connected, bounded domain with Lipschitz-continuous boundary, and

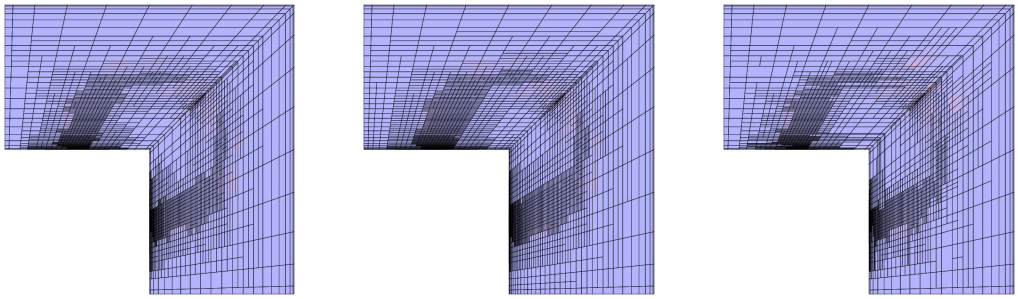
$$\begin{cases} -\dfrac{\partial^2 u}{\partial x^2} - \dfrac{\partial^2 u}{\partial y^2} = f, & (x,y) \in \Omega, \\ u = g & \text{on } \partial\Omega. \end{cases}$$

The solution approximation can be achieved using adaptive AST-spline based Isogeometric Analysis (IGA), which is solved via a linear system. Adaptive IGA typically consists of four steps: calculation, estimation, marking, and refinement. In each iteration, we compute the L^2 error estimator for each face element on the AS T-mesh $\mathcal{T}_j$, where the L^2 error is measured against the true solution. Subsequently, we mark the faces where the error estimators exceed a pre-defined threshold. Each marked face $\mathcal{M}_j$ is then refined into four sub-faces, generating a new T-mesh $\mathcal{T}_{j+1}$. It is important to note that due to differences in refinement algorithms, the estimation step could input edges in any levels and directions for refinement. Since the primary focus of this study is to compare the increase in control points across different algorithms, we define $\mathcal{M}_j$ as the set of faces marked for refinement by all three algorithms. The partial differential equation (PDE) is solved over two distinct domains, with the exact solution given by

$$u = 1 + \tanh\left(\frac{0.25 - \sqrt{(x-0.5)^2 + (y-0.5)^2}}{0.03}\right).$$

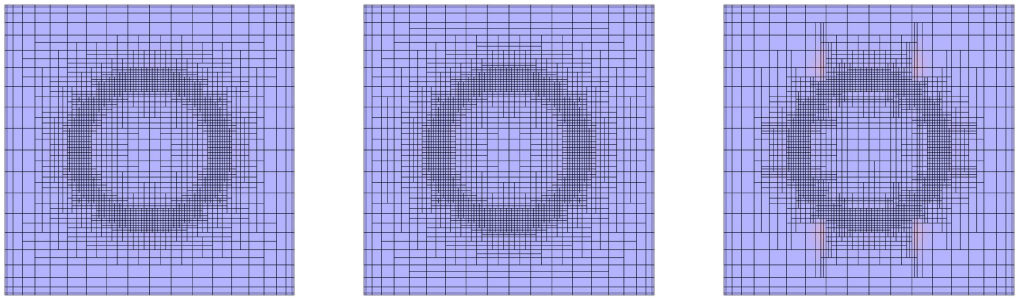
The functions f and g can be defined in terms of partial differential equation.

The first example considers an L-shaped region defined by $[0,1] \times [0,1] \setminus [0,0.5] \times [0,0.5]$, while the second domain is a standard rectangular region within $[0,1] \times [0,1]$. The refined meshes for both examples are depicted in Figs. 4.2 and 4.3. The computational results for the total number of control points after applying different numbers of refinement iterations are summarized in Table 4.1, which presents the total number of control points (DOFs) at different refinement iterations for two test cases. Notably, the new algorithm discussed in this paper yields a higher number of degrees of freedom (DOFs) than the algorithm introduced by [28] due to the increased constraints associated with maintaining linear complexity. Nonetheless, in



(a) Methods of present paper (b) Results of [25] (c) Results of [28]

Fig. 4.2. Comparison of control points propagation of three different algorithms on a L-shaped Region. Figs. (a), (b) and (c) are results produced by the present method, the method from [25] and [28] respectively.



(a) Methods of present paper (b) Results of [25] (c) Results of [28]

Fig. 4.3. Comparison of control points propagation of three different algorithms on a rectangular region. Figs. (a), (b) and (c) are results produced by the present method, the method from [25] and [28] respectively.

Table 4.1: Comparison of refinement algorithms on two T-meshes.

Refinement levels	Example 1			Example 2		
	[28]	[25]	Present	[28]	[25]	Present
0	529	529	529	529	529	529
1	967	995	979	877	895	877
2	2020	2271	2096	1694	1829	1739
3	3714	4144	3784	3294	3457	3301
4	4307	4812	4433	6697	7371	7005

both examples, our proposed algorithm significantly reduces the number of control points when compared to the linear complexity algorithm by [25]. In conclusion, our algorithm not only adheres to linear complexity but also substantially decreases the degrees of freedom required.

5. Conclusion

We have presented a greedy, adaptive local refinement algorithm for AST-splines that maintains linear complexity. The constraints on edges and their neighborhoods ensure the feasibility

of refinement strategies, confining the subdivision extensions related to each T-junction within a specific region. The new algorithm has been demonstrated to maintain linear complexity, consistent with the results in [11, 25]. Numerical results in Section 4 highlight the superiority of the new algorithm in terms of control point reduction and computational efficiency.

Previous studies, such as [11,25], have already extended these methods to three-dimensional cases, and [24] has shown that the approach retains linear complexity when applied to unstructured T-splines. Therefore, our algorithm can be generalized to three-dimensional and unstructured T-splines following a similar framework.

In the future, we plan to explore further extensions of our work. While our current algorithm primarily considers the scenario where each selected face is refined into four sub-faces, it is also capable of handling cases where the input edges in the index T-mesh have a finite level. However, addressing the case of dividing each face into m sub-faces poses a challenge, as a larger m could lead to a more localized local refinement algorithm. All these issues will be left as future work.

Acknowledgements. The authors were supported by the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDB0640000), by the Key Grant Project of the NSF of China (Grant No.12494552), by the NSF of China (Grant No.12471360).

References

[1] Y. Bazilevs, V.M. Calo, J.A. Cottrell, J.A. Evans, T.J.R. Hughes, S. Lipton, M.A. Scott, and T.W. Sederberg, Isogeometric analysis using T-splines, *Comput. Methods Appl. Mech. Engrg.*, **199**:5-8 (2010), 229–263.

[2] A. Buffa, D. Cho, and 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.

[3] A. Buffa and C. Giannelli, Adaptive isogeometric methods with hierarchical splines: Error estimator and convergence, *Math. Models Methods Appl. Sci.*, **26**:01 (2016), 1–25.

[4] A. Buffa and C. Giannelli, Adaptive isogeometric methods with hierarchical splines: Optimality and convergence rates, *Math. Models Methods Appl. Sci.*, **27**:14 (2017), 2781–2802.

[5] A. Buffa, C. Giannelli, P. Morgenstern, and D. Peterseim, Complexity of hierarchical refinement for a class of admissible mesh configurations, *Comput. Aided Geom. Design*, **47** (2016), 83–92.

[6] J.A. Cottrell, T.J. Hughes, and Y. Bazilevs, *Isogeometric Analysis: Toward Integration of CAD and FEA*, John Wiley & Sons, 2009.

[7] J. Deng, F. Chen, X. Li, C. Hu, W. Tong, Z. Yang, and Y. Feng, Polynomial splines over hierarchical T-meshes, *Graphical Models*, **74** (2008), 76–86.

[8] T. Dokken, T. Lyche, and K.F. Pettersen, Polynomial splines over locally refined box-partitions, *Comput. Aided Geom. Design*, **30** (2013), 331–356.

[9] S. Du, Q. Hou, and X. Xie, A linearized adaptive dynamic diffusion finite element method for convection-diffusion-reaction equations, *Ann. Appl. Math.*, **39**:3 (2023), 323–351.

[10] D.R. Forsey and R.H. Bartels, Hierarchical B-spline refinement, in: *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, Vol. 22, ACM, 205–212, 1988.

[11] G. Gantner and D. Praetorius, Adaptive IGAFEM with optimal convergence rates: T-splines, *Comput. Aided Geom. Design*, **81** (2020), 101906.

[12] C. Giannelli, B. Jüttler, S.K. Kleiss, A. Mantzaflaris, B. Simeon, and J. Špeh, THB-splines: An effective mathematical technology for adaptive refinement in geometric design and isogeometric analysis, *Comput. Methods Appl. Mech. Engrg.*, **299** (2016), 337–365.

[13] C. Giannelli, B. Jüttler, and H. Speleers, THB-splines: The truncated basis for hierarchical splines, *Comput. Aided Geom. Design*, **29**:7 (2012), 485–498.

[14] G. Greiner and K. Hormann, Interpolating and approximating scattered 3D-data with hierarchical tensor product B-splines, in: *Surface Fitting and Multiresolution Methods*, Vanderbilt University Press, 163–172, 1996.

[15] P. Hennig, M. Kästner, P. Morgenstern, and D. Peterseim, Adaptive mesh refinement strategies in isogeometric analysis: Computational comparison, *Comput. Methods Appl. Mech. Engrg.*, **316** (2017), 424–448.

[16] T.J. Hughes, J.A. Cottrell, and Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, *Comput. Methods Appl. Mech. Engrg.*, **194**:39-41 (2005), 4135–4195.

[17] R. Kraft, *Adaptive and Linearly Independent Multilevel B-splines*, Universität Stuttgart, Sonderforschungsbereich 404, 1997.

[18] X. Li, Some properties for analysis-suitable T-splines, *J. Comput. Math.*, **33**:4 (2015), 428–442.

[19] X. Li, J. Deng, and F. Chen, Surface modeling with polynomial splines over hierarchical T-meshes, *Vis. Comput.*, **23** (2007), 1027–1033.

[20] X. Li and L. Hong, ON T-spline classification, *J. Comput. Math.*, **40**:3 (2022), 472–483.

[21] X. Li and M.A. Scott, Analysis-suitable T-splines: Characterization, refineability, and approximation, *Math. Models Methods Appl. Sci.*, **24**:06 (2014), 1141–1164.

[22] X. Li, J. Zheng, T.W. Sederberg, T.J. Hughes, and M.A. Scott, On linear independence of T-spline blending functions, *Comput. Aided Geom. Design*, **29**:1 (2012), 63–76.

[23] L. Liu, Y. Zhang, T.J.R. Hughes, M.A. Scott, and T.W. Sederberg, Volumetric T-spline construction using Boolean operations, *Eng. Comput.*, **30** (2014), 425–439.

[24] R. Maier, P. Morgenstern, and T. Takacs, Adaptive refinement for unstructured T-splines with linear complexity, *Comput. Aided Geom. Design*, **96** (2022), 102117.

[25] P. Morgenstern and D. Peterseim, Analysis-suitable adaptive T-mesh refinement with linear complexity, *Comput. Aided Geom. Design*, **34** (2015), 50–66.

[26] D. Schillinger, L. Dede, M.A. Scott, J.A. Evans, M.J. Borden, E. Rank, and T.J. 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** (2012), 116–150.

[27] M.A. Scott, M.J. Borden, C.V. Verhoosel, T.W. Sederberg, and T.J. Hughes, Isogeometric finite element data structures based on Bézier extraction of T-splines, *Internat. J. Numer. Methods Engrg.*, **88**:2 (2011), 126–156.

[28] M.A. Scott, X. Li, T.W. Sederberg, and T.J. Hughes, Local refinement of analysis-suitable T-splines, *Comput. Methods Appl. Mech. Engrg.*, **213** (2012), 206–222.

[29] M.A. Scott, R.N. Simpson, J.A. Evans, S. Lipton, S.P. Bordas, T.J. Hughes, and T.W. Sederberg, Isogeometric boundary element analysis using unstructured T-splines, *Comput. Methods Appl. Mech. Engrg.*, **254** (2013), 197–221.

[30] T.W. Sederberg, D.L. Cardon, G.T. Finnigan, N.S. North, J. Zheng, and T. Lyche, T-spline simplification and local refinement, *ACM Trans. Graphics*, **23**:3 (2004), 276–283.

[31] T.W. Sederberg, G.T. Finnigan, X. Li, H. Lin, and H. Ipson, Watertight trimmed NURBS, *ACM Trans. Graphics*, **27**:3 (2008), 79.

[32] T.W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri, T-splines and T-NURCCs, *ACM Trans. Graphics*, **22**:3 (2003), 477–484.

[33] L.B. Veiga, A. Buffa, and D.C.G. Sangalli, Analysis-suitable T-splines are Dual-Compatible, *Comput. Methods Appl. Mech. Engrg.*, **249–252** (2012), 42–51.

[34] C.V. Verhoosel, M.A. Scott, R. de Borst, and T.J.R. Hughes, An isogeometric approach to cohesive zone modeling, *Internat. J. Numer. Methods Engrg.*, **87** (2011), 336–360.