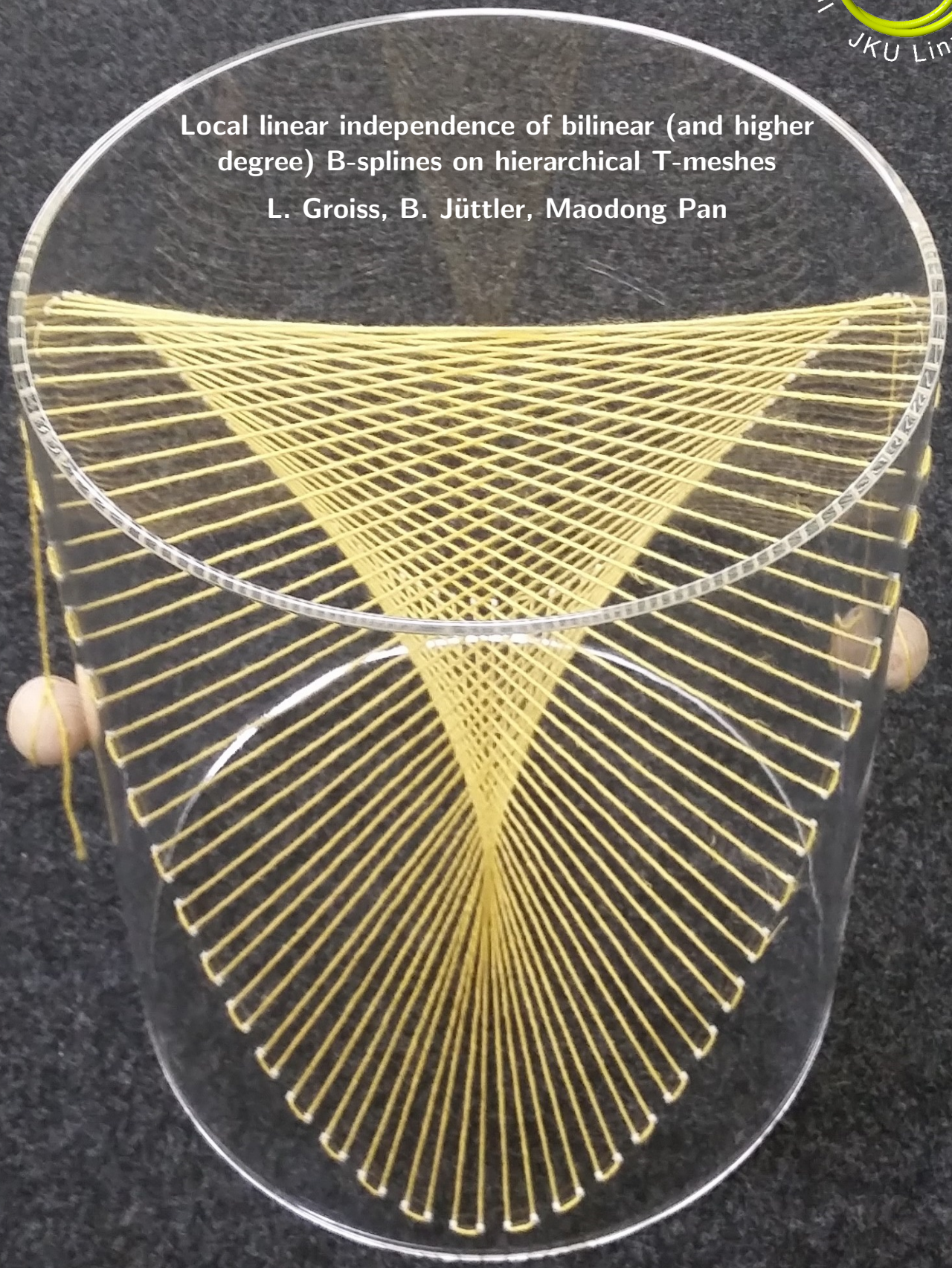
JKU

Institute of Applied Geometry
JKU Linz

Local linear independence of bilinear (and higher degree) B-splines on hierarchical T-meshes

L. Groiss, B. Jüttler, Maodong Pan



AG Report No. 97
March 30, 2023
www.ag.jku.at

Local linear independence of bilinear (and higher degree) B-splines on hierarchical T-meshes

Lisa Groiss[a], Bert Jüttler[a], Maodong Pan[b,*]

[a] *Institute of Applied Geometry, Johannes Kepler University, Linz/Austria*
[b] *Nanjing University of Aeronautics and Astronautics, China*

Abstract

We generate hierarchical T-meshes by repeatedly inserting new line segments, in order to adapt both the size and the shape of the cells to the specific requirements of the underlying application. The associated spaces of bilinear spline functions are spanned by the locally refined (LR) B-splines of Dokken et al. (2013), which are products of univariate B-splines defined over local knot vectors. Our method guarantees that these functions are linearly independent for the obtained class of hierarchical T-meshes. Furthermore, they even possess the property of local linear independence, since exactly 4 functions take non-zero values on each cell of the mesh, and they form a partition of unity without the need to perform additional scaling or truncation.

The correctness of our refinement algorithm is verified by enumerating the (newly introduced) local configurations, which represent the possible topologies of the mesh in the vicinity of a cell. It is then shown that the method works well in all the resulting situations. The fairly large number (385) of cases sheds some light on the challenges associated with linear independence in the context of LR B-spline refinement.

Additionally we apply our method to meshes that possess the additional property local semi-regularity (first studied by Weller and Hagen, 1995), where only 49 topologically different situations may arise. However, we observe that the resulting class of hierarchical T-meshes is less flexible, which leads to a substantial amount of excess refinement in applications.

Finally we note that the same construction can be applied to generate basis functions for C^s–smooth spline spaces of degree $p = 2s + 1$, again with the property of local linear independence.

Keywords: Adaptive refinement, polynomial splines over hierarchical T-meshes, locally refined B-splines, local linear independence, local semi-regularity

*Corresponding author

Email addresses: lisa.groiss@jku.at (Lisa Groiss), bert.juettler@jku.at (Bert Jüttler), mdpan@mail.ustc.edu.cn (Maodong Pan)

1. Introduction

Generalizations of tensor-product splines that support adaptive refinement form an important mathematical technology for Geometric Modeling and Isogeometric Analysis (IGA). Several options are available:

Hierarchical B-splines were established by Forsey and Bartels (1988), who introduced the concepts of local refinement and multi-resolution editing, thereby breaking the tensor-product nature of traditional NURBS. The problem of their linear independence was addressed in the PhD thesis of Kraft (1998). He provided a specific selection mechanism that creates a system of (linearly independent) basis functions. While linear independence is not essential in some geometric design applications, it is a basic requirement in IGA. More than ten years later, Giannelli et al. (2012) introduced the truncation mechanism that modifies Kraft's basis to obtain a partition of unity. These splines found applications in geometric design and IGA (Kiss et al., 2014; Zheng and Chen, 2022).

T-splines were invented by Sederberg et al. (2003) in order to address the limitation of NURBS regarding T-junctions in the control meshes for geometric modeling. The linear independence of the T-spline blending functions was discussed by Li et al. (2012) and Scott et al. (2012), who noted that this property can be guaranteed only for the restricted subset of "analysis-suitable" (AS) T-splines. T-splines were used for geometric modeling and numerical simulation (Bazilevs et al., 2010; Wang et al., 2013). The truncation mechanism was extended to T-splines by Wei et al. (2017).

Polynomial splines over hierarchical T-meshes (PHT-splines) were introduced by Deng et al. (2008) in the bicubic case as spaces of functions with C^1 smoothness. For meshes created by cross insertion, the spaces are equipped with a non-negative basis that forms a partition of unity, which is constructed by locally modifying tensor-product B-splines with the help of their Bézier representations. The potential of PHT-splines for geometric modeling and analysis has been demonstrated in a number of publications (Li et al., 2007; Wang et al., 2011; Li et al., 2016). Several variants of PHT-splines were studied in order to avoid the so-called decay phenomenon and to increase the flexibility (Kang et al., 2015; Zhu and Chen, 2017; Ni et al., 2019).

Locally refined B-splines (LR B-splines), which were invented by Dokken et al. (2013), provide another approach to the local refinement of spline spaces. Some properties of these splines were established by Bressan (2013), in particular the non-nested support (N_2S) condition that characterizes local linear independence. Based on these results, Bressan and Jüttler (2015) proposed a refinement strategy that mimics the construction of hierarchical splines. Furthermore, Patrizi et al. (2020) proposed the non-nested support & structured refinement (N_2S_2) strategy, which uses local tensorization with respect to one direction – similar to the notion of local semi-regularity studied by Weller and Hagen (1995) – in order to eliminate pairs of basis functions with nested supports. Patrizi and Dokken (2020) explored necessary conditions for guaranteeing the linear independence of LR B-splines. Various applications to geometric design and simulations and generalizations to non-polynomial spline spaces have also been presented in the literature (Johannessen et al., 2014; Skytt et al., 2015; Bracco et al., 2016).

In the present paper, we discuss a simple algorithm that generates hierarchical T-meshes by repeatedly inserting new line segments, in order to adapt freely the size and the shape of the cells to the specific requirements of the underlying application. Compared to the existing constructions, we gain additional flexibility for the geometry of the mesh.

More precisely, we adopt the framework of LR B-splines to construct the basis functions for PHT-splines, i.e., for the C^s-smooth bivariate tensor-product spline spaces of degree $p = 2s + 1$ on hierarchical T-meshes. Our method guarantees that the generated functions possess the properties of local linear independence and partition of unity. Furthermore, we apply the proposed refinement algorithm to the more restricted meshes with the additional property of local semi-regularity. The correctness of the new algorithm is verified both theoretically and experimentally.

The paper is organized as follows. Section 2 establishes the basic concepts and notations needed to describe the new refinement algorithm. In Section 3, we introduce the (standardized) local configurations for analyzing the properties of the corresponding spline space. The refinement algorithm is presented in Section 4 and its correctness is verified theoretically in Section 5. Several numerical results are presented in Section 6 to validate the effectiveness of the proposed algorithm. Finally, we summarize the paper and identify directions for future work.

2. Preliminaries

This section establishes the notation needed to describe the refinement algorithm. Furthermore, we discuss basic properties of the resulting spline basis.

2.1. B-splines and the sets of local knots

We construct spaces of spline functions with domain $\Omega = (a, b)^2$, where we assume that the boundaries satisfy $a < b$. The spaces are spanned by sets of bilinear B-splines

$$\beta_{(X,Y)}(u, v) = N^1_X(u)N^1_Y(v) \tag{1}$$

that are defined by two local knot vectors $X = (x, x', x'')$ and $Y = (y, y', y'')$. For visualization we associate the *anchor point* (x', y') with each B-spline. In order to build the generating system

$$B^{(\ell)} = \{\beta_{(X,Y)} \ : \ (X, Y) \in K^{(\ell)}\} \tag{2}$$

that spans the spline space, we employ pairs (X, Y) that vary in the finite *sets of local knots* (more precisely, of pairs of knot vectors)

$$K^{(\ell)} \subset \{((x, x', x''), (y, y', y'')) : x < x' < x'', y < y' < y''\} \ .$$

These sets will be created by an iterative refinement procedure, and the upper index $\ell = 0, 1, 2, \ldots$, which is called the *level*, indicates the number of steps. The iteration is initialized by choosing

$$\begin{aligned} K^{(0)} = \{&((a-1, a, b), (a-1, a, b)), ((a-1, a, b), (a, b, b+1)), \\ &((a, b, b+1), (a-1, a, b)), ((a, b, b+1), (a, b, b+1))\} \ . \end{aligned} \tag{3}$$

This set of local knots corresponds to the four bilinear tensor-product Bernstein polynomials $B^{(0)}$ (which are obtained as restrictions of bilinear B-splines) on Ω. Note that we use additional "phantom" knots at $a-1$ and $b+1$.

In each step of the iteration we generate a new set $K^{(\ell+1)}$ by splitting all the knot vector pairs in $K^{(\ell)}$, using *refinement operators* that are based on *sets of splits*. These sets and the operators are described in the next two sections:

2.2. The sets of splits

We employ vertical and horizontal splits, which are simply line segments

$$\begin{aligned} \mathbb{V} &= \big\{\{x\} \times [y, y'] : x, y, y' \in [a-1, b+1], y < y'\big\} \quad \text{and} \\ \mathbb{H} &= \big\{[x, x'] \times \{y\} : x, x', y \in [a-1, b+1], x < x'\big\} \ , \end{aligned} \tag{4}$$

respectively. The end points of each split (line segment)

$$v = \{x\} \times [y, y'] \in \mathbb{V} \quad \text{or} \quad h = [x, x'] \times \{y\} \in \mathbb{H}$$

are denoted as

$$\partial v = \{x\} \times \{y, y'\} \quad \text{or} \quad \partial h = \{x, x'\} \times \{y\} \ , \tag{5}$$

respectively.

We consider a sequence $(S^{(\ell)})_{\ell=0,1,\dots}$ of *sets of splits*. Each instance is the union of a finite number of vertical and horizontal splits, and the sequence is initialized by the set

$$S^{(0)} = \{a-1, a, b, b+1\} \times [a-1, b+1] \ \cup \ [a-1, b+1] \times \{a-1, a, b, b+1\} \tag{6}$$

that covers the knot line segments of the four bilinear B-splines in $B^{(0)}$.

In each step of the iteration, we generate the next set of splits iteratively by adding a *single* split (line segment) s, which is called the *new split*, to the previous set of splits. The new split is either horizontal or vertical and it is required to connect two points from the previous level,

$$\exists s \in \mathbb{V} \cup \mathbb{H} : \ S^{(\ell+1)} = s \cup S^{(\ell)} \ \wedge \ \partial s \subset S^{(\ell)} \ . \tag{7}$$

The sets of splits $S^{(\ell)}$ define the cells

$$C^{(\ell)} = \mathrm{CC}(\Omega \setminus S^{(\ell)}), \tag{8}$$

where $\mathrm{CC}(S)$ is the set of connected components of a set S. The closures of the cells $C^{(\ell)}$ form a *hierarchical T-mesh* that covers the closure $\overline{\Omega}$ of the domain.

2.3. The fixed-point refinement operator

For a given sequence of sets of splits $S^{(\ell)}$, we generate the associated sets of local knots $K^{(\ell)}$ with the help of the *fixed-point refinement operator*, which is defined as follows:

Firstly, each vertical (or horizontal) line segment v or h defines an associated refinement operator R_v (R_h) that acts on the knot vector pairs according to

$$R_v(X,Y) = \begin{cases} \{((x,\hat{x},x'),(y,y',y'')),((\hat{x},x',x''),(y,y',y''))\} & \text{if } \hat{x} \in (x,x') \wedge [y,y''] \subseteq [\hat{y},\hat{y}'] \ , \\ \{((x,x',\hat{x}),(y,y',y'')),((x',\hat{x},x''),(y,y',y''))\} & \text{if } \hat{x} \in (x',x'') \wedge [y,y''] \subseteq [\hat{y},\hat{y}'] \ , \\ \{((x,x',x''),(y,y',y''))\} & \text{otherwise} \ , \end{cases}$$

where $v = \{\hat{x}\} \times [\hat{y},\hat{y}']$, $X = (x,x',x'')$ and $Y = (y,y',y'')$ (and analogously for a horizontal line segment). See Fig. 1 for a visualization. Note that we do not allow multiple knots.

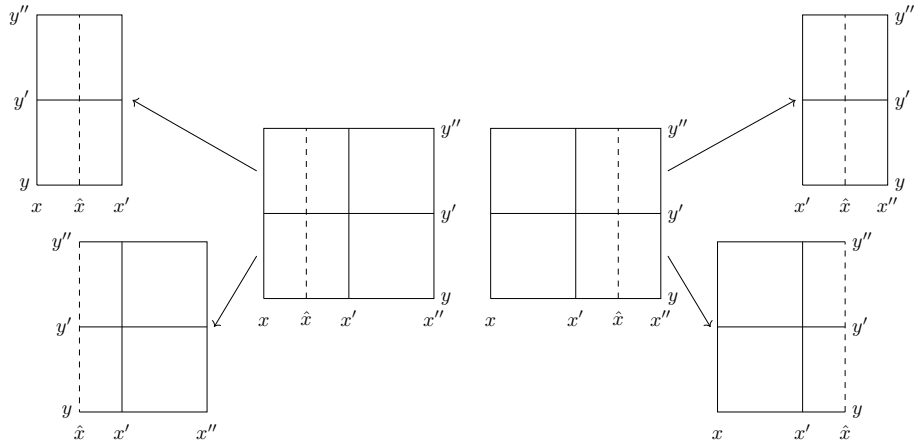


Figure 1: Refining a bilinear B-spline by applying a vertical split $v = \hat{x} \times [\hat{y},\hat{y}']$ with $[y,y''] \subseteq [\hat{y},\hat{y}']$ and $\hat{x} \in (x,x')$ (left) or $\hat{x} \in (x',x'')$ (right).

Secondly, these refinement operators are extended to the sets of local knot vectors $K^{(\ell)}$,

$$R_v K^{(\ell)} = \bigcup_{(X,Y)\in K^{(\ell)}} R_v(X,Y) \qquad \text{and} \qquad R_h K^{(\ell)} = \bigcup_{(X,Y)\in K^{(\ell)}} R_h(X,Y) \ .$$

Furthermore we generalize them to sets of splits via

$$R_{S^{(\ell+1)}} K^{(\ell)} = \bigcup_{\substack{v\in\mathbb{V} \\ v\subset S^{(\ell+1)}}} R_v K^{(\ell)} \cup \bigcup_{\substack{h\in\mathbb{H} \\ h\subset S^{(\ell+1)}}} R_h K^{(\ell)} \ .$$

Note that we consider all the vertical or horizontal line segments in $\bigcup S^{(\ell+1)}$.

Finally we apply the splits repeatedly until the local knots are not modified anymore,

$$R^{\star}_{S^{(\ell+1)}} K^{(\ell)} = R^{n}_{S^{(\ell+1)}} K^{(\ell)} \quad \text{ such that } n \in \mathbb{N} \text{ fulfills } \quad R^{n+1}_{S^{(\ell+1)}} K^{(\ell)} = R^{n}_{S^{(\ell+1)}} K^{(\ell)} \ ,$$

where

$$R^n_{S^{(\ell+1)}} K^{(\ell)} = \underbrace{R_{S^{(\ell+1)}}(R_{S^{(\ell+1)}}(\ldots R_{S^{(\ell+1)}}(}_{n \text{ times}} K^{(\ell)})\ldots)) \ .$$

We shall refer to $R^\star_{S^{(\ell+1)}}$ as the *fixed-point refinement operator*.

Summing up, the local knots of level $\ell+1$ are generated by applying the fixed-point refinement operator with respect to the splits of level $\ell+1$ to the local knots of the previous level,

$$K^{(\ell+1)} = R^\star_{S^{(\ell+1)}} K^{(\ell)} \ .$$

As shown by Dokken et al. (2013), the associated generating system – and hence also the space spanned by it – is independent of the splits and their ordering. In other words, each set $K^{(\ell)}$ of local knot vectors is fully determined by the set $S^{(\ell)}$.

2.4. TABs: T-meshes with Associated Bilinear B-splines

Summing up, for each level ℓ we get a triplet

$$\mu^{(\ell)} = (\Omega, S^{(\ell)}, K^{(\ell)}) \quad \ell = 0, 1, \ldots \tag{9}$$

that defines a *T-mesh with Associated Bilinear B-splines* (TAB) on the domain Ω. We chose the Greek character μ to represent specific instances of these triplets since we essentially refer to a *mesh*.

In the next sections we will also consider more general TABs, independently of an iterative refinement procedure. They take the form

$$\mu = (\Delta, S, K) \tag{10}$$

with an arbitrary domain Δ that is either equal to Ω or to a subset thereof (e.g., a single cell). As before, the sets S and K consist of splits and local knots, respectively. The associated bilinear B-splines then form the set

$$B = \{\beta_{(X,Y)} \ : \ (X,Y) \in K\}$$

and the cells $C = \mathrm{CC}(\Delta \setminus S)$ are again the connected components of $\Delta \setminus S$.

3. The local and the standardized local configurations

In order to study the properties of the spline space, we analyze the active B-splines on a single cell $c \in C$.

3.1. The local configurations

We consider the relevant basis functions and their knot vectors,

$$B_c = \{\beta_{(X,Y)} \in B \ : \ \beta_{(X,Y)}|_c \neq 0\} \quad \text{and} \quad K_c = \{(X,Y) \in K \ : \ \beta_{(X,Y)}|_c \neq 0\} \ ,$$

and we collect the knots[1] in two vectors

$$X_c = \bigcup_{(X,Y)\in K_c} X \qquad \text{and} \qquad Y_c = \bigcup_{(X,Y)\in K_c} Y \ .$$

The support extension of the cell

$$\hat{c} = \bigcup_{\beta_{(X,Y)}\in B_c} \operatorname{supp} \beta_{(X,Y)} \tag{11}$$

is the closed set formed by the union of the supports of all the B-splines that are non-zero on it. The geometry of the mesh in the vicinity of a cell is described by the *set of local splits*

$$S_c = \bigcup_{\substack{x\in X_c;\, y,y'\in Y_c \\ \{x\}\times[y,y']\subset S\cap\hat{c}}} \{x\}\times[y,y'] \quad \cup \quad \bigcup_{\substack{x,x'\in X_c;\, y\in Y_c \\ [x,x']\times\{y\}\subset S\cap\hat{c}}} [x,x']\times\{y\}$$

that are relevant for the cell. The *local configuration λ of a cell $c \in C$* is the triplet

$$\lambda = (c, S_c, K_c) \ . \tag{12}$$

The local configurations characterize two properties of the TAB:

- The system B_c of B-splines for a cell c possesses the property of *local linear independence (LLI)* if $|B_c| = 4$ is satisfied by all cells c. Clearly, local linear independence implies the linear independence of B.[2] LLI also implies that the B-splines form a partition of unity without scaling (Bressan and Jüttler, 2015, Theorem 4).

- The local configuration is *vertically semi-regular* (Weller and Hagen, 1995) if $|X_c| = 4$ and the knot vector pairs K_c satisfy

  $$X \in \{(x, x', x''), (x', x'', x''')\} \quad \forall (X, Y) \in K_c \ , \tag{13}$$

  where $X_c = (x, x', x'', x''')$. *Horizontally semi-regular* cells are defined analogously. We say that a TAB has the property of *local semi-regularity* (LSR) if all the local configurations are vertically or horizontally semi-regular. LLI is implied by LSR.

3.2. Standardized local configurations

Even if the knot numbers are bounded, the local configurations vary in an infinite set. However, the behaviour of the refinement algorithm (which will be specified later) depends solely on the topology, which is captured by the *standardized local configuration*.

We use the two standardization functions

$$\xi_c : X_c \mapsto 2\mathbb{Z} + 1 \quad \text{and} \quad \eta_c : Y_c \mapsto 2\mathbb{Z} + 1 \ ,$$

which are uniquely determined by requiring that

[1] Applying the set union operator to the local knot vectors X and Y involves a slight abuse of notation since these vectors need to be considered as sets. Moreover, we represent the results again as *vectors*, with elements sorted in ascending order.

[2] LLI is an essential requirement for the construction of hierarchical spline bases (Kraft, 1998), and it simplifies the derivation of spline projectors (Giust et al., 2020).

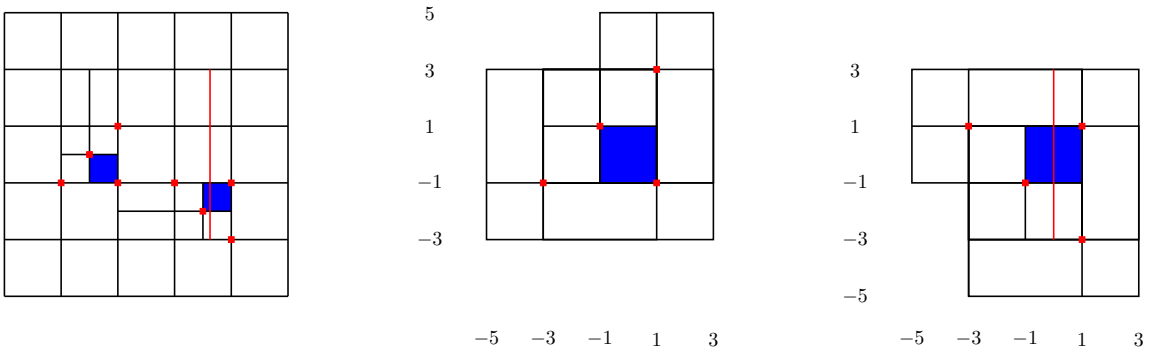


Figure 2: Left: All the active basis functions on the blue cells are represented by their anchor points (red squares). A new split is depicted in red. Middle and right: The associated standardized configurations are shown for the left cell (middle) and for the right cell (right). The standardized split can be seen for the second configuration.

- they are strictly increasing,
- values at adjacent knots differ by 2 and
- the knots bounding the x- and the y-range of c are transformed into -1 and 1.

Applying them to the local configuration gives the *standardized set of local knots*

$$\tilde{K}_c = \{(\xi_c(X), \eta_c(Y)) \ : \ (X,Y) \in K_c\}$$

and the *standardized sets of local splits*

$$\tilde{S}_c = \bigcup_{\{x\}\times[y,y']\subset S_c;\, y<y'} \{\xi_c(x)\} \times [\eta_c(y), \eta_c(y')] \quad \cup \bigcup_{[x,x']\times\{y\}\subset S_c;\, x<x'} [\xi_c(x), \xi_c(x')] \times \{\eta_c(y)\} \ .$$

The cell c is transformed into $(-1,1)^2$, see Fig. 2. This procedure transforms any local configuration (12) into an associated *standardized local configuration*

$$\sigma = (\,(-1,1)^2, \tilde{S}_c, \tilde{K}_c)$$

that possesses the same characteristics (number of basis functions, local linear independence, semi-regularity) as the original configuration.

For further reference, we establish the algorithm

$$\textsc{Extract\&Standardize}(\text{TAB } \mu) \tag{14}$$

that transforms any TAB (see Eq. (10)) into the associated set

$$\{(\,(-1,1)^2, \tilde{S}_c, \tilde{K}_c) \mid c \in \mathrm{CC}(\Delta \setminus S)\}$$

of standardized local configurations.

Algorithm: INSERT&EXTEND(new split s, TAB μ))

```
1  global LSR;
2  finished ← false;
3  while not finished do
4  |  S ← s ∪ S;
5  |  K ← R*_S K;
6  |  B ← {β_(X,Y) : (X,Y) ∈ K};                          /* see also Eq. (2) */
7  |  C ← CC(Δ \ S);                                      /* see also Eq. (8) */
8  |  finished ← true;
9  |  for c ∈ C do
10 |  |  if |B_c| ≠ 4 or (LSR = "on" and (c, S_c, K_c) is not semi-regular) then
11 |  |  |  s ← s⁺ ∩ ĉ;
12 |  |  |  if s ⊄ S then
13 |  |  |  |  finished ← false;
14 |  |  |  |  break;                                    /* exit the for loop */
15 return (Δ, S, K);
```

4. The split insertion algorithm

Our refinement procedure, which is described by Algorithm INSERT&EXTEND (see page 9), inserts a new split s into a TAB $\mu = (\Delta, S, K)$. It proceeds as follows:

- The input consists of a new split s and a TAB μ.
  - The TAB is a triplet that contains the domain Δ, the set S of already existing splits and the set K of local knots, cf. (10).
  - The new split s is required to be *feasible*, in the sense that it splits at least one basis function and contains no parts that are unnecessary. More precisely, a split $s \in \mathbb{V} \cup \mathbb{H}$ is said to be *elementary feasible*, if

    $$\exists (X,Y) \in K : R_s(X,Y) \neq \{(X,Y)\} \ \wedge \ s \subset \operatorname{supp} \beta_{(X,Y)} \ .$$

    A *feasible* new split is an element of $\mathbb{V} \cup \mathbb{H}$, which is a union of elementary feasible splits and splits that are already present in S.

  In addition, the globally defined flag LSR specifies which local configurations are accepted by the algorithm. While local linear independence is always required, local semi-regularity is additionally enforced if this variable is set to "on". We say that a TAB is *acceptable* if the local configurations of all cells are acceptable (i.e., they do not pass the test in line 10).

- The algorithm proceeds by adding the new feasible split s to S and simultaneously extending it – while keeping the feasibility – until all configurations are acceptable or if the remaining violations (i.e., non-acceptable local configurations) cannot be resolved by extending it further.

- First, we apply the refinement operator to the set of local knots. Second, we generate the associated basis functions and we identify the cells. This enables us to check the acceptability of the local configurations for all the cells (see line 10).
- In order to deal with non-acceptable local configurations, we perform extensions of the new split (see line 11). These are defined with the help of the support extension $\hat{c}$, see (11), and the vertical or horizontal line

$$s^+ = \begin{cases} \{x\} \times \mathbb{R} & \text{if } s = \{x\} \times [y, y'] \in \mathbb{V} \\ \mathbb{R} \times \{y\} & \text{if } s = [x, x'] \times \{y\} \in \mathbb{H} \end{cases} \tag{15}$$

  that contains the initial new split s. The feasibility of the new split is preserved by this extension. We keep only the information about the extended part of the new split since the remaining part has already been added to the set of splits S at this stage.
- The control variable "finished" takes the value false (line 13) only if the remaining violations lead to a further extension of the new split s. Thus, the algorithm terminates if either all local configurations are acceptable or if the split extensions triggered by the remaining violations are already present in S.

5. Generating acceptable meshes

We state the main result:

Theorem. *For each sequence $(s^{(\ell)})_{\ell=1,2,\dots}$ of feasible splits, the algorithm* Insert&Extend *generates acceptable T-meshes with associated bilinear B-splines (TABs)*

$$\mu^{(\ell+1)} = \textsc{Insert\&Extend}(s^{(\ell+1)}, \mu^{(\ell)}) \ , \quad \ell = 0, 1, 2, \dots \ , \tag{16}$$

from the tensor-product mesh $\mu^{(0)} = (\Omega, S^{(0)}, K^{(0)})$, cf. Eqns. (3) *and* (6)*. The B-splines span the full bilinear spline space associated with the T-mesh.*

More precisely, the algorithm enables refinement by inserting feasible new splits into TABs, while simultaneously maintaining the properties of local linear independence or local semi-regularity. The remainder of this section presents the proof of this fact.

5.1. Standardized new splits

First we prepare the proof by presenting a few auxiliary facts and notions. The standardized local configurations of a TAB created by Insert&Extend depend solely on the topological relation of the new split and the initial TAB. In order to capture this formally, we generalize the extraction and standardization procedure.

Firstly, we extend the standardization function ξ_c (and similarly η_c) by assigning even integers to the remaining arguments (that are not in X_c) while keeping the monotonicity. More precisely, the extended standardization function takes the values

$$\xi_c^+(x) = \begin{cases} \xi_c(x) & \text{if } x \in X_c \,, \\ \xi_c(\overline{x}) - 1 & \text{where } \overline{x} = \min\{\tilde{x} \in X_c \,:\, \tilde{x} > x\} \text{ if } x \notin X_c \text{ and } x < \max X_c \ , \\ \xi_c(\underline{x}) + 1 & \text{where } \underline{x} = \max\{\tilde{x} \in X_c \,:\, \tilde{x} < x\} \text{ if } x \notin X_c \text{ and } x > \min X_c \ . \end{cases}$$

Note that the second and the third branch give equivalent results if $\min X_c < x < \max X_c$.

Secondly, the standardized form of the vertical new split $s = \{x\} \times [y, y']$ is given by

$$\tilde{s}_c = \{\xi_c^+(x)\} \times [\eta_c(\min([y, y'] \cap Y_c), \eta_c(\max([y, y'] \cap Y_c)] \ .$$

Horizontal new splits are dealt with analogously. As an example, Fig. 2 includes a split (left, shown in red) and the associated standardized split (right, shown in red).

Finally, we use this procedure to establish the generalized version

$$\textsc{Extract\&Standardize}^+(\text{new split } s, \text{ TAB } \mu)$$

of the algorithm in (14). It transforms the input of the algorithm Insert&Extend (a split and a TAB) into the associated set

$$\{\Big(\tilde{s}_c, \big((-1, 1)^2, \tilde{S}_c, \tilde{K}_c\big)\Big) \mid c \in \mathrm{CC}(\Delta \setminus S)\}$$

that is formed by standardized local configurations and the associated standardized splits.

5.2. Compatibility of Extract&Standardize *and* Insert&Extend

A new split s is said to be *saturated* if the algorithm Insert&Extend accepts the resulting mesh without extending the split, i.e., if the condition in line 10 is never satisfied.

Lemma 1. *The compositions of the algorithms*

$$\textsc{Extract\&Standardize} \circ \textsc{Insert\&Extend} \circ \textsc{Extract\&Standardize}^+$$

and

$$\textsc{Extract\&Standardize} \circ \textsc{Insert\&Extend}$$

give identical results if applied to a saturated new split s of an acceptable TAB μ.

The example shown in Fig. 3 demonstrates the necessity of the assumption regarding saturated splits. If a non-saturated split s (solid red) is inserted into the mesh, then the standardized local configurations obtained from the blue cell are different for the two compositions of the algorithms. Indeed, a split extension (dashed red) is triggered by a violation of the local linear independence on another cell (green), hence it is not considered when performing the initial standardization of the split s and of the blue cell in the first composition of algorithms.

Proof of Lemma 1. We consider a single cell c in the TAB μ and show that the resulting standard configurations in the sets obtained by the two compositions of algorithms are identical. This is obvious if none of the active B-splines is split by s. Otherwise, the result follows by noting that the original split and its standardized version are topologically equivalent on the cell's local configuration, due to the definition of the standardization procedure. Clearly, this carries over to the resulting standardized configuration. Moreover, the second composition of algorithms uses the same split on c as the first one, as the split is not extended by Insert&Extend, due to the assumption that s is saturated. □

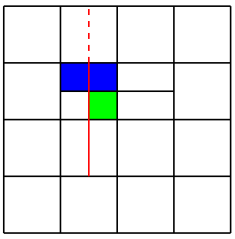


Figure 3: Saturated splits are a necessary assumption in Lemma 1. After inserting a split (solid red), the violation of LLI on the green cell triggers an extension (dashed red), leading to different standardized configurations of the blue cells.

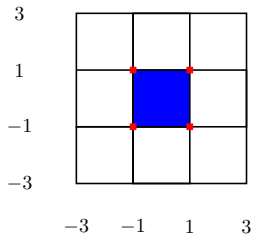


Figure 4: The standardized tensor-product configuration for the cell shown in blue. The red dots are the anchor points of the four B-splines.

5.3. INSERT&EXTEND *generates a finite number of standardized local configurations*

Now we consider the set Σ of standardized local configurations. Its elements are TABs $\sigma = ((-1,1)^2, S, K)$. The algorithm INSERT&EXTEND followed by EXTRACT&STANDARDIZE establishes a relation between them: The TAB σ splits into σ', represented as

$$\sigma \rightsquigarrow \sigma' \ ,$$

if there exists a new split s such that

$$\sigma' \in \textsc{Extract\&Standardize}\big(\textsc{Insert\&Extend}(s, \sigma)\big) \ .$$

The new split s can be assumed to be a standardized one, according to Lemma 1. We form the transitive closure $\rightsquigarrow^+$ of this relation,

$$\sigma \rightsquigarrow^+ \sigma' \iff \exists m \in \mathbb{Z}_+ \, \exists \sigma_1, \ldots, \sigma_m : \sigma = \sigma_1 \rightsquigarrow \sigma_2 \rightsquigarrow \ldots \rightsquigarrow \sigma_m = \sigma' \ ,$$

and use it to introduce the *set of descendants* of a local configuration

$$\langle \sigma \rangle = \{\sigma' \, : \, \sigma \rightsquigarrow^+ \sigma'\} \ .$$

Lemma 2. *The set of descendants $\langle \tau \rangle$ of the tensor-product configuration τ is finite. Moreover, all the descendants $((-1,1)^2, S, K)$ have the property of local linear independence. Additionally, they are all semi-regular if* LSR *is* "on".

Proof. The relation $\rightsquigarrow$ defines a directed graph, where the nodes are the standardized local configurations. We enumerate all the descendants of τ by the following procedure:

1. The set $\mathcal{N}$ of non-visited nodes is initialized by $\{\tau\}$. We also create the set $\mathcal{D}$ of descendants, which is initialized by the empty set.
2. We pick an element σ of $\mathcal{N}$ and remove it from the set.
3. All the possible standardized new splits of σ are generated (there are only finitely many of them).

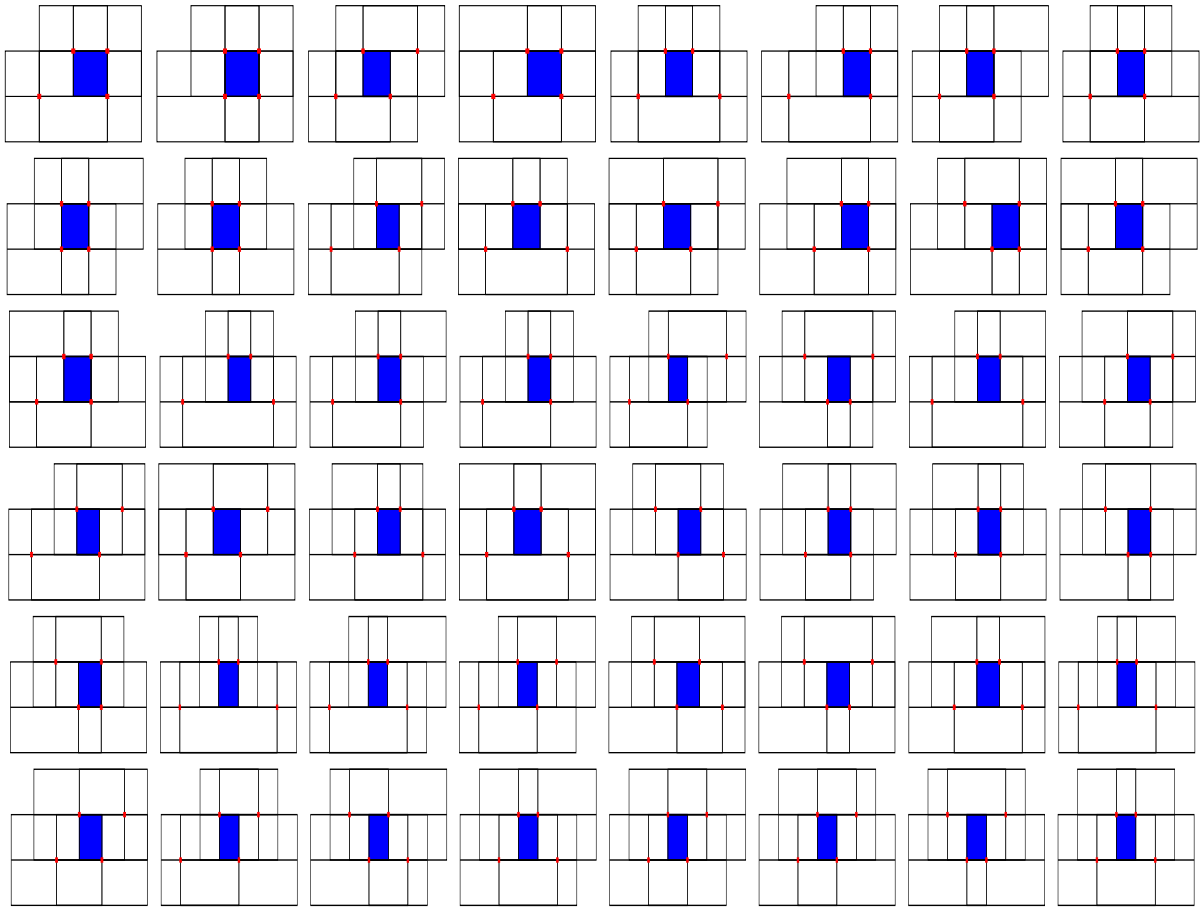


Figure 5: The 48 semi-regular standardized local configurations, which are different from the tensor-product configuration, see Fig. 4. The anchor points of the B-splines are shown as red squares.

4. We call EXTRACT&STANDARDIZE(INSERT&EXTEND(s, σ)) for all the new splits s that were found in the previous step. The resulting standardized local configurations are added to the set of descendants $\mathcal{D}$. In addition, all the newly found descendants (the resulting standardized local configurations that were not yet in $\mathcal{D}$) are added to $\mathcal{N}$.
5. Continue with the second step unless $\mathcal{N}$ is empty.

We apply this procedure for both values of the global variable LSR. It terminates in both cases.

For LSR="on", we find 49 standardized local configurations, which are shown in Fig. 4 and Fig. 5. The generation of three of the local configurations is depicted in Fig. 6. All 49 configurations are semi-regular, and therefore also locally linearly independent. Note that two configurations are considered as equivalent if a rigid displacement (rotation or reflection) transforms them into each other. Fig. 5 shows only one representative of each equivalence class.

For LSR="off", we find even 385 standardized local configurations. All of them possess

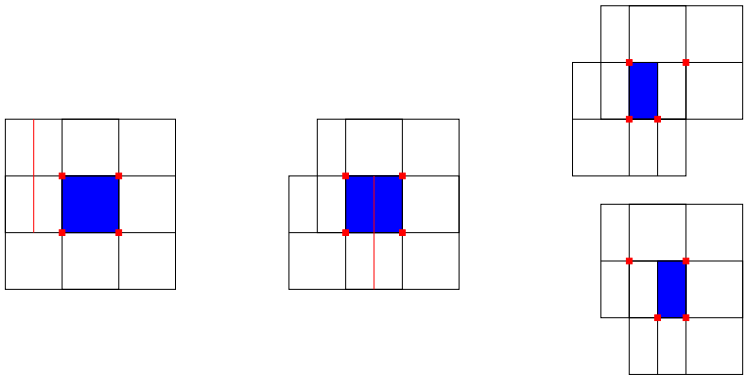


Figure 6: Applying the red split to the tensor-product configuration τ (left) produces one descendant (center). Applying another split (again shown in red) results in two descendants (right). The associated standardized local configuration are included in Fig. 5 (second, seventh and first configuration in line 1).

the property of local linear independence. Fig. 7 depicts eight selected instances, and the full list is available in the appendix. [3] □

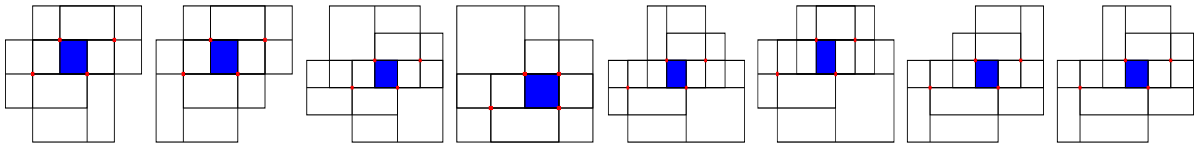


Figure 7: Selected standardized local configurations that possess the property of local linear independence but are not semi-regular.

We are now ready to prove the main result.

5.4. Proof of the Theorem

We consider a sequence of TABs $\mu^{(\ell)} = (\Omega, S^{(\ell)}, K^{(\ell)})$, $\ell = 0, \ldots, n$, which is generated iteratively by inserting new splits $s^{(1)}, s^{(2)}, \ldots, s^{(n)}$, i.e.,

$$\mu^{(\ell+1)} = \textsc{Insert\&Extend}(s^{(\ell+1)}, \mu^{(\ell)}) \ . \tag{17}$$

For each TAB $\mu^{(\ell)}$ there is the associated set

$$\Sigma^{(\ell)} = \{((-1,1)^2, \tilde{S}_c^{(\ell)}, \tilde{K}_c^{(\ell)}) \ : \ c \in CC(\Omega \setminus S^{(\ell)})\} = \textsc{Extract\&Standardize}(\mu^{(\ell)}) \tag{18}$$

of standardized local configurations. Without loss of generality, we assume that all the new splits are saturated. Indeed, calling the refinement algorithm with a non-saturated split gives the same result as calling it directly with the saturated split $s^{(\ell+1)}$ that has been found

[3] A closer inspection reveals that the anchor points are located on the two horizontal lines $y = \pm 1$ or on the two vertical lines $x = \pm 1$ in all the 385 cases, where each line carries two of them.

by the algorithm. More precisely, the split $s^{(\ell+1)}$ represents the differences between the set of splits S before and after the application of the procedure. This split is not always unique since it may or may not contain mesh line segments that were already present. The initial TAB $\mu^{(0)} = (\Omega, S^{(0)}, K^{(0)})$ is a tensor-product TAB, hence the associated set of standardized local configurations is equal to $\Sigma^{(0)} = \{\tau\}$.

We prove the theorem by contradiction and assume that all the TABs are acceptable, except for the final one $\mu^{(n)} = (\Omega, S^{(n)}, K^{(n)})$. More precisely, there exists a cell $\hat{c} \in CC(\Omega \setminus S^{(n)})$ with a local configuration

$$\sigma^\star = ((-1,1)^2, \tilde{S}^{(n)}_{\hat{c}}, \tilde{K}^{(n)}_{\hat{c}}) \in \Sigma^{(n)}$$

that is not acceptable.

First we use Lemma 1 to establish the relation

$$\forall \sigma' \in \Sigma^{(\ell+1)} \, \exists \sigma \in \Sigma^{(\ell)} \, : \, \sigma \rightsquigarrow \sigma' \tag{19}$$

between consecutive sets of standardized local configurations. On the one hand, considering the left part of Fig. 8, we note that the two horizontal arrows represent the definitions (18) of the set $\Sigma^{(\ell)}$, while the left vertical arrow corresponds to Eq. (17). Lemma 1 then confirms that the diagram commutes, hence

$$\Sigma^{(\ell+1)} = \textsc{Extend\&Standardize}(\textsc{Insert\&Extend}(s^{(\ell+1)}, \Sigma^{(\ell)})) \; .$$

On the other hand, the right-hand side of the figure visualizes the definition of the set of all the successors of $\Sigma^{(\ell)}$,

$$\Sigma^{(\ell)}_{\rightsquigarrow} = \{\sigma' \,|\, \exists \sigma \in \Sigma^{(\ell)} \, : \, \sigma \rightsquigarrow \sigma'\} \; ,$$

where $\hat{S}$ is the set of all the standardized splits. Comparing it with the marked portion of the left part of the figure confirms (19) by noting that $\Sigma^{(\ell+1)} \subset \Sigma^{(\ell)}_{\rightsquigarrow}$ since $\hat{s}^{(\ell+1)} \subset \hat{S}$.

Second we use this observation to conclude that there exist standardized local configurations $\sigma^{(1)} \in \Sigma^{(1)}, \ldots, \sigma^{(n-1)} \in \Sigma^{(n-1)}$ such that

$$\tau \rightsquigarrow \sigma^{(1)} \rightsquigarrow \sigma^{(2)} \cdots \rightsquigarrow \sigma^\star \; .$$

This implies $\sigma^\star \in \langle \tau \rangle$, which is in contradiction with Lemma 2.

An analysis of the local configurations confirms that there is one basis function for each cross vertex in the mesh. Consequently, the B-splines span the full bilinear spline space associated with the T-mesh as their number is equal to the dimension of the spline space, according to the dimension formula of Deng et al. (2006, Theorem 4.2). □

6. Experimental results

In order to validate the effectiveness of the proposed algorithm, we carry out adaptive approximation of three functions defined on the domain $[1,5] \times [1,5]$. The refinement is initialized by a tensor-product mesh consisting of 4×4 cells, with 25 bilinear B-splines. All the numerical tests were performed in C++ using the library `G+Smo` (Mantzaflaris, 2020).

Given the function, the error tolerance ε and the local semi-regularity condition, the adaptive approximation procedure repeats the following steps:

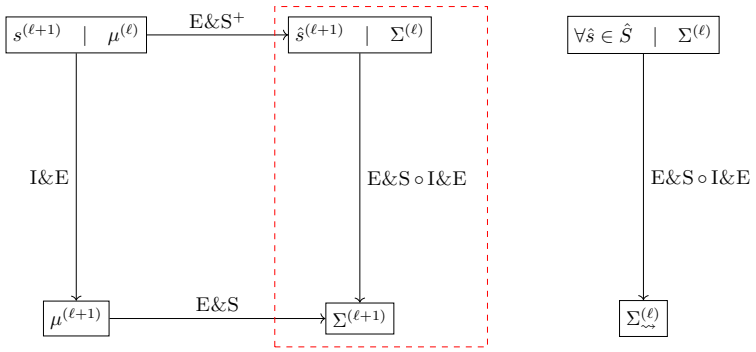


Figure 8: Using Lemma 1 to prove the relation (19), where the labels E&S and I&E represent the algorithms Extract&Standardize and Insert&Extend, respectively.

1. Mark the cell with the largest approximation error. If this error does not exceed ε, then terminate the entire procedure.
2. Construct the corresponding split. This step is described in detail below.
3. Perform the Algorithm Insert&Extend with the constructed split and generate the associated approximating spline.

Recall that the support extension $\hat{c}$ of a cell c is formed by the union of all the supports of the active basis functions on this cell, cf. (11). Given the cell and the direction of the split, the split s is chosen such that it traverses the entire support extension, i.e., such that it satisfies $s = s^+ \cap \hat{c}$ with s^+ as defined in (15). In addition, it splits the cell into two smaller cells having (roughly) equal size. The direction of the split is determined by comparing the approximation errors of the bilinear interpolants on the vertically or horizontally splitted cell c. We select the horizontal split if the approximation error on the horizontally splitted two subcells is smaller than the one in the vertically splitted two subcells, and the vertical split otherwise. This simple direction indicator, which neglects the effect of T-joints that might be present, worked sufficiently well in all examples.

6.1. The effect of local semi-regularity (LSR)

First we analyze the effect of LSR condition. In addition to visually judging the quality of the meshes, we introduce the quantity

$$\mathrm{RER} = \frac{|B^{(n)}|}{2n + 25} - 1 \ ,$$

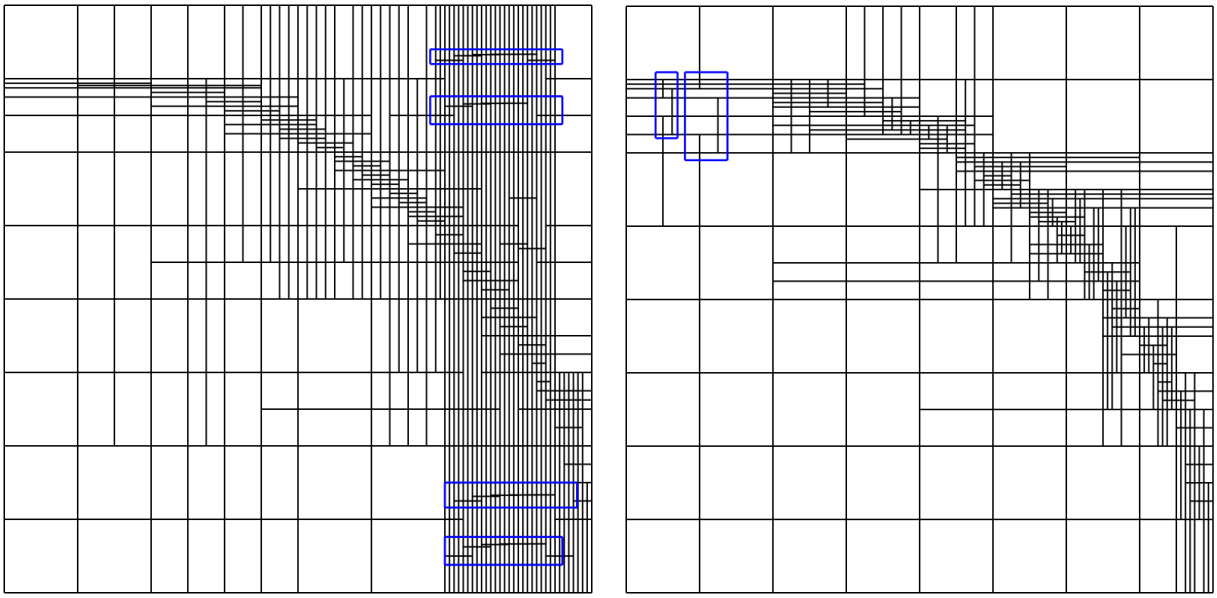
where n is the total number of refinement steps and $|B^{(n)}|$ is the number of basis functions, measuring the *rate of excess refinement* that is caused by performing the split extensions. The initial mesh possesses 25 basis functions and every refinement step creates at least two new basis functions since it introduces two new cross or boundary vertices in the mesh. All the additional basis functions are considered as excess refinement caused by the split extensions.

Circular arc example. We perform adaptive approximation of the function

$$f(x, y) = |(x - 1.5)^2 + (y - 1)^2 - 3.5^2| \tag{20}$$

using both versions of the algorithm INSERT&EXTEND. The function is not smooth along a circular arc. The tolerance ε is chosen as 0.15 in this example.

Fig. 9(a) and Fig. 9(b) show the resulting meshes obtained by switching the LSR condition on or off. It demonstrates that using this condition results in a large number of additional refinements, and that it essentially prevents truly local refinement. The numbers (including RER, numbers of cells and basis functions) listed in Table 1 also support this observation. Without LSR, the algorithm creates a mesh where both the shape and the size of the cells is adapted to the specific approximation problem. We get smaller cells along the circular arc, which are aligned with its direction as far as possible.



(a) LSR = "on" (b) LSR = "off"

Figure 9: Circular arc example: The obtained meshes.

6.2. Improvement of mesh quality via dyadic splitting

A closer examination of the meshes in Fig. 9 reveals that certain line segments are not aligned, see the regions marked in blue. This is caused by the selection of the split, which always subdivides a cell into two smaller cells of exactly the same size. This results in additional T-joints.

In order to avoid this phenomenon, we use dyadic splits during the refinement process, as follows. In order to perform a vertical split of a cell $[a, b] \times [c, d]$, we find the lowest level

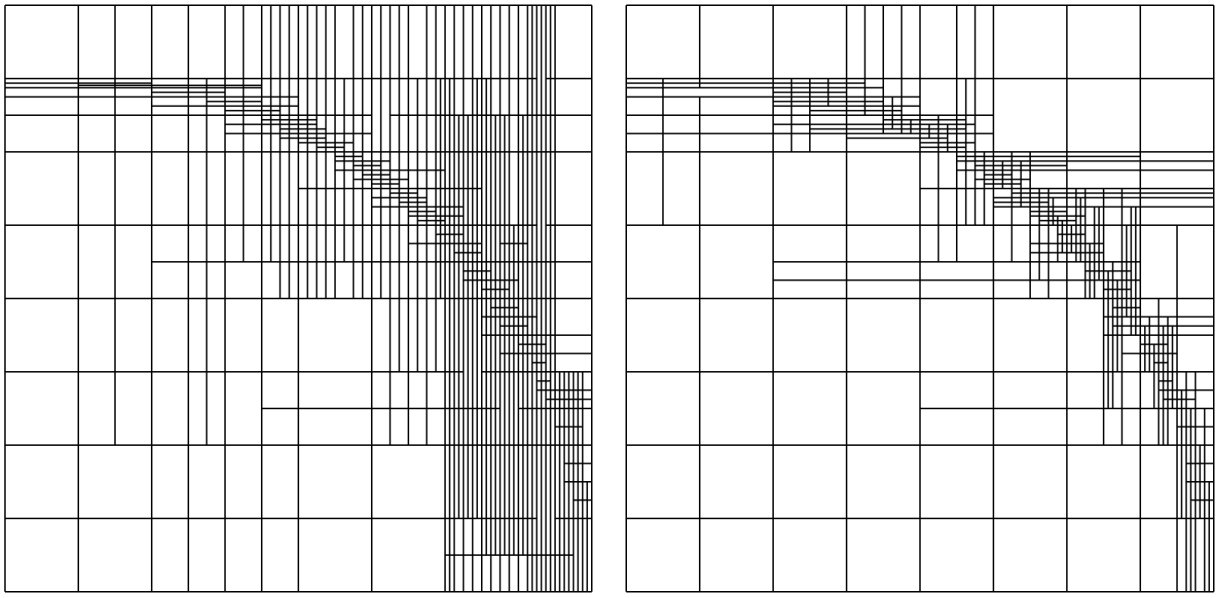
$$\ell = -\lfloor \log_2(b - a) \rfloor$$

Example	Meshes	LSR	DS	$\lvert C^{(n)}\rvert$	$\lvert B^{(n)}\rvert$	# extensions	RER
Circular arc	Fig. 9(a)	on	off	935	903	314	173 %
	Fig. 9(b)	off	off	598	535	76	86 %
Circular arc – revisited	Fig. 10(a)	on	on	858	832	287	153 %
	Fig. 10(b)	off	on	595	535	76	86 %
Three peak example	Fig. 11(a)	on	on	722	701	253	179 %
	Fig. 11(b)	off	on	552	483	44	78 %
Diagonal refinement	Fig. 12(a)	on	on	1,882	1,813	941	248 %
	Fig. 12(b)	off	on	1,134	1,003	121	93 %

Table 1: Information (including the numbers of cells, basis functions (degrees of freedom) and extensions, and the rate of excess refinement) about the resulting meshes. DS indicates the use of dyadic splitting.

that satisfies $b - a > \frac{1}{2^\ell}$. The x-coordinate of the split is chosen as a number in the set $\mathbb{Z}/2^\ell \cap (a, b)$, which possesses either one or two elements. In the latter case we pick the one that is closer to $(a + b)/2$. The y-coordinate of a horizontal split is found analogously.

Circular arc example – revisited. We perform the same computations as in the previous example, but now using dyadic splitting. The resulting meshes are depicted in Fig. 10. We note that dyadic splitting significantly improves the vertical/horizontal alignment of the inserted line segments. The numbers reported in Table 1 also indicate the improved performance. Without LSR, the algorithm again creates a mesh where both the shape and the size of the cells is adapted to the specific approximation problem.



(a) LSR = "on" (b) LSR = "off"

Figure 10: Circular arc example – revisited: Using dyadic splitting improves the mesh quality.

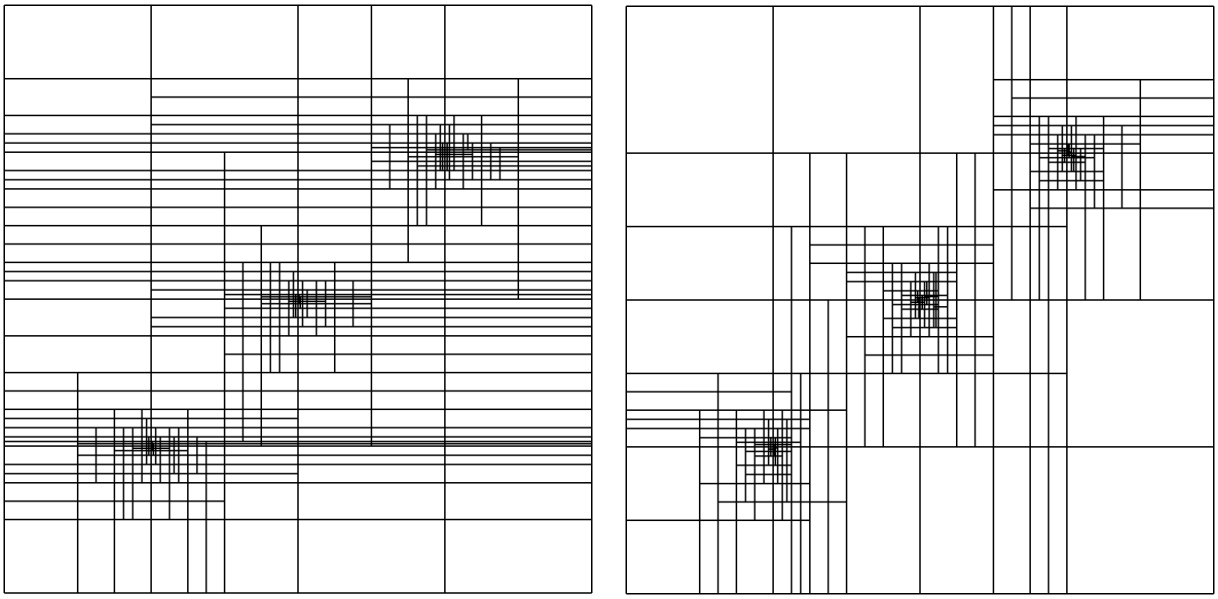
6.3. Further examples

We always use dyadic refinement in the remaining examples.

Three peak function. The second function we consider is

$$f(x,y) = \sum_{i=2}^{4} \exp(-10\sqrt{(x-i)^2 + (y-i)^2}) \tag{21}$$

with three peaks around the points $(2,2)$, $(3,3)$ and $(4,4)$. The error tolerance ε is chosen as 0.02 in this test. Fig. 11 depicts the resulting meshes of approximating this function with and without LSR, respectively. It indicates that this condition leads to more cells and basis functions as shown in Table 1. It also reveals that the inserted splits are well aligned since we use dyadic splitting. Local refinement works reasonably well if LSR is "off".



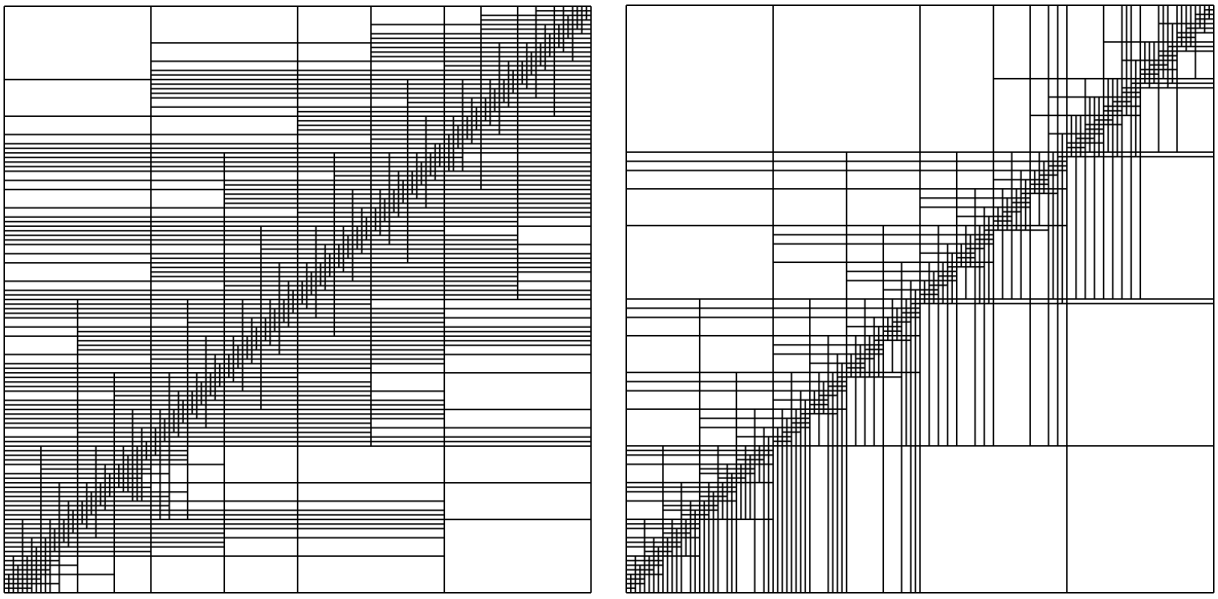
(a) LSR = "on" (b) LSR = "off"

Figure 11: The resulting meshes obtained when approximating the three peak function (21).

Diagonal refinement. Next we consider the example

$$f(x,y) = |x-y| \tag{22}$$

which is non-smooth along the diagonal $x - y = 0$. The tolerance ε is chosen as 0.02. The results presented in Fig. 12 and the numbers listed in Table 1 again show that the LSR condition induces excess refinements. Local refinement works reasonably well if LSR is "off". The shape of some of the cells is quite extreme, since we did not (yet) put any constraint on the size of the cells.



(a) LSR = "on" (b) LSR = "off"

Figure 12: The resulting meshes obtained when approximating the function (22) via diagonal refinement.

6.4. Extension to the higher degree splines

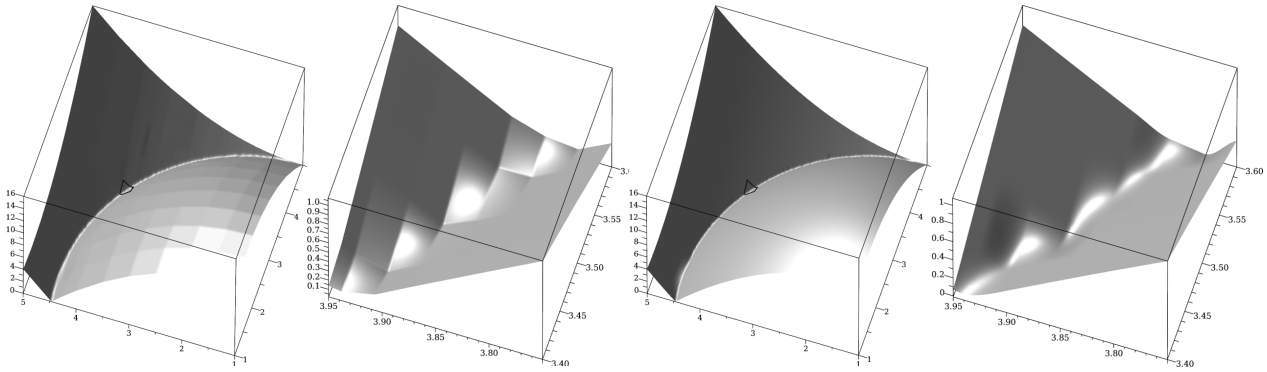
By generalizing the definition of bilinear B-splines in Eqns. (1) and (2), we consider four bicubic B-splines $\beta_{(X,Y)}(u,v) = N_X^3(u)N_Y^3(v)$, with the local knot vectors

$$\begin{aligned}(X,Y) \;\in\; \{&((x,x,x',x',x''),(y,y,y',y',y'')),\ ((x,x,x',x',x''),(y,y',y',y'',y'')),\\ &((x,x',x',x'',x''),(y,y,y',y',y'')),\ ((x,x',x',x'',x''),(y,y',y',y'',y''))\}\end{aligned}$$

for each pair $((x,x',x''),(y,y',y'')) \in K^{(\ell)}$. Note that we use knot multiplicity 2 for all the inner knots, thus C^1-smooth B-splines.

Based on the mesh shown in Fig. 10(b), which was created when approximating the circular arc function (20) using bilinear B-splines, we approximate the circular arc function using the generated bicubic B-splines via Hermite interpolation. More precisely, the approximating spline and its derivatives match the values of f, f_x, f_y and f_{xy} at all the cross and boundary vertices of the hierarchical T-mesh.

Fig. 13 presents the plots of linear and cubic spline functions that approximate the circular arc function. It demonstrates that the refinement algorithm Insert&Extend presented in this paper is not limited to the bilinear case, but can be applied to splines of higher degrees and smoothness as well. Indeed, the generated meshes admit the construction of splines of degree $2s+1$ that are C^s-smooth, simply by considering $(s+1)^2$ B-splines for each element of K^ℓ.



(a) Bilinear approximation (b) Enlarged section (c) Bicubic approximation (d) Enlarged section

Figure 13: For the mesh shown in Fig. 10(b), the C^0-smooth bilinear and C^1-smooth bicubic spline approximations are depicted in (a) and (c). Enlarged sections of the surfaces (marked in black) are presented in (b) and (d), respectively.

7. Conclusion

We have presented a simple algorithm that produces hierarchical T-meshes by repeated split insertion. It ensures that the associated spline spaces are equipped with bases consisting of tensor-product B-splines over local knot vectors that possess the properties of local linear independence and partition of unity. As its main advantage, our method is able to adapt both the size and the shape of the cells, while guaranteeing the good properties of the basis.

Since the current framework is limited to C^s-smooth splines of degree $p = 2s + 1$, we plan to explore the generalization to other values of the degree in our future work. Clearly, the extension to the trivariate case would also be highly interesting.

References

Bazilevs, Y., Calo, V.M., Cottrell, J.A., Evans, J.A., Hughes, T.J.R., Lipton, S., Scott, M.A., Sederberg, T.W., 2010. Isogeometric analysis using T-splines. Comput. Meth. Appl. Mech. Eng. 199, 229–263.

Bracco, C., Lyche, T., Manni, C., Roman, F., Speleers, H., 2016. Generalized spline spaces over T-meshes: Dimension formula and locally refined generalized B-splines. Applied Mathematics and Computation 272, 187–198.

Bressan, A., 2013. Some properties of LR-splines. Comput. Aided Geom. Des. 30, 778–794.

Bressan, A., Jüttler, B., 2015. A hierarchical construction of LR meshes in 2D. Comput. Aided Geom. Des. 37, 9–24.

Deng, J., Chen, F., Feng, Y., 2006. Dimensions of spline spaces over T-meshes. Journal of Computational and Applied Mathematics 194, 267–283.

Deng, J., Chen, F., Li, X., Hu, C., Tong, W., Yang, Z., Feng, Y., 2008. Polynomial splines over hierarchical T-meshes. Graphical Models 70, 76–86.

Dokken, T., Lyche, T., Pettersen, K.F., 2013. Polynomial splines over locally refined box-partitions. Comput. Aided Geom. Des. 30, 331–356.

Forsey, D.R., Bartels, R.H., 1988. Hierarchical B-spline refinement, in: Proceedings of the 15th annual conference on Computer Graphics and Interactive Techniques, pp. 205–212.

Giannelli, C., Jüttler, B., Speleers, H., 2012. THB-splines: The truncated basis for hierarchical splines. Comput. Aided Geom. Des. 29, 485–498.

Giust, A., Jüttler, B., Mantzaflaris, A., 2020. Local (T)HB-spline projectors via restricted hierarchical spline fitting. Comput. Aided Geom. Des. 80, 101865.
Johannessen, K.A., Kvamsdal, T., Dokken, T., 2014. Isogeometric analysis using LR B-splines. Comput. Meth. Appl. Mech. Eng. 269, 471–514.
Kang, H., Xu, J., Chen, F., Deng, J., 2015. A new basis for PHT-splines. Graphical Models 82, 149–159.
Kiss, G., Giannelli, C., Zore, U., Jüttler, B., Großmann, D., Barner, J., 2014. Adaptive CAD model (re-) construction with THB-splines. Graphical Models 76, 273–288.
Kraft, R., 1998. Adaptive und linear unabhängige Multilevel B-Splines und ihre Anwendungen. Ph.D. thesis. Universität Stuttgart.
Li, X., Chen, F., Kang, H., Deng, J., 2016. A survey on the local refinable splines. Science China Mathematics 59, 617–644.
Li, X., Deng, J., Chen, F., 2007. Surface modeling with polynomial splines over hierarchical T-meshes. The Visual Computer 23, 1027–1033.
Li, X., Zheng, J., Sederberg, T.W., Hughes, T.J., Scott, M.A., 2012. On linear independence of T-spline blending functions. Comput. Aided Geom. Des. 29, 63–76.
Mantzaflaris, A., 2020. An overview of Geometry Plus Simulation Modules, in: Slamanig, D., Tsigaridas, E., Zafeirakopoulos, Z. (Eds.), Mathematical Aspects of Computer and Information Sciences. Springer, Cham. volume 11989 of *Lecture Notes in Computer Science*, pp. 453–456.
Ni, Q., Wang, X., Deng, J., 2019. Modified PHT-splines. Comput. Aided Geom. Des. 73, 37–53.
Patrizi, F., Dokken, T., 2020. Linear dependence of bivariate Minimal Support and Locally Refined B-splines over LR-meshes. Comput. Aided Geom. Des. 77, 101803.
Patrizi, F., Manni, C., Pelosi, F., Speleers, H., 2020. Adaptive refinement with locally linearly independent LR B-splines: Theory and applications. Comput. Meth. Appl. Mech. Eng. 369, 113230.
Scott, M.A., Li, X., Sederberg, T.W., Hughes, T.J., 2012. Local refinement of analysis-suitable T-splines. Comput. Meth. Appl. Mech. Eng. 213, 206–222.
Sederberg, T.W., Zheng, J., Bakenov, A., Nasri, A., 2003. T-splines and T-NURCCs. ACM Transactions on Graphics 22, 477–484.
Skytt, V., Barrowclough, O., Dokken, T., 2015. Locally refined spline surfaces for representation of terrain data. Computers & Graphics 49, 58–68.
Wang, P., Xu, J., Deng, J., Chen, F., 2011. Adaptive isogeometric analysis using rational PHT-splines. Computer-Aided Design 43, 1438–1448.
Wang, W., Zhang, Y., Liu, L., Hughes, T., 2013. Trivariate solid T-spline construction from boundary triangulations with arbitrary genus topology. Computer-Aided Design 45, 351–360.
Wei, X., Zhang, Y., Liu, L., Hughes, T.J., 2017. Truncated T-splines: fundamentals and methods. Comput. Meth. Appl. Mech. Eng. 316, 349–372.
Weller, F., Hagen, H., 1995. Tensor product spline spaces with knot segments, in: Dæhlen, M., Lyche, T., Schumaker, L. (Eds.), Mathematical Methods for Curves and Surfaces. Vanderbilt University Press, Nashville, pp. 563–572.
Zheng, Y., Chen, F., 2022. Volumetric parameterization with truncated hierarchical B-splines for isogeometric analysis. Comput. Meth. Appl. Mech. Eng. 401, 115662.
Zhu, Y., Chen, F., 2017. Modified bases of PHT-splines. Commun. Math. Stat. 5, 381–397.

Appendix: The 385 standardized local configurations, descendants of the tensor-product configuration τ, which are used in the proof of Lemma 2