
Article:
May, S., Vignollet, J. and de Borst, R. orcid.org/0000-0002-3457-3574 (2015) The role of the Bézier extraction operator for T-splines of arbitrary degree: linear dependencies, partition of unity property, nesting behaviour and local refinement. International Journal for Numerical Methods in Engineering, 103 (8). pp. 547-581. ISSN 0029-5981

https://doi.org/10.1002/nme.4902

INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 0000; **00**:1–37
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nme

The role of the Bézier extraction operator for T-splines of arbitrary degree: linear dependencies, partition of unity property, nesting behaviour, and local refinement

Stefan May[1]*, Julien Vignollet[1], René de Borst[1]

[1]*University of Glasgow, School of Engineering, Rankine Building, Oakfield Avenue, Glasgow G12 8LT, UK.*

SUMMARY

We determine linear dependencies and the partition of unity property of T-spline meshes of arbitrary degree using the Bézier extraction operator. Local refinement strategies for standard, semi-standard and non-standard T-splines – also by making use of the Bézier extraction operator – are presented for meshes of even and odd polynomial degree. A technique is presented to determine the nesting between two T-spline meshes, again exploiting the Bézier extraction operator. Finally, the hierarchical refinement of standard, semi-standard and non-standard T-spline meshes is discussed. This technique utilises the reconstruction operator, which is the inverse of the Bézier extraction operator. Copyright © 0000 John Wiley & Sons, Ltd.

Received ...

KEY WORDS: T-splines, isogeometric analysis, Bézier extraction, linear dependency, partition of unity, hierarchical refinement

1. INTRODUCTION

Isogeometric analysis was introduced in [1]. It is based on the concept that the same shape functions are used to represent the geometry and to approximate the field variables. Initially, Non-Uniform Rational B-Splines (NURBS) have been used as shape functions in isogeometric analysis. Since NURBS have a tensor product structure, refinement occurs globally. Furthermore, it can be difficult to model watertight surfaces with NURBS patches. T-splines, which can be conceived as a generalisation of NURBS, were introduced in [2, 3] and do not suffer from the limitations that are inherent in NURBS. Local refinement is now possible and watertight surfaces can be created. Moreover, T-splines allow for the reduction of superfluous control points. Use of T-spline blending functions as shape functions in a finite element context was proposed in [4, 5].

NURBS and T-splines meet a growing acceptance in the engineering community, which is considerably facilitated by the technique of Bézier extraction [6, 7]. Bézier extraction allows for an implementation that is identical to that typically used in finite element codes. However, in [8] the concern was raised that for T-spline meshes, linear independence – which is a necessary condition to perform the analysis – is not an inherent property of the blending functions. In [9], a definition for analysis-suitable T-spline meshes was proposed which results in a mildly restricted subset of T-splines. A topological algorithm was developed as well: a T-spline mesh was deemed analysis-suitable when there are no two orthogonal T-node extensions which intersect in the extended T-spline mesh. A considerable amount of research has been spent since then on the

*Correspondence to: Stefan May, University of Glasgow, School of Engineering, Oakfield Avenue, Rankine Building, Glasgow G12 8LT, UK. E-mail: s.may.2@research.gla.ac.uk

properties of analysis-suitable T-spline meshes [10–12]. In [13] an algorithm based on the T-spline mesh topology was presented to refine analysis-suitable T-spline meshes. Recently, a hierarchical refinement algorithm for analysis-suitable T-splines based on the reconstruction operator was introduced in [14, 15]. Furthermore, the partition of unity property and linear dependencies for T-splines without multiple knots were investigated in [16] and [17], respectively.

Using the Bézier extraction procedure [7], each blending function can be defined in a normalised fashion by a linear combination of Bernstein polynomials. We will show that linear dependencies and the partition of unity property can be determined for T-spline meshes with the Bézier extraction operator at hand. It will be demonstrated that this approach can be applied to T-spline meshes of arbitrary degree. The Bézier extraction operator also enables to determine the nesting behaviour between two T-spline meshes. Moreover, we show how standard, semi-standard and non-standard T-spline meshes can be refined locally using information from the Bézier extraction operator.

This paper is organised as follows. In the first section we give a concise description of T-splines. Next, we present a brief overview on the construction of the Bézier extraction operator for T-splines. Subsequently, linear dependence and the partition of unity property of T-spline meshes are investigated using the Bézier extraction operator. In Section 5 a refinement method is proposed for T-spline meshes by adding anchors while the Bézier extraction operator is utilised for the determination of the nesting behaviour between two T-spline meshes. The capabilities of the method are demonstrated for meshes of even and of odd polynomial degree. Finally, a technique is introduced to refine hierarchically standard, semi-standard and non-standard T-spline meshes.

2. T-SPLINES

This section provides a brief overview of T-splines. For a more elaborate demonstration of T-splines in a finite element environment we refer to [5]. Note, that herein we limit ourselves to two-dimensional problems but the methods developed in this paper can also be used in three dimensions – the only requirement is that we are able to elaborate the Bézier extraction operator. Index notation is adopted throughout with respect to a Cartesian frame.

2.1. Definition of the domains

In Figure 1 the physical domain (x_ℓ), the parent domain ($\tilde{\xi}_\ell$), the index domain (u_ℓ), the parameter domain (ξ_ℓ^u), and the sub-parameter domain (ξ_ℓ) are shown for T-splines. Each element e can be mapped from the physical domain x_ℓ onto the parent domain $\tilde{\xi}_\ell \in [-1, 1]$, where Gaussian integration can be carried out. The sub-parameter domain ξ_ℓ is obtained when only the unique values of the parameter domain ξ_ℓ^u are considered.

2.2. Definition of the local knot vector

The index domain in Figure 1 represents a tiling of a region in $\mathbb{R}^2$ while all edges of each rectangle have a positive integer value. T-spline meshes of odd and of even polynomial order have to be treated differently when defining the local knot vectors from the parameter domain. The local knot vectors are necessary to define the blending functions, see Section 2.3.

For a T-spline mesh of even degree p_ℓ in both directions, a so-called anchor – to which a single multivariate blending function is attached – is placed in the centre of each rectangle, see the quadratic T-spline mesh in Figure 2(a). A local knot vector for a T-spline mesh of even degree is obtained from the parameter domain by – starting at the anchor – marching horizontally (both left and right) and vertically (both up and down), until a number of $p_\ell/2 + 1$ edges are crossed in all four directions, thus giving a vector length of $p_\ell + 2$. Every time an edge is crossed, the corresponding parameter value is added to the local knot vector. If fewer than $p_\ell/2 + 1$ edges are crossed, and there are no more edges left to be crossed, the parameter value that has been added last is repeated until $p_\ell/2 + 1$ parameter values are added in this direction. For the blue anchor A sitting at $(3.5, 5.5)$ in the index domain in Figure 2(a), the local knot vectors are $\Xi_1^A = \{0, \frac{1}{2}, 1, 1\}$ and $\Xi_2^A = \{\frac{1}{3}, \frac{2}{3}, 1, 1\}$, respectively.

DOI: 10.1002/nme

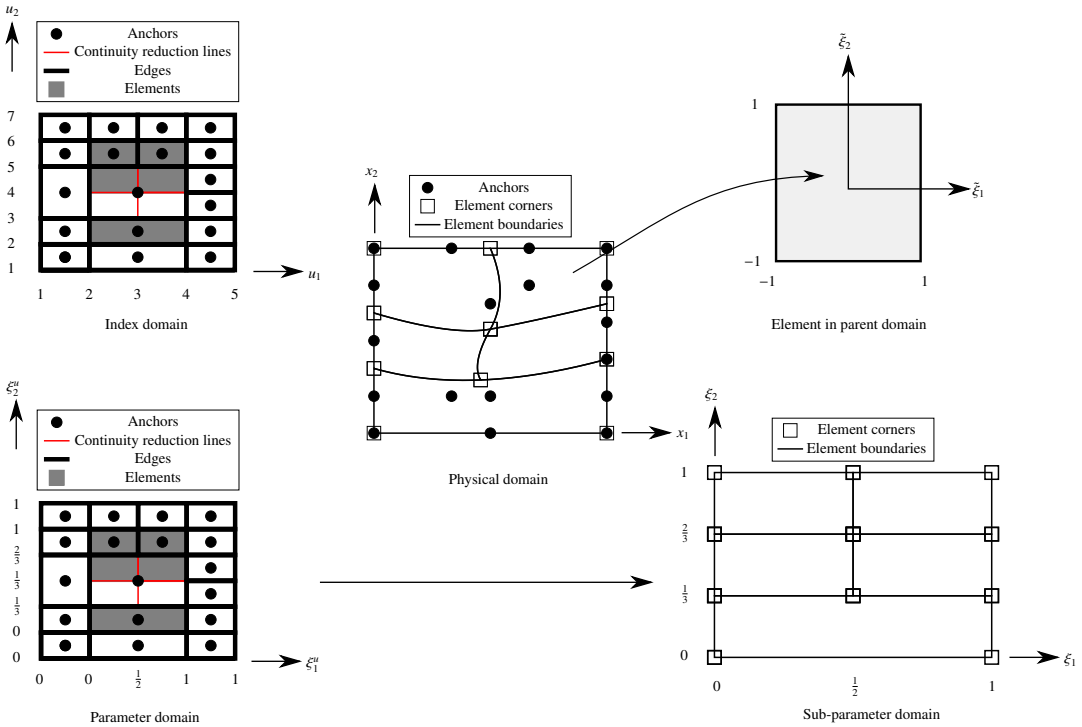


Figure 1. Illustration of the physical domain (x_ℓ), the parent domain ($\tilde{\xi}_\ell$), the index domain (u_ℓ), the parameter domain (ξ^u_ℓ), and the sub-parameter domain (ξ_ℓ) on a quadratic T-spline mesh.

For a T-spline mesh of odd degree p_ℓ in both directions, anchors are located at the vertices of the rectangles, see the cubic T-spline mesh in Figure 2(b). In order to obtain the local knot vector of an anchor, the parameter ξ^u_ℓ at the vertex is added to the local knot vectors for each direction. Afterwards, we march again – starting at the location of the anchor – horizontally to the right and left, and vertically up and down, until $(p_\ell + 1)/2$ edges have been crossed in all four directions, thus yielding again a local knot vector of length $p_\ell + 2$. If there are no more edges to be crossed, then the value of the last added parameter is repeated until $(p_\ell + 1)/2$ values are added in this direction to the local knot vector. Consider, for instance, the blue anchor B sitting at $(2, 2)$ in the index domain for the cubic T-spline mesh in Figure 2(b). The local knot vectors are $\Xi^B_1 = \{0, 0, 0, 1, 1\}$ and $\Xi^B_2 = \{0, 0, 0, \frac{1}{3}, \frac{2}{3}\}$.

2.3. Construction of the blending functions

Let us consider a T-spline mesh containing n anchors. Each anchor i is equipped with a single multivariate blending function N^i. Each multivariate blending function N^i is defined in the sub-parameter domain ξ_ℓ as follows

$$N^i(\underline{\boldsymbol{\xi}}) = \prod_{\ell=1}^{d} N^i_\ell(\xi_\ell) \tag{1}$$

with the univariate blending functions N^i_ℓ for each anchor i and the dimension d. The univariate blending function N^i_ℓ of order p_ℓ for anchor i is given by

$$N^i_\ell(\xi_\ell) = N^i_{\ell\ 1,p_\ell}(\xi_\ell) \tag{2}$$

DOI: 10.1002/nme

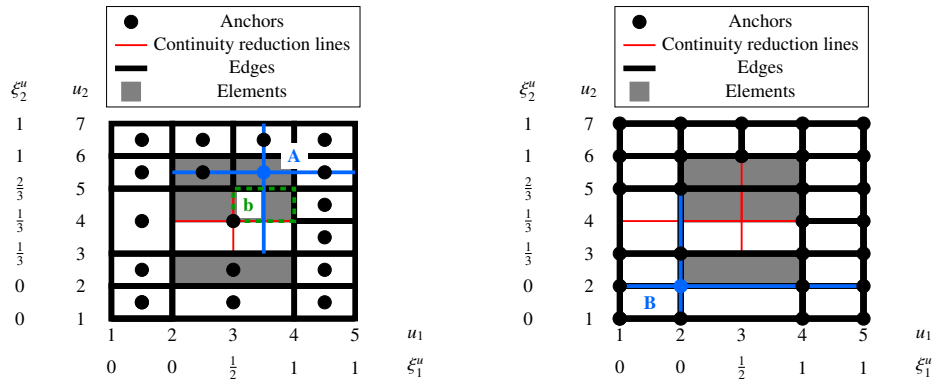


Figure 2. Determination of the local knot vectors for a T-spline mesh of (a) even (quadratic, $p_\ell = 2$) and (b) odd (cubic, $p_\ell = 3$) degree: every time an edge is crossed in all four directions, the corresponding parameter value is added to the local knot vector. (a) The local knot vectors for the blue anchor A are $\Xi_1^A = \{\xi_1^2, \xi_1^3, \xi_1^4, \xi_1^5\} = \{0, \frac{1}{2}, 1, 1\}$ and $\Xi_2^A = \{\xi_2^3, \xi_2^5, \xi_2^6, \xi_2^7\} = \{\frac{1}{3}, \frac{2}{3}, 1, 1\}$, (b) The local knot vectors for the blue anchor B are $\Xi_1^B = \{\xi_1^1, \xi_1^1, \xi_1^2, \xi_1^4, \xi_1^5\} = \{0, 0, 0, 1, 1\}$ and $\Xi_2^B = \{\xi_2^1, \xi_2^1, \xi_2^2, \xi_2^3, \xi_2^5\} = \{0, 0, 0, \frac{1}{3}, \frac{2}{3}\}$.

where the $N^i_{\ell\ a,p_\ell}$ (with $a = 1$ the single blending function for anchor i is obtained) can be defined with the local knot vector $\Xi^i_\ell = \{\xi^i_{\ell\ 1}, \xi^i_{\ell\ 2}, \ldots, \xi^i_{\ell\ p_\ell+2}\}$ of anchor i for $p_\ell = 0$ with

$$N^i_{\ell\ a,0}(\xi_\ell) = \begin{cases} 1 & \text{if } \xi^i_{\ell\ a} \leq \xi_\ell < \xi^i_{\ell\ a+1} \\ 0 & \text{otherwise} \end{cases}. \tag{3}$$

For $p_\ell \geq 1$ they are given by the Cox - de Boor recursion formula [18, 19]

$$N^i_{\ell\ a,p_\ell}(\xi_\ell) = \frac{\xi_\ell - \xi^i_{\ell\ a}}{\xi^i_{\ell\ a+p_\ell} - \xi^i_{\ell\ a}} N^i_{\ell\ a,p_\ell-1}(\xi_\ell) + \frac{\xi^i_{\ell\ a+p_\ell+1} - \xi_\ell}{\xi^i_{\ell\ a+p_\ell+1} - \xi^i_{\ell\ a+1}} N^i_{\ell\ a+1,p_\ell-1}(\xi_\ell). \tag{4}$$

Herein we will only consider cases with an equal polynomial order p_ℓ in the ξ_1 direction and the ξ_2 direction.

2.4. Element definition

The red anchor A with index coordinates $(3.5, 5.5)$ for the quadratic T-spline mesh in Figure 3(a) has the local knot vectors $\Xi_1^A = \{\xi_1^2, \xi_1^3, \xi_1^4, \xi_1^5\}$ and $\Xi_2^A = \{\xi_2^3, \xi_2^5, \xi_2^6, \xi_2^7\}$. Anchor A has non-zero blending functions in the green parameter domain $[\xi_1^2, \xi_1^5] \times [\xi_2^3, \xi_2^7]$. Within this domain, the net of red dashed lines depicted in Figure 3(a) is obtained upon drawing all the values contained in the local knot vectors Ξ_ℓ^A. Along those lines, we have a reduced continuity, which is indicated by a multiplicity larger than zero in the local knot vectors. If one of these lines is not already an edge, this line is added to the T-spline mesh, see Figure 3(b). The added line is called a continuity reduction line. For T-splines, elements are defined by the union of all edges and continuity reduction lines with non-zero parametric area in the parameter space ξ^u_ℓ, see also Figure 2(a).

3. BÉZIER EXTRACTION FOR T-SPLINES

For details on the Bézier extraction method for T-splines, reference is made to [7]. Here, we give a succinct summary on the calculation of the Bézier extraction operator and illustrate the method by means of an example.

DOI: 10.1002/nme

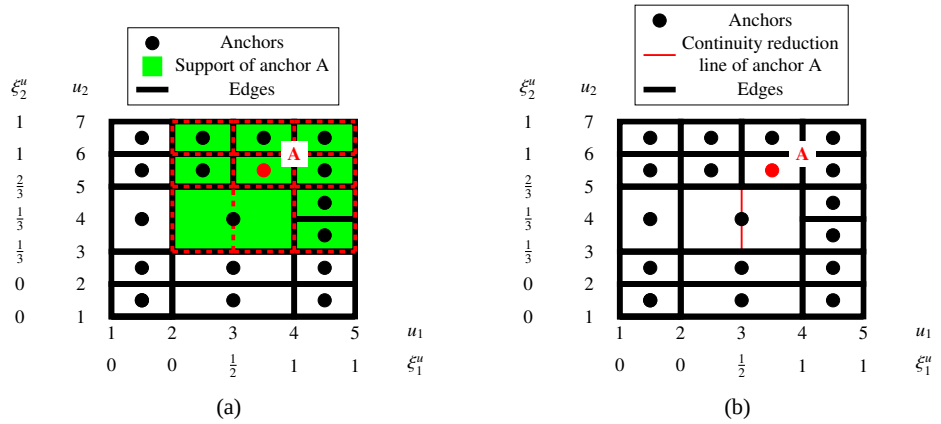


Figure 3. Continuity reduction lines: consider the red anchor A with index coordinates $(3.5, 5.5)$. (a) This anchor has a support (non-zero blending functions) in the green shaded domain $[\xi_1^2, \xi_1^5] \times [\xi_2^3, \xi_2^7]$. Drawing all the values contained in the local knot vectors $\Xi_1^A = \{\xi_1^2, \xi_1^3, \xi_1^4, \xi_1^5\}$ and $\Xi_2^A = \{\xi_2^3, \xi_2^5, \xi_2^6, \xi_2^7\}$ gives the net of dashed red lines. (b) If a red dashed line in (a) is not already an edge then it is added to the T-spline mesh.

We suppose that the domain is divided into E elements. Then, the blending function N_e^i of anchor i over element e can be written as a linear combination of the Bernstein polynomials

$$N_e^i(\underline{\boldsymbol{\xi}}) = \underline{\boldsymbol{C}}_e^{i\,T} \underline{\boldsymbol{B}}_e(\underline{\boldsymbol{\xi}}) \tag{5}$$

where the $(p_\ell + 1)^2$ bivariate Bernstein polynomials $\underline{\boldsymbol{B}}_e$ for element e are expressed as follows

$$\underline{\boldsymbol{B}}_e(\underline{\boldsymbol{\xi}}) = \begin{bmatrix} B_{1\,e}^1(\xi_1) B_{2\,e}^1(\xi_2) \\ \vdots \\ B_{1\,e}^{p_\ell+1}(\xi_1) B_{2\,e}^1(\xi_2) \\ \vdots \\ B_{1\,e}^{p_\ell+1}(\xi_1) B_{2\,e}^{p_\ell+1}(\xi_2) \end{bmatrix}. \tag{6}$$

The bivariate Bernstein polynomials $\underline{\boldsymbol{B}}_e$ are equal for each element e in the parent domain $\tilde{\xi}_\ell$. A univariate blending function $N_{\ell\,e}^i$ of anchor i over element e can be expressed in terms of the univariate Bernstein basis $B_{\ell\,e}^a$ with

$$N_{\ell\,e}^i(\xi_\ell) = \begin{bmatrix} C_{\ell\,e}^{i\,1} & \dots & C_{\ell\,e}^{i\,p_\ell+1} \end{bmatrix} \begin{bmatrix} B_{\ell\,e}^1(\xi_\ell) \\ \vdots \\ B_{\ell\,e}^{p_\ell+1}(\xi_\ell) \end{bmatrix}, \tag{7}$$

where $C_{\ell\,e}^{i\,a}$ are the coefficients for anchor i and element e corresponding to $B_{\ell\,e}^a$. The $a = 1 \dots p_\ell + 1$ univariate Bernstein polynomials B_ℓ^a of order p_ℓ are defined over the interval $\tilde{\xi}_\ell \in [-1, 1]$ by

$$B_\ell^a(\tilde{\xi}_\ell) = \frac{1}{2^{p_\ell}} \binom{p_\ell}{a-1} (1 - \tilde{\xi}_\ell)^{p_\ell - (a-1)} (1 + \tilde{\xi}_\ell)^{a-1}. \tag{8}$$

The univariate Bernstein polynomials $B_{\ell\,e}^a$ are equal in the parent domain $\tilde{\xi}_\ell$ for each element e in each direction. In Equation (5), $\underline{\boldsymbol{C}}_e^i$ is the Bézier extraction operator of anchor i with support over

DOI: 10.1002/nme

element e

$$\underline{\boldsymbol{C}}_e^i = \begin{bmatrix} C_{1\,e}^{i\,1} C_{2\,e}^{i\,1} \\ \vdots \\ C_{1\,e}^{i\,p_\ell+1} C_{2\,e}^{i\,1} \\ \vdots \\ C_{1\,e}^{i\,p_\ell+1} C_{2\,e}^{i\,p_\ell+1} \end{bmatrix} . \tag{9}$$

To illustrate the notation, we again consider the anchor A at $(3.5, 5.5)$ in Figure 2(a). The local knot vectors are $\Xi_1^{\mathrm{A}} = \{0, \frac{1}{2}, 1, 1\}$ and $\Xi_2^A = \{\frac{1}{3}, \frac{2}{3}, 1, 1\}$ for the ξ_1 direction and the ξ_2 direction, respectively. We now evaluate, for anchor A, the Bézier extraction operator over the element b in Figure 2(a) with range $[3, 4] \times [4, 5]$ in the index domain. The range for the element b is $[\xi_1^3, \xi_1^4] \times [\xi_2^4, \xi_2^5]$ in the parameter domain and $[\frac{1}{2}, 1] \times [\frac{1}{3}, \frac{2}{3}]$ in the sub-parameter domain. In Figure 4 the blending functions N_ℓ^A are shown for each direction in the sub-parameter domain ξ_ℓ. The part of the blending functions N_ℓ^A which has a support over element b with range $[\frac{1}{2}, 1] \times [\frac{1}{3}, \frac{2}{3}]$ in the sub-parameter domain ξ_ℓ – i. e. $N_{\ell\,b}^A$ – has been plotted with a solid black line. Expressing the blending functions $N_{\ell\,b}^A$ of anchor A with support over element b for each direction ξ_ℓ in terms of the Bernstein basis $B_{\ell\,b}^a$ of element b, gives for the Bézier extraction operator in each direction

$$N_{1\,b}^A = \begin{bmatrix} C_{1\,b}^{A\,1} & C_{1\,b}^{A\,2} & C_{1\,b}^{A\,3} \end{bmatrix} \begin{bmatrix} B_{1\,b}^1 \\ B_{1\,b}^2 \\ B_{1\,b}^3 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 1 & 0 \end{bmatrix} \begin{bmatrix} B_{1\,b}^1 \\ B_{1\,b}^2 \\ B_{1\,b}^3 \end{bmatrix} , \tag{10}$$

$$N_{2\,b}^A = \begin{bmatrix} C_{2\,b}^{A\,1} & C_{2\,b}^{A\,2} & C_{2\,b}^{A\,3} \end{bmatrix} \begin{bmatrix} B_{2\,b}^1 \\ B_{2\,b}^2 \\ B_{2\,b}^3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} B_{2\,b}^1 \\ B_{2\,b}^2 \\ B_{2\,b}^3 \end{bmatrix} . \tag{11}$$

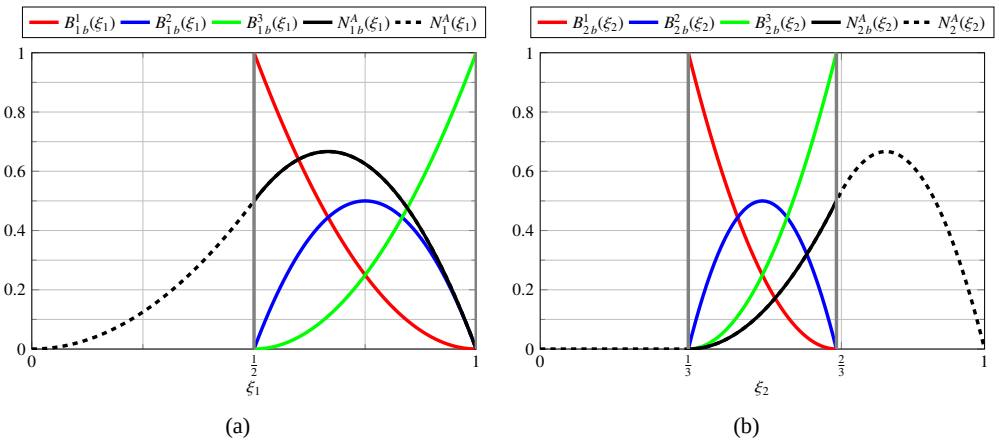


Figure 4. Illustration of the blending functions N_ℓ^A for anchor A and the Bernstein polynomials B_b^a for element b in Figure 2(a) over the sub-parameter domain in (a) the ξ_1 direction and (b) the ξ_2 direction.

Now, using Equation (9) and combining the unidirectional Bézier extraction operators defined in Equation (10) and Equation (11), the Bézier extraction operator $\underline{\boldsymbol{C}}_b^A$ for the anchor A with support over element b, see Figure 2(a), reads

$$\underline{\boldsymbol{C}}_b^A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{2} & 0 \end{bmatrix}^T . \tag{12}$$

If, for an anchor i, this procedure is applied to all elements E, then we end up with the Bézier extraction operator for anchor i

$$\underline{\boldsymbol{C}}^i = \begin{bmatrix} \underline{\boldsymbol{C}}_1^i \\ \vdots \\ \underline{\boldsymbol{C}}_E^i \end{bmatrix} . \tag{13}$$

DOI: 10.1002/nme

The Bézier extraction operator for all n anchors is then given by

$$\underline{\underline{\mathbf{C}}} = \begin{bmatrix} \underline{\boldsymbol{C}}^{1\,T} \\ \vdots \\ \underline{\boldsymbol{C}}^{n\,T} \end{bmatrix}. \tag{14}$$

$\underline{\underline{\mathbf{C}}}$ is called global Bézier extraction operator. Hence, the vector with all n blending functions

$$\underline{\boldsymbol{N}}(\underline{\boldsymbol{\xi}}) = \begin{bmatrix} N^1(\underline{\boldsymbol{\xi}}) \\ \vdots \\ N^n(\underline{\boldsymbol{\xi}}) \end{bmatrix} \tag{15}$$

can be written as

$$\underline{\boldsymbol{N}}(\underline{\boldsymbol{\xi}}) = \underline{\underline{\mathbf{C}}}\,\underline{\boldsymbol{B}}(\underline{\boldsymbol{\xi}}) \tag{16}$$

where $\underline{\boldsymbol{B}}$ is the vector which contains the elemental Bernstein polynomials $\underline{\boldsymbol{B}}_e$

$$\underline{\boldsymbol{B}}(\underline{\boldsymbol{\xi}}) = \begin{bmatrix} \underline{\boldsymbol{B}}_1(\underline{\boldsymbol{\xi}}) \\ \vdots \\ \underline{\boldsymbol{B}}_E(\underline{\boldsymbol{\xi}}) \end{bmatrix}. \tag{17}$$

A single blending function N^i can be expressed as

$$N^i(\underline{\boldsymbol{\xi}}) = \underline{\boldsymbol{C}}^{i\,T}\,\underline{\boldsymbol{B}}(\underline{\boldsymbol{\xi}}). \tag{18}$$

The blending functions $\underline{\boldsymbol{N}}_e$ with support in element e are determined by

$$\underline{\boldsymbol{N}}_e(\underline{\boldsymbol{\xi}}) = \underline{\underline{\mathbf{C}}}_e\,\underline{\boldsymbol{B}}_e(\underline{\boldsymbol{\xi}}) \tag{19}$$

with the elemental Bézier extraction operator $\underline{\underline{\mathbf{C}}}_e$.

4. CLASSIFICATION OF T-SPLINES

In this section T-spline meshes are classified according to the linear dependencies exhibited by their blending functions. The partition of unity property is also investigated. The classification methods in this section can be applied to T-spline meshes of arbitrary degree, T-spline meshes with extraordinary points and three-dimensional T-spline meshes: the only requirement is the Bézier extraction operator.

4.1. Classification of T-splines according to the type of linear dependence

In the following, the Bézier extraction operator is used to gather meshes into three categories based on the type of linear dependence of their blending functions:

- globally linearly independent,
- locally linearly independent with a non-square matrix $\underline{\underline{\mathbf{C}}}_e$,
- locally linearly independent with a square matrix $\underline{\underline{\mathbf{C}}}_e$.

4.1.1. Global linear independence

A T-spline mesh with n anchors has globally linearly independent blending functions if and only if the solution for

$$\sum_{i=1}^{n} \alpha^i N^i(\underline{\boldsymbol{\xi}}) = 0 \tag{20}$$

DOI: 10.1002/nme

is $\alpha^i = 0$ for $i = 1 \dots n$. We recall that each blending function N^i of anchor i can be expressed using the Bézier extraction operator. Substituting Equation (18) into Equation (20) leads to

$$\sum_{i=1}^{n} \alpha^i \underline{\boldsymbol{C}}^{i^T} \underline{\boldsymbol{B}}(\underline{\boldsymbol{\xi}}) = 0. \tag{21}$$

Since the Bernstein polynomials in $\underline{\boldsymbol{B}}$ are linearly independent, we can replace Equation (21) by

$$\sum_{i=1}^{n} \alpha^i \underline{\boldsymbol{C}}^{i^T} = \underline{\mathbf{0}}^T \tag{22}$$

which is equivalent to

$$\begin{bmatrix} \underline{\boldsymbol{C}}^1 & \dots & \underline{\boldsymbol{C}}^n \end{bmatrix} \begin{bmatrix} \alpha^1 \\ \vdots \\ \alpha^n \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{23}$$

Thus, using Equation (14), Equation (23) can be rewritten as

$$\underline{\underline{\mathbf{C}}}^T \underline{\boldsymbol{\alpha}} = \underline{\mathbf{0}}. \tag{24}$$

Since a T-spline mesh has globally linearly independent blending functions N^i when the only solution for Equation (24) is $\alpha^i = 0$ for $i = 1 \dots n$, it follows directly from rank inspection of the global Bézier extraction operator $\underline{\underline{\mathbf{C}}}$ in Equation (24) whether the blending functions N^i of a T-spline mesh are globally linearly independent. If the global Bézier extraction operator $\underline{\underline{\mathbf{C}}}$ has full rank, then the rank of $\underline{\underline{\mathbf{C}}}$ is equal to the number of anchors n and consequently, the blending functions N^i are globally linearly independent. In sum, the condition for global linear independence is

$$\operatorname{rank}(\underline{\underline{\mathbf{C}}}) = n. \tag{25}$$

Note, that the size of the global Bézier extraction operator is

$$\operatorname{size}(\underline{\underline{\mathbf{C}}}) = n \times \left(E \times \prod_{\ell=1}^{d} p_\ell + 1 \right). \tag{26}$$

If a T-spline mesh is globally linearly dependent, the dependencies between anchors can be detected by transforming Equation (24) into a row echelon form by Gaussian elimination, Figure 5.

A T-spline mesh with globally linearly dependent blending functions cannot be used for analysis since in a finite element context, this results in a system of equations that cannot be solved.

4.1.2. Local linear independence

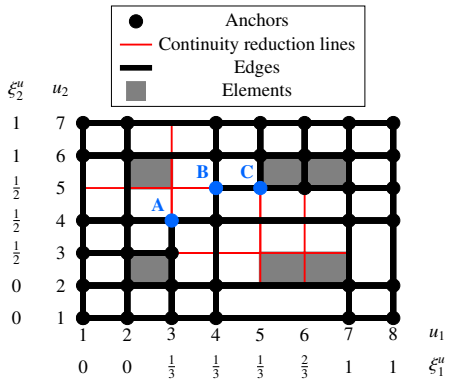
Repeating the procedure of the previous section at the elemental level, the condition for local linear independence is

$$\operatorname{rank}(\underline{\underline{\mathbf{C}}}_e) = n_e \quad \text{for } e = 1 \dots E \tag{27}$$

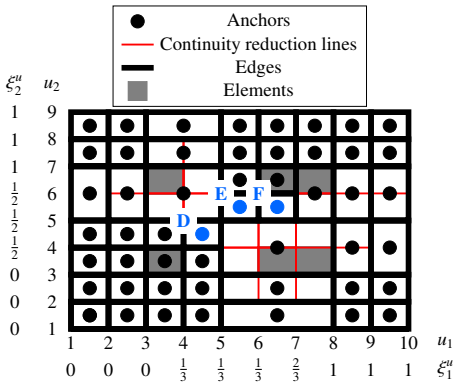
with the elemental Bézier extraction operator $\underline{\underline{\mathbf{C}}}_e$ and the number of anchors n_e with support in element e. When Equation (27) holds, the size of $\underline{\underline{\mathbf{C}}}_e$ is

$$\operatorname{size}(\underline{\underline{\mathbf{C}}}_e) = n_e \times \left(\prod_{\ell=1}^{d} p_\ell + 1 \right). \tag{28}$$

DOI: 10.1002/nme



(a) Globally linearly dependent cubic T-spline mesh [8], transforming Equation (24) into row echelon form yields the following linear dependencies between the anchors A, B and C: $-3N^A(\underline{\boldsymbol{\xi}}) + 3N^B(\underline{\boldsymbol{\xi}}) + N^C(\underline{\boldsymbol{\xi}}) = 0$.



(b) Globally linearly dependent quartic T-spline mesh, transforming Equation (24) into row echelon form yields the following linear dependencies between the anchors D, E and F: $-3N^D(\underline{\boldsymbol{\xi}}) + 2N^E(\underline{\boldsymbol{\xi}}) + N^F(\underline{\boldsymbol{\xi}}) = 0$.

Figure 5. Globally linearly dependent T-spline meshes of cubic and of quartic polynomial degree.

4.1.3. Local linear independence with a square matrix $\underline{\underline{\mathbf{C}}}_e$

A subset of locally linearly independent T-spline meshes (i. e. when Equation (27) holds) can be defined when the following additional property is valid for each element e

$$\operatorname{rank}(\underline{\underline{\mathbf{C}}}_e) = \prod_{\ell=1}^{d} p_\ell + 1 = n_e \quad \text{for } e = 1 \dots E. \tag{29}$$

When Equation (29) holds, the size of $\underline{\underline{\mathbf{C}}}_e$ is

$$\operatorname{size}(\underline{\underline{\mathbf{C}}}_e) = \left(\prod_{\ell=1}^{d} p_\ell + 1 \right) \times \left(\prod_{\ell=1}^{d} p_\ell + 1 \right). \tag{30}$$

Note, that Equation (29) implies Equation (27). Also, Equation (27) implies Equation (25) – local linear independence inherently results in global linear independence.

If a T-spline mesh is locally linearly dependent, then the non-zero coefficients $\underline{\boldsymbol{\alpha}}_e$ are obtained analogously to the global case by transforming

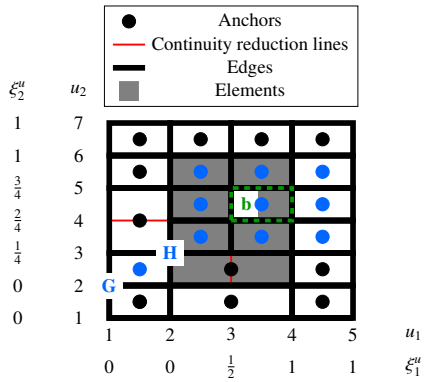
$$\underline{\underline{\mathbf{C}}}_e^T \underline{\boldsymbol{\alpha}}_e = \underline{\mathbf{0}} \tag{31}$$

into row echelon form. In Figure 6 examples are given for a locally linearly dependent T-spline mesh, a T-spline mesh for which Equation (27) holds and a T-spline mesh for which Equation (29) holds.

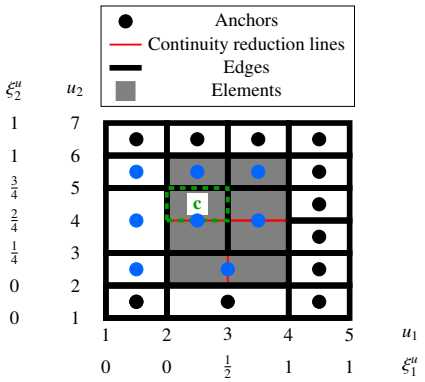
4.2. Partition of unity property for T-splines

In this section we address the partition of unity property of the blending functions (N^i) and of the rational blending functions (R^i), respectively. It will be elaborated how T-spline meshes can be classified as standard, semi-standard and non-standard using the Bézier extraction operator. We will also show that an affine transformation only exists when the partition of unity property is satisfied, with the ensuing consequence for the satisfaction of the patch test.

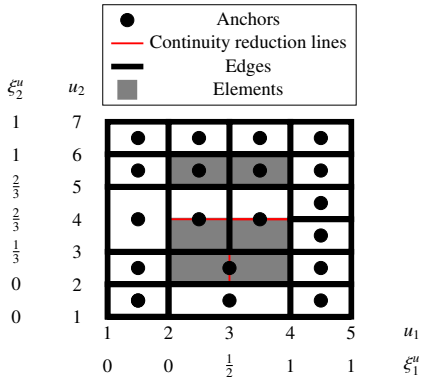
DOI: 10.1002/nme



(a) Locally linearly dependent T-spline mesh, ten anchors (blue) have a support in element b (dashed green); transforming Equation (31) into row echelon form yields the dependencies in element b between anchors G and H: $6N^G(\underline{\boldsymbol{\xi}}) - N^H(\underline{\boldsymbol{\xi}}) = 0$.



(b) Locally linearly independent T-spline mesh; $\mathrm{rank}(\underline{\underline{\mathbf{C}}}_e) = n_e$ for element c; in element c (dashed green) are only eight anchors (blue) with a support and therefore $\underline{\underline{\mathbf{C}}}_e$ is not a square matrix for element c.



(c) Locally linearly independent T-spline mesh; $\mathrm{rank}(\underline{\underline{\mathbf{C}}}_e) = \prod_{\ell=1}^{d} p_\ell + 1$ for $e = 1 \dots E$.

Figure 6. Local dependencies in a quadratic T-spline mesh.

4.2.1. Partition of unity property of the rational blending functions R^i

The multivariate rational T-spline blending function for an anchor i can be constructed as

$$R^i(\underline{\boldsymbol{\xi}}) = \frac{w^i N^i(\underline{\boldsymbol{\xi}})}{\sum_{j=1}^{n} w^j N^j(\underline{\boldsymbol{\xi}})} \tag{32}$$

with the weight w^i associated to anchor i. Note that, in view of Equation (32), the rational blending functions R^i always form a partition of unity (all R^i sum to one).

4.2.2. Partition of unity property of the blending functions N^i

In [3] T-spline meshes have been classified according to the partition of unity property of the blending functions N^i,

$$\sum_{i=1}^{n} \beta^i N^i(\underline{\boldsymbol{\xi}}) = 1, \tag{33}$$

into

- Standard T-spline meshes: all $\beta^i = 1$,

DOI: 10.1002/nme

- Semi-standard T-spline meshes: some $\beta^i \neq 1$,
- Non-standard T-spline meshes: no solution for β^i.

We note, that only for standard T-spline meshes the blending functions N^i *and* the rational blending functions R^i satisfy the partition of unity property.

4.2.3. Partition of unity property of the blending functions N^i using the Bézier extraction operator
We will now show how the global Bézier extraction operator can be used to determine the partition of unity property of the blending functions N^i. Rewriting Equation (33) using Equation (18) yields

$$\sum_{i=1}^{n} \beta^i \underline{\boldsymbol{C}}^{i^T} \underline{\boldsymbol{B}}(\underline{\boldsymbol{\xi}}) = 1. \tag{34}$$

Substituting Equations (13) and (17) into Equation (34) and elaboration gives

$$\underbrace{\left(\beta^1 \underline{\boldsymbol{C}}_1^{1^T} + \ldots + \beta^n \underline{\boldsymbol{C}}_1^{n^T}\right)}_{\underline{\boldsymbol{\gamma}}_1^T} \underline{\boldsymbol{B}}_1(\underline{\boldsymbol{\xi}}) + \ldots + \underbrace{\left(\beta^1 \underline{\boldsymbol{C}}_E^{1^T} + \ldots + \beta^n \underline{\boldsymbol{C}}_E^{n^T}\right)}_{\underline{\boldsymbol{\gamma}}_E^T} \underline{\boldsymbol{B}}_E(\underline{\boldsymbol{\xi}}) = 1. \tag{35}$$

The Bernstein polynomials $\underline{\boldsymbol{B}}_e$ in Equation (35) form a partition of unity if and only if $\underline{\boldsymbol{\gamma}}_e = \underline{\mathbf{1}}$ for each element e. This statement can be expressed in a vector-matrix format as

$$\begin{bmatrix} \beta^1 \underline{\boldsymbol{C}}_1^1 + \ldots + \beta^n \underline{\boldsymbol{C}}_1^n \\ \vdots \\ \beta^1 \underline{\boldsymbol{C}}_E^1 + \ldots + \beta^n \underline{\boldsymbol{C}}_E^n \end{bmatrix} = \begin{bmatrix} \underline{\mathbf{1}} \\ \vdots \\ \underline{\mathbf{1}} \end{bmatrix} \tag{36}$$

which is equivalent to

$$\begin{bmatrix} \underline{\boldsymbol{C}}^1 & \ldots & \underline{\boldsymbol{C}}^n \end{bmatrix} \begin{bmatrix} \beta^1 \\ \vdots \\ \beta^n \end{bmatrix} = \underline{\mathbf{1}}. \tag{37}$$

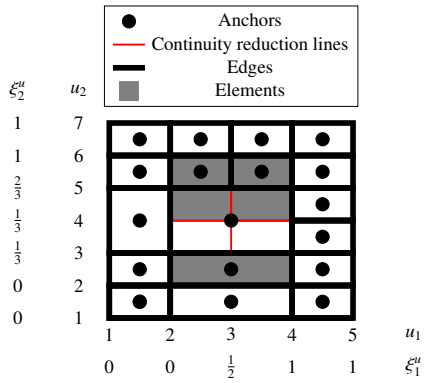
With the global Bézier extraction operator $\underline{\underline{\mathbf{C}}}$ in Equation (14) we obtain

$$\underline{\underline{\mathbf{C}}}^T \underline{\boldsymbol{\beta}} = \underline{\mathbf{1}}. \tag{38}$$

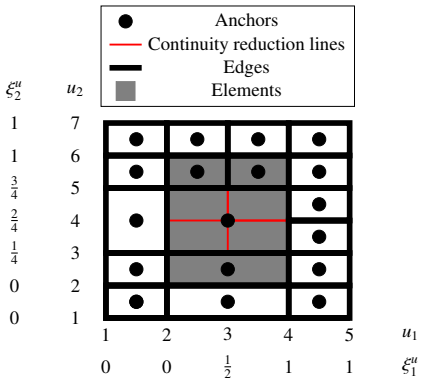
The row echelon form of Equation (38) then provides the means to assess whether a T-spline mesh is standard, semi-standard or non-standard.

Figures 7 and 8 show a set of T-spline meshes of quadratic and cubic degree and their classification for linear dependence and the partition of unity property according to the previous definitions using the Bézier extraction operator. As can be observed from Figures 7 and 8, changing the knot intervals gives a different classification for the T-spline mesh – (a), (c), (e) are standard T-spline meshes whereas (b), (d) and (f) are non-standard or semi-standard T-spline meshes.

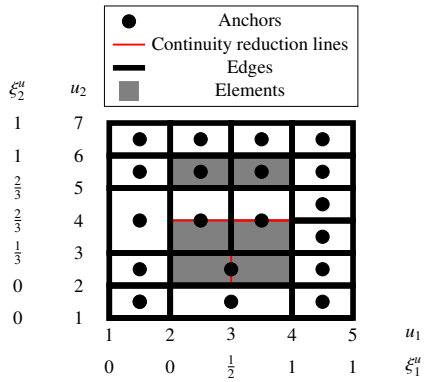
DOI: 10.1002/nme



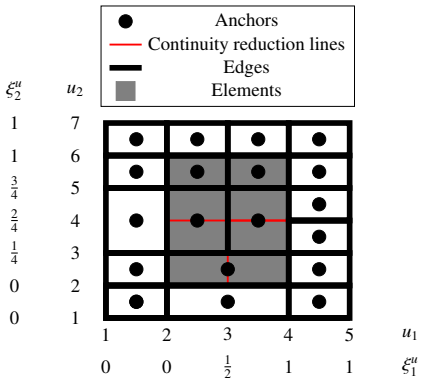
(a) Standard, $\mathrm{rank}(\underline{\underline{\mathbf{C}}}_e) = \prod_{\ell=1}^{d} p_\ell + 1$ for $e = 1 \dots E$.



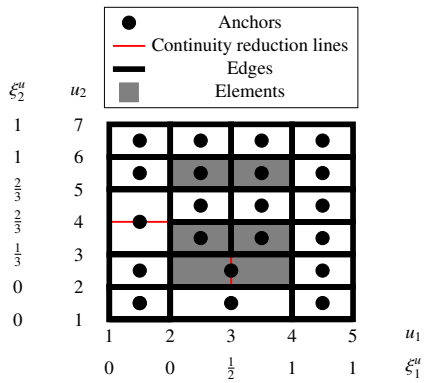
(b) Non-standard, $\mathrm{rank}(\underline{\underline{\mathbf{C}}}) = n$.



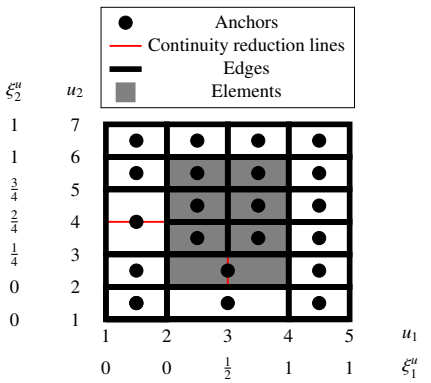
(c) Standard, $\mathrm{rank}(\underline{\underline{\mathbf{C}}}_e) = \prod_{\ell=1}^{d} p_\ell + 1$ for $e = 1 \dots E$.



(d) Non-standard, $\mathrm{rank}(\underline{\underline{\mathbf{C}}}_e) = n_e$ for $e = 1 \dots E$.



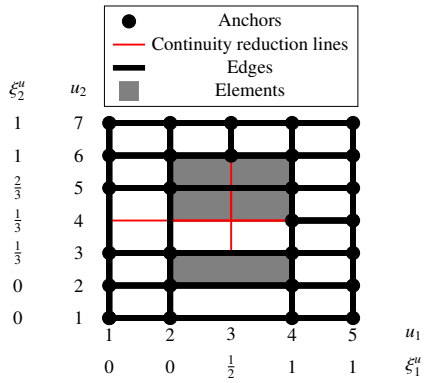
(e) Standard, $\mathrm{rank}(\underline{\underline{\mathbf{C}}}_e) = \prod_{\ell=1}^{d} p_\ell + 1$ for $e = 1 \dots E$.



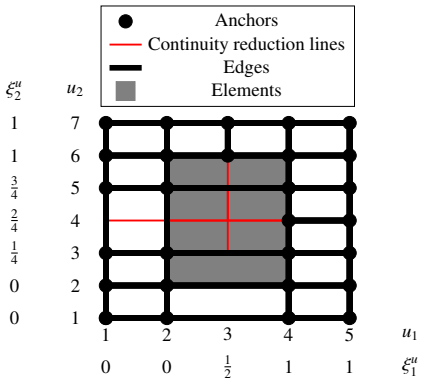
(f) Semi-standard, $\mathrm{rank}(\underline{\underline{\mathbf{C}}}) = n$.

Figure 7. Classification of quadratic T-spline meshes according to the level of linear independence and the partition of unity property. (a), (c) and (e) are standard T-spline meshes, changing the knot intervals results in non-standard or semi-standard T-spline meshes.

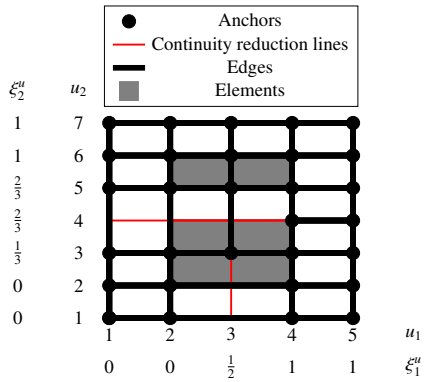
DOI: 10.1002/nme



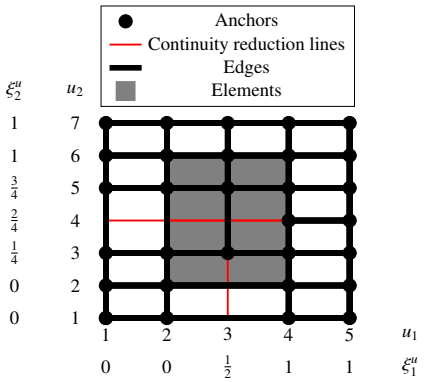
(a) Standard, $\mathrm{rank}(\underline{\underline{\mathbf{C}}}_e) = \prod_{\ell=1}^{d} p_\ell + 1$ for $e = 1 \dots E$.



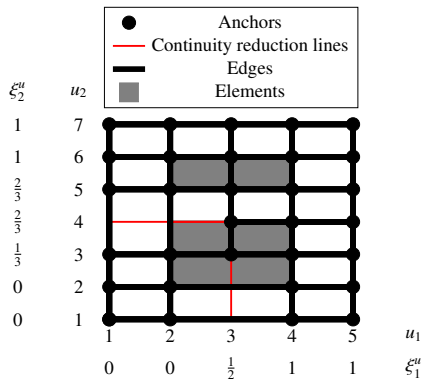
(b) Non-standard, $\mathrm{rank}(\underline{\underline{\mathbf{C}}}) = n$.



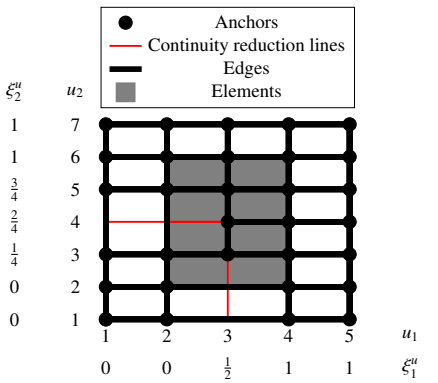
(c) Standard, $\mathrm{rank}(\underline{\underline{\mathbf{C}}}_e) = \prod_{\ell=1}^{d} p_\ell + 1$ for $e = 1 \dots E$.



(d) Non-standard, $\mathrm{rank}(\underline{\underline{\mathbf{C}}}_e) = n_e$ for $e = 1 \dots E$.



(e) Standard, $\mathrm{rank}(\underline{\underline{\mathbf{C}}}_e) = \prod_{\ell=1}^{d} p_\ell + 1$ for $e = 1 \dots E$.



(f) Semi-standard, $\mathrm{rank}(\underline{\underline{\mathbf{C}}}) = n$.

Figure 8. Classification of cubic T-spline meshes according to the level of linear independence and the partition of unity property. (a), (c) and (e) are standard T-spline meshes, changing the knot intervals results in non-standard or semi-standard T-spline meshes.

DOI: 10.1002/nme

4.2.4. Affine transformation requires partition of unity

Any T-spline surface $\underline{\boldsymbol{T}}$ in the physical domain (x_ℓ) can be expressed by the mapping from the sub-parameter (ξ_ℓ) domain as follows

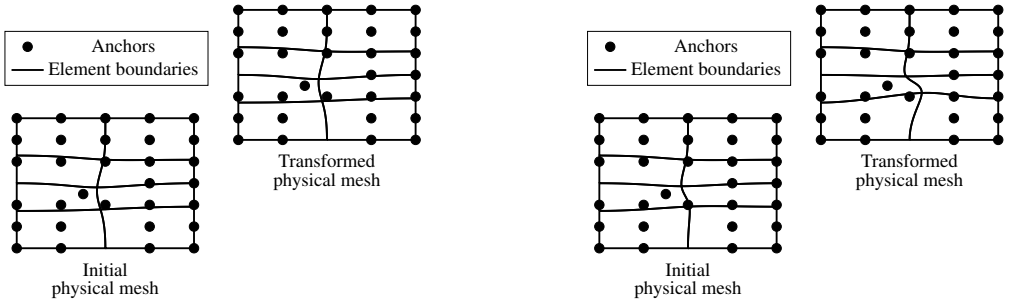
$$\underline{\boldsymbol{T}}(\underline{\boldsymbol{\xi}}) = \sum_{i=1}^{n} R^i(\underline{\boldsymbol{\xi}})\underline{\boldsymbol{P}}^i \tag{39}$$

where R^i are the rational blending functions and $\underline{\boldsymbol{P}}^i = (x_1^i, x_2^i)$ are the control points associated to anchor i. Applying a transformation to the control points $\underline{\boldsymbol{P}}$ of the form

$$\underline{\boldsymbol{P}}_{\mathrm{T}} = \underline{\underline{\mathbf{A}}}\,\underline{\boldsymbol{P}} + \underline{\boldsymbol{b}}, \tag{40}$$

with the control points $\underline{\boldsymbol{P}}_{\mathrm{T}}$ after transformation, results in an affine transformation since the rational blending functions R^i in Equation (39) form a partition of unity.

However, when the T-spline in Equation (39) would have been defined with the blending functions N^i instead of the rational blending functions R^i, then an affine transformation is only obtained for standard T-spline meshes since semi-standard and non-standard T-spline meshes do not have the partition of unity property for the blending functions N^i, see also Figure 9 with a rigid body motion applied to the control points of the anchors.



(a) Rational blending functions R^i. (b) Blending functions N^i.

Figure 9. Applying a rigid body motion to the control points of the anchors results in an affine transformation when the partition of unity property is fulfilled. (a) An affine transformation for the semi-standard T-spline mesh in Figure 8(f) is obtained for the rational blending functions R^i; (b) using the blending functions N^i instead of R^i in Equation (39) gives no affine transformation – the element boundaries are different – since the N^i do not form a partition of unity for semi-standard meshes.

The patch test is always satisfied when an affine transformation is possible, i. e. for the rational blending functions R^i, and for the blending functions N^i of a standard T-spline mesh.

4.3. Standard T-splines are locally linearly independent with a square matrix $\underline{\underline{\mathbf{C}}}_e$

We will show next that standard T-spline meshes are always locally linearly independent and that the elemental Bézier operators $\underline{\underline{\mathbf{C}}}_e$ are always a square matrix. We start with the global partition of unity property of standard T-spline meshes,

$$\sum_{i=1}^{n} \beta^i N^i(\underline{\boldsymbol{\xi}}) = 1 \quad \text{with } \beta^i = 1. \tag{41}$$

The global partition of unity property for standard T-spline meshes in Equation (41) implies the local partition of unity property for each element e

$$\sum_{i=1}^{n_e} \beta_e^i N^i(\underline{\boldsymbol{\xi}}) = 1 \quad \text{with } \beta_e^i = 1. \tag{42}$$

DOI: 10.1002/nme

Now we add to Equation (42) the expression

$$\sum_{i=1}^{n_e} \alpha_e^i N^i(\underline{\boldsymbol{\xi}}) = 0 \tag{43}$$

which results in

$$\sum_{i=1}^{n_e} (\beta_e^i + \alpha_e^i) N^i(\underline{\boldsymbol{\xi}}) = 1 \quad \text{with } \beta_e^i = 1. \tag{44}$$

If there existed an anchor i with $\alpha_e^i \neq 0$ in Equation (44), the T-spline mesh would not be a standard T-spline mesh (see also the proof in [16] for the global case). Therefore, the only solution is $\alpha_e^i = 0$ for $i = 1 \dots n_e$ in Equation (44) which means that we have for each element e in Equation (43)

$$\sum_{i=1}^{n_e} \alpha_e^i N^i(\underline{\boldsymbol{\xi}}) = 0 \quad \text{with } \alpha_e^i = 0. \tag{45}$$

Hence, the global partition of unity property implies local linear independence of the T-spline mesh.

However, we do not know yet whether $\underline{\underline{\mathbf{C}}}_e$ is a square matrix or not. To further pursue this issue, we write the result $\beta_e^i = 1$ in Equation (42) as follows

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} \beta_e^1 \\ \vdots \\ \beta_e^{n_e} \end{bmatrix} = \underline{\underline{\mathbf{I}}}_e \, \underline{\boldsymbol{\beta}}_e = \underline{\mathbf{1}} \tag{46}$$

where $\underline{\underline{\mathbf{I}}}_e$ is the unity matrix

$$\underline{\underline{\mathbf{I}}}_e = \mathrm{diag}(1, 1, \dots, 1). \tag{47}$$

It is important to note that the size of the unity matrix $\underline{\underline{\mathbf{I}}}_e$ is

$$\mathrm{size}(\underline{\underline{\mathbf{I}}}_e) = \left(\prod_{\ell=1}^{d} p_\ell + 1 \right) \times \left(\prod_{\ell=1}^{d} p_\ell + 1 \right) \tag{48}$$

and therefore, $n_e = \prod_{\ell=1}^{d} p_\ell + 1$. By writing Equation (38) for element e in an elemental form

$$\underline{\underline{\mathbf{C}}}_e^T \underline{\boldsymbol{\beta}}_e = \underline{\mathbf{1}}, \tag{49}$$

we can draw the conclusion that Equation (46) is the row echelon form of Equation (49). Hence, we can infer that $\underline{\underline{\mathbf{C}}}_e$ has the same size as $\underline{\underline{\mathbf{I}}}_e$ and that $\underline{\underline{\mathbf{C}}}_e$ is also a square matrix,

$$\mathrm{size}(\underline{\underline{\mathbf{C}}}_e) = \mathrm{size}(\underline{\underline{\mathbf{I}}}) = \left(\prod_{\ell=1}^{d} p_\ell + 1 \right) \times \left(\prod_{\ell=1}^{d} p_\ell + 1 \right). \tag{50}$$

We recall that the global partition of unity property results in locally linearly independent blending functions. With Equation (50) this leads to the conclusion that we have the case in Equation (29) since $\underline{\underline{\mathbf{C}}}_e$ is a square matrix. In sum, all standard T-splines have the following property

$$\mathrm{rank}(\underline{\underline{\mathbf{C}}}_e) = \prod_{\ell=1}^{d} p_\ell + 1 = n_e \quad \text{for } e = 1 \dots E. \tag{51}$$

Equation (51) is a necessary condition for standard T-spline meshes: if Equation (51) is not fulfilled, then the T-spline mesh cannot be a standard T-spline mesh. However, even when Equation (51) is fulfilled, the T-spline mesh can still be semi-standard or non-standard. In order to determine standard T-spline meshes, it is necessary *and* sufficient to show that Equation (38) yields $\underline{\boldsymbol{\beta}} = \underline{\mathbf{1}}$.

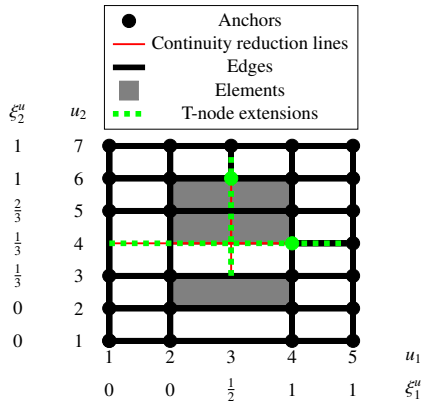
DOI: 10.1002/nme

4.4. Analysis-suitable T-splines

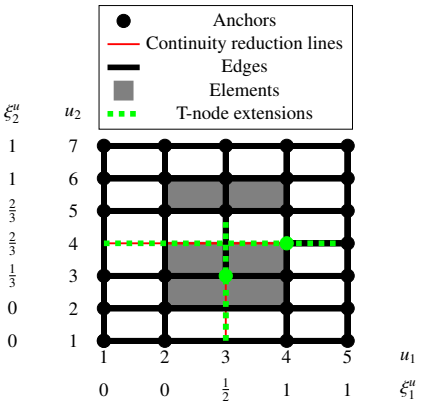
Analysis-suitable T-splines have been defined in [9]. In order to detect them, the extended T-spline mesh was introduced, and a mesh was deemed analysis-suitable when there are no two orthogonal T-node extensions which intersect in the extended T-spline mesh. This definition holds for any knot interval and is of topological nature; it allows to distinguish between analysis-suitable and non-analysis-suitable T-splines.

The new approach in this paper which is based on the Bézier extraction operator is an algebraic viewpoint and allows a classification of T-splines into standard, semi-standard and non-standard with Equation (38).

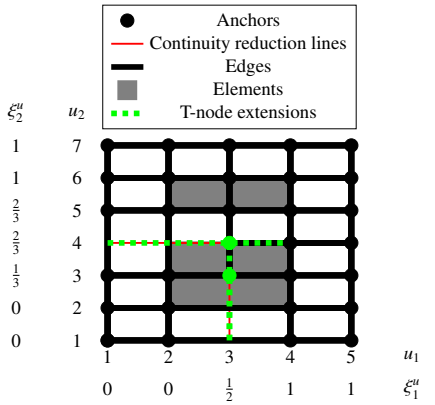
Figure 10 reveals that a standard T-spline is not necessarily an analysis-suitable T-spline. In Figure 10, T-node extensions intersect in the extended T-spline mesh and the T-spline meshes are therefore non-analysis-suitable. From Figure 8(a), 8(c) and 8(e) we know that these T-spline meshes are standard.



(a) T-node extensions for the T-spline mesh in Figure 8(a).



(b) T-node extensions for the T-spline mesh in Figure 8(c).



(c) T-node extensions for the T-spline mesh in Figure 8(e).

Figure 10. Extended T-spline meshes for Figure 8(a), 8(c) and 8(e); these standard T-spline meshes are non-analysis-suitable according to [9] since T-node extensions intersect in the extended T-spline mesh.

DOI: 10.1002/nme

5. LOCAL REFINEMENT OF STANDARD, SEMI-STANDARD AND NON-STANDARD T-SPLINES BY ADDING ANCHORS

In this section we show how standard, semi-standard and non-standard T-spline meshes of even and odd polynomial degree can be refined locally by adding anchors using information from the Bézier extraction operator.

A requirement for the refinement algorithm is that the initial and the refined T-spline mesh are nested – this condition will be defined in the following section, together with a method to fulfil it using the Bézier extraction operator. We also show how the location of the control points in the refined T-spline mesh can be obtained. Afterwards, we explain the algorithm for the local refinement of T-splines and give some examples. In the examples we first focus on refining standard T-spline meshes (Section 5.4, Appendix A) followed by an example to show that also non-standard meshes can be refined locally by adding anchors (Appendix B).

5.1. Computation of the refinement matrix and nesting behaviour

A refinement matrix $\underline{\underline{\mathbf{M}}}$ of size $n \times n_R$ gives the relation between the blending functions $\underline{\boldsymbol{N}}_R$ of a refined mesh with n_R anchors and the blending functions $\underline{\boldsymbol{N}}$ of an initial mesh which has n anchors

$$\underline{\boldsymbol{N}}(\underline{\boldsymbol{\xi}}) = \underline{\underline{\mathbf{M}}}\ \underline{\boldsymbol{N}}_R(\underline{\boldsymbol{\xi}}). \tag{52}$$

Expressing the blending functions on both sides using the Bernstein polynomials, Equation (16), gives

$$\underline{\underline{\mathbf{C}}}\ \underline{\boldsymbol{B}}_R(\underline{\boldsymbol{\xi}}) = \underline{\underline{\mathbf{M}}}\ \underline{\underline{\mathbf{C}}}_R \underline{\boldsymbol{B}}_R(\underline{\boldsymbol{\xi}}), \tag{53}$$

while the blending functions $\underline{\boldsymbol{N}}$ on the initial mesh must be defined in terms of the elements of the refined mesh with the Bernstein polynomials $\underline{\boldsymbol{B}}_R$. The linear independence of the Bernstein polynomials $\underline{\boldsymbol{B}}_R$ in Equation (53) results in

$$\underline{\underline{\mathbf{C}}} = \underline{\underline{\mathbf{M}}}\ \underline{\underline{\mathbf{C}}}_R. \tag{54}$$

The coefficients of a row of the refinement matrix $\underline{\underline{\mathbf{M}}}$ can be evaluated as follows. Expanding Equation (54) using Equation (14) yields

$$\begin{bmatrix} \underline{\boldsymbol{C}}^{1\,T} \\ \vdots \\ \underline{\boldsymbol{C}}^{n\,T} \end{bmatrix} = \begin{bmatrix} \underline{\boldsymbol{M}}^{1\,T} \\ \vdots \\ \underline{\boldsymbol{M}}^{n\,T} \end{bmatrix} \begin{bmatrix} \underline{\boldsymbol{C}}_R^{1\,T} \\ \vdots \\ \underline{\boldsymbol{C}}_R^{n_R\,T} \end{bmatrix}. \tag{55}$$

Applying the transpose to both sides results in

$$\begin{bmatrix} \underline{\boldsymbol{C}}^1 & \dots & \underline{\boldsymbol{C}}^n \end{bmatrix} = \begin{bmatrix} \underline{\boldsymbol{C}}_R^1 & \dots & \underline{\boldsymbol{C}}_R^{n_R} \end{bmatrix} \begin{bmatrix} \underline{\boldsymbol{M}}^1 & \dots & \underline{\boldsymbol{M}}^n \end{bmatrix} \tag{56}$$

which makes it possible to determine the rows $\underline{\boldsymbol{M}}^{i\,T}$ for $i = 1 \dots n$ of the refinement matrix $\underline{\underline{\mathbf{M}}}$ by transforming the systems

$$\underline{\boldsymbol{C}}^i = \underline{\underline{\mathbf{C}}}_R{}^T\, \underline{\boldsymbol{M}}^i \quad \text{for } i = 1 \dots n \tag{57}$$

into a row echelon form. In the case that there is no solution for the $\underline{\boldsymbol{M}}^i$ for anchor i in Equation (57), the initial and the refined T-spline mesh are not nested, which means that it is not possible to represent *all* blending functions $\underline{\boldsymbol{N}}$ of the initial T-spline mesh as a linear combination of the blending functions $\underline{\boldsymbol{N}}_R$ of the refined T-spline mesh. One can resolve this as will be explained in Section 5.4 (quadratic case, $p_\ell = 2$) and Appendix A (cubic case, $p_\ell = 3$).

It is interesting to note that when nestedness is ensured and the initial mesh is standard, the refined T-spline mesh can only be a standard or semi-standard T-spline mesh: knowing that the initial T-spline mesh is standard and satisfies the partition of unity property in Equation (41) (all

Prepared using nmeauth.cls
DOI: 10.1002/nme

$\underline{\boldsymbol{\beta}} = \underline{\mathbf{1}}$) and using the rows $\underline{\boldsymbol{M}}^{i^T}$ of the refinement matrix $\underline{\underline{\mathbf{M}}}$ from Equation (57) results in

$$1 = \sum_{i=1}^{n} \beta^i N^i(\underline{\boldsymbol{\xi}}) = \sum_{i=1}^{n} \beta^i \underline{\boldsymbol{M}}^{i^T} \underline{\boldsymbol{N}}_R(\underline{\boldsymbol{\xi}}) = \sum_{j=1}^{n_R} \beta_R^j N_R^j(\underline{\boldsymbol{\xi}}) \tag{58}$$

where the coefficients $\underline{\boldsymbol{\beta}}_R$ are given by

$$\underline{\boldsymbol{\beta}}_R = \underline{\underline{\mathbf{M}}}^T \underline{\boldsymbol{\beta}}. \tag{59}$$

From Equation (59) it can be concluded that there always exists a solution for the coefficients $\underline{\boldsymbol{\beta}}_R$ when nestedness is ensured ($\underline{\underline{\mathbf{M}}}$ exists) and therefore the refined T-spline mesh can only be a standard or semi-standard T-spline mesh when the initial mesh is standard.

5.2. Determination of the coordinates for the anchors in the refined T-spline mesh

In this section we assume that the initial and the refined T-spline mesh are nested. We show how the coordinates and weights of the anchors in a refined T-spline mesh can be determined. The weighted (polynomial) curve of Equation (39) is given by [20]

$$\underline{\boldsymbol{T}}_w(\underline{\boldsymbol{\xi}}) = \sum_{i=1}^{n} N^i(\underline{\boldsymbol{\xi}}) \underline{\boldsymbol{P}}_w^i \tag{60}$$

with the weighted control points

$$\underline{\boldsymbol{P}}_w^i = (w^i x_1^i, w^i x_2^i, w^i). \tag{61}$$

We require that the refined and the initial weighted curves – $\underline{\boldsymbol{T}}_{w\,R}$ and $\underline{\boldsymbol{T}}_w$, respectively – represent the same geometry

$$\underline{\boldsymbol{T}}_{w\,R}(\underline{\boldsymbol{\xi}}) = \underline{\boldsymbol{T}}_w(\underline{\boldsymbol{\xi}}), \tag{62}$$

and insert Equation (60) into the left- and right-hand side of Equation (62) to obtain

$$\sum_{j=1}^{n_R} N_R^j(\underline{\boldsymbol{\xi}}) \underline{\boldsymbol{P}}_{w\,R}^j = \sum_{i=1}^{n} N^i(\underline{\boldsymbol{\xi}}) \underline{\boldsymbol{P}}_w^i. \tag{63}$$

Using the Bézier extraction operator subsequently gives

$$\sum_{j=1}^{n_R} \underline{\boldsymbol{C}}_R^{j\,T} \underline{\boldsymbol{B}}_R(\underline{\boldsymbol{\xi}}) \underline{\boldsymbol{P}}_{w\,R}^j = \sum_{i=1}^{n} \underline{\boldsymbol{C}}^{i^T} \underline{\boldsymbol{B}}_R(\underline{\boldsymbol{\xi}}) \underline{\boldsymbol{P}}_w^i \tag{64}$$

or, since the Bernstein polynomials $\underline{\boldsymbol{B}}_R$ are linearly independent

$$\sum_{j=1}^{n_R} \underline{\boldsymbol{C}}_R^{j\,T} \underline{\boldsymbol{P}}_{w\,R}^j = \sum_{i=1}^{n} \underline{\boldsymbol{C}}^{i^T} \underline{\boldsymbol{P}}_w^i. \tag{65}$$

Elaborating Equation (65) yields

$$\begin{bmatrix} \underline{\boldsymbol{C}}_R^1 & \cdots & \underline{\boldsymbol{C}}_R^{n_R} \end{bmatrix} \begin{bmatrix} \underline{\boldsymbol{P}}_{w\,R}^1 \\ \vdots \\ \underline{\boldsymbol{P}}_{w\,R}^{n_R} \end{bmatrix} = \begin{bmatrix} \underline{\boldsymbol{C}}^1 & \cdots & \underline{\boldsymbol{C}}^n \end{bmatrix} \begin{bmatrix} \underline{\boldsymbol{P}}_w^1 \\ \vdots \\ \underline{\boldsymbol{P}}_w^n \end{bmatrix} \tag{66}$$

or, in the global form

$$\underline{\underline{\mathbf{C}}}_R^{\,T} \underline{\boldsymbol{P}}_{w\,R} = \underline{\underline{\mathbf{C}}}^T \underline{\boldsymbol{P}}_w, \tag{67}$$

so that with Equation (54), we obtain

$$\underline{\underline{\mathbf{C}}}_R^{\,T} \underline{\boldsymbol{P}}_{w\,R} = \underline{\underline{\mathbf{C}}}_R^{\,T} \underline{\underline{\mathbf{M}}}^T \underline{\boldsymbol{P}}_w. \tag{68}$$

Hence, the weighted control points $\underline{\boldsymbol{P}}_{w\,R}$ for the refined mesh follow from

$$\underline{\boldsymbol{P}}_{w\,R} = \underline{\underline{\mathbf{M}}}^T \underline{\boldsymbol{P}}_w. \tag{69}$$

DOI: 10.1002/nme

5.3. The algorithm for local refinement of standard T-splines

In our local refinement algorithm (see also Algorithm 1) for standard T-splines we proceed as follows: after inserting new anchors into the T-spline mesh (refining), we check whether the necessary condition for standard T-spline meshes, Equation (51), holds. If this is not the case, the mesh resulting from local refinement will not be standard either. Then, Equation (31) plays a key role: it allows us to determine whether $\underline{\underline{\mathbf{C}}}_e$ is a square matrix or not, but also, when presented in row echelon form, to detect and remove linear dependencies, leading to the necessary condition for standard T-spline meshes in Equation (51). Once Equation (51) is fulfilled, we evaluate each row of the refinement matrix $\underline{\underline{\mathbf{M}}}$ in Equation (57). Should the blending functions of some anchors of the initial mesh not be nested in the refined mesh, then we modify the mesh accordingly. Finally, when nestedness is satisfied, Equation (38) is assessed whether $\underline{\boldsymbol{\beta}} = \underline{\mathbf{1}}$ holds. If not, then we have a semi-standard mesh according to Equation (59) and anchors are added to the mesh within the support of anchors for which $\beta^i \neq 1$. Otherwise, the initial and the refined mesh are nested standard T-spline meshes.

```
// Start with a standard T-spline mesh
// Number of refinement steps: N
for i = 1 : N do
    RefinementSuccessful = 0;
    while RefinementSuccessful = 0 do
        // Check whether necessary condition for standard T-splines in Equation (51) holds:
        if C_e ≠ ∏_{ℓ=1}^{d} (p_ℓ + 1)   for e = 1 ... E   then
            // add additional anchors by inspecting the Bézier extraction operator in Equation (31):
            // (a) ensure that C_e is a square matrix
            // (b) remove linear dependencies
        else
            // Check with Equation (57) whether the initial and the refined mesh are nested:
            if Refinement matrix M cannot be computed then
                // add additional anchors by assessing the Bézier extraction operators of the initial and the refined
                mesh: localise, which anchors are not nested in Equation (57)
            else
                // Check whether T-spline mesh is standard by assessing Equation (38):
                if β ≠ 1 then
                    // mesh is semi-standard according to Equation (59)
                    // add anchors to the mesh within the support of the anchors i for which β^i ≠ 1
                else
                    // Compute the weighted control points P_{w R} of the refined mesh using Equation (69)
                    RefinementSuccessful = 1;
                end
            end
        end
    end
end
```

Algorithm 1: Local refinement algorithm based on the insertion of new anchors for standard T-spline meshes.

5.4. Local refinement of standard T-splines of even degree by adding anchors

This section explains how the necessary condition for standard T-spline meshes in Equation (51) and nestedness for meshes of even degree can be enforced using the Bézier extraction operator. It should be noted, that in order to be able implement the methods described in the following, the local knot vectors for each anchor are required in the index (u_ℓ) and sub-parameter (ξ_ℓ) domain – it is not sufficient to have only access to the Bézier extraction operator.

5.4.1. Example 1: Ensuring that $\underline{\underline{\mathbf{C}}}_e$ is a square matrix and nestedness

Prepared using nmeauth.cls
DOI: 10.1002/nme

Initial refinement

We consider the quadratic standard T-spline mesh in the index domain and the physical domain in Figure 11. It is refined by insertion of an anchor which results in the rectangle $[\xi_1^2, \xi_1^4] \times [\xi_2^3, \xi_2^5]$ being split vertically, see Figure 12(a).

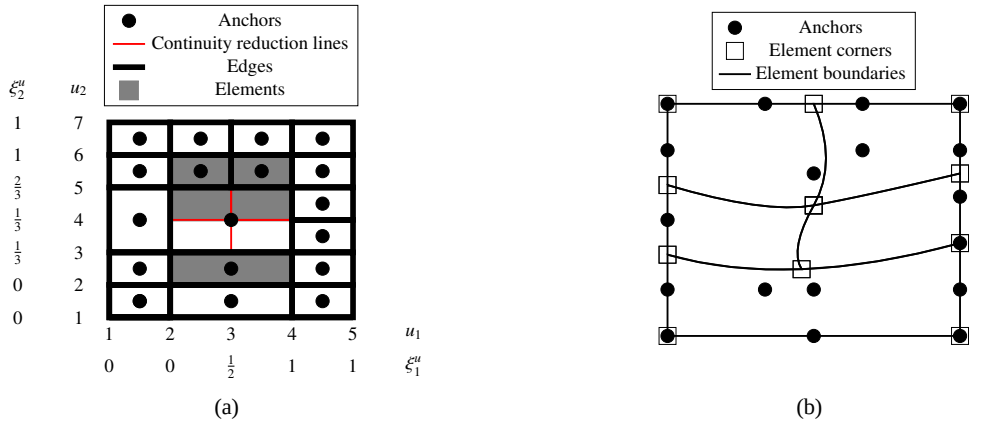


Figure 11. Initial quadratic standard T-spline mesh in (a) the index domain and (b) the physical domain.

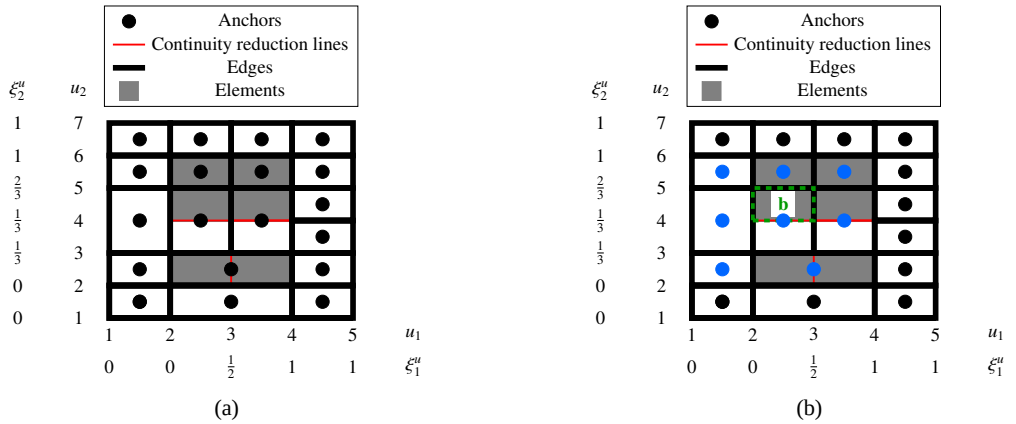


Figure 12. (a) Refined quadratic non-standard T-spline mesh from Figure 11(a) in the index domain. (b) The T-spline mesh is locally linearly independent – but as only eight anchors (blue) have a support in element b (dashed green line), $\underline{\underline{\mathbf{C}}}_e$ is not a square matrix for element b.

Ensuring that $\underline{\underline{\mathbf{C}}}_e$ is a square matrix

The resulting mesh is locally linearly independent, but non-standard and, $\underline{\underline{\mathbf{C}}}_e$ is not a square matrix for all elements. Indeed, for element b (bounded by a dashed green line), we have $\mathrm{rank}(\underline{\underline{\mathbf{C}}}_e) = n_e$, as there are only eight anchors (blue) with a support, $n_e = 8$, see Figure 12(b). Hence, additional anchors need to be inserted in order to obtain a square matrix $\underline{\underline{\mathbf{C}}}_e$. Each local knot vector of the blue anchors with support in element b in Figure 12(b) contains the sub-parameter values of the boundaries of element b – $[0, \frac{1}{2}] \times [\frac{1}{3}, \frac{2}{3}]$ in the ξ_1 direction and the ξ_2 direction, respectively, except for the anchors A and B in Figure 13(a). The local knot vectors of the anchors A and B in the ξ_1 direction do not contain the sub-parameter value $\xi_1 = \frac{1}{2}$. Therefore, rectangle c needs to be split. This results in the standard T-spline mesh in Figure 13(b).

DOI: 10.1002/nme

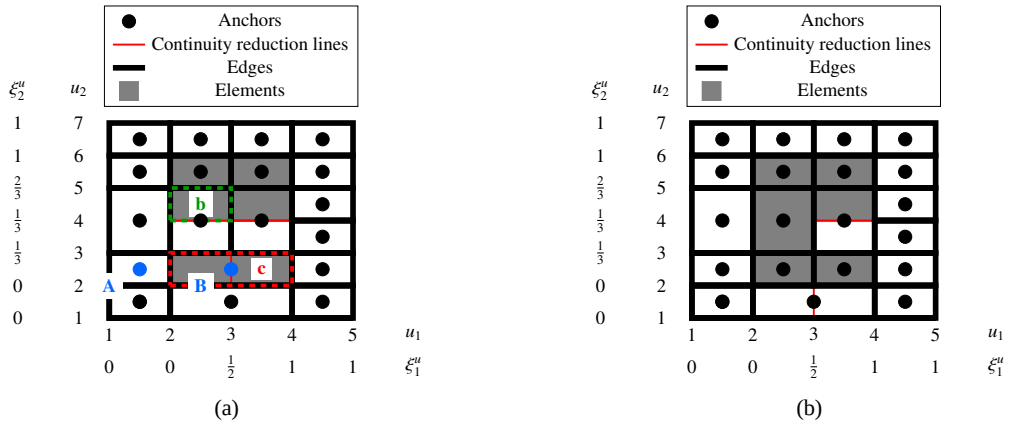


Figure 13. Procedure to obtain a square matrix $\underline{\underline{\mathbf{C}}}_e$. (a) The local knot vectors of the anchors A and B (blue) do not contain the sub-parameter value $\xi_1 = \frac{1}{2}$, which is a boundary of element b (dashed green). The local knot vectors of all other anchors with support in element b (see Figure 12(b)) contain the sub-parameter values $0, \frac{1}{2}$ in the ξ_1 direction and $\frac{1}{3}, \frac{2}{3}$ in the ξ_2 direction – $[0, \frac{1}{2}] \times [\frac{1}{3}, \frac{2}{3}]$ represents the boundary values of element b in the sub-parameter domain. Hence, the rectangle c needs to be split so that the local knot vectors of the anchors A and B also contain the knot $\xi_1 = \frac{1}{2}$. (b) The resulting standard mesh and the initial mesh in 11(a) are not nested.

Nestedness

The initial T-spline mesh in Figure 11(a) and the refined mesh in Figure 13(b) are not nested: the blending functions of anchors C, D and E (see mesh in Figure 14) cannot be expressed as a linear combination of the blending functions of the refined mesh in Figure 13(b). This can be identified by inspection of the row echelon form of Equation (57) for these anchors.

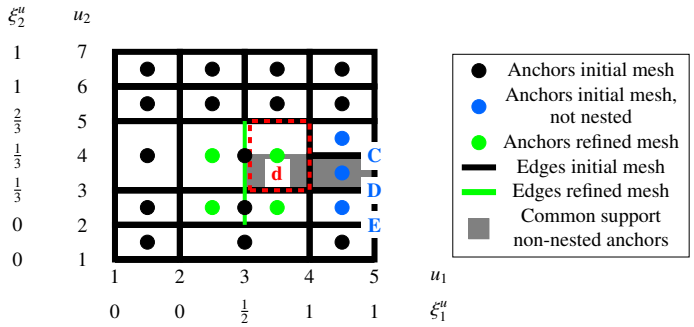


Figure 14. Superposition of the initial T-spline mesh in the index domain from Figure 11(a) and the refined mesh in Figure 13(b). Transforming Equation (57) into row echelon form gives no results for the anchors C, D and E (blue) since the meshes in Figure 11(a) and Figure 13(b) are not nested. Edges and anchors from the refined mesh in Figure 13(b), which were added during refinement, are inserted in the initial mesh from Figure 11(a) and marked with green. Within the grey domain all three anchors C, D and E from the initial mesh have a support, while the grey domain is bounded by the newly inserted green edges. In this grey domain an additional anchor needs to be inserted, i. e. the dashed red rectangle d needs to be subdivided, see Figure 15.

Therefore, an additional anchor has to be inserted. We draw the new edges and anchors of the refined mesh in Figure 13(b) in the initial mesh of Figure 11(a) as illustrated with solid green lines and green points in Figure 14. Then, the grey domain is drawn, highlighting the common support of the three anchors C, D and E which is bounded by the new green edges. Within the grey domain a new anchor needs to be inserted, i. e. the dashed red rectangle d needs to be subdivided. The resulting

DOI: 10.1002/nme

refined mesh has now the sought properties: it is standard and is nested with the initial (non-refined) mesh, i. e. the blending function $\underline{\boldsymbol{N}}$ of each anchor in Figure 11(a) can be represented as a linear combination of the blending functions $\underline{\boldsymbol{N}}_R$ of the anchors in the refined mesh in Figure 15(a).

Refined physical mesh

So far, refinement has only been considered in the index domain in order to obtain a standard and nested T-spline mesh. Next, the evaluation of the weighted control points in the physical domain is addressed.

The location of the weighted control points for the refined mesh $\underline{\boldsymbol{P}}_{w\,R}$ is determined using Equation (69). The physical mesh is shown in Figure 15(b) which preserves the same geometry as the physical mesh in Figure 11(b). This can be observed by comparing for instance the shape of the element boundaries of the initial and the refined physical mesh.

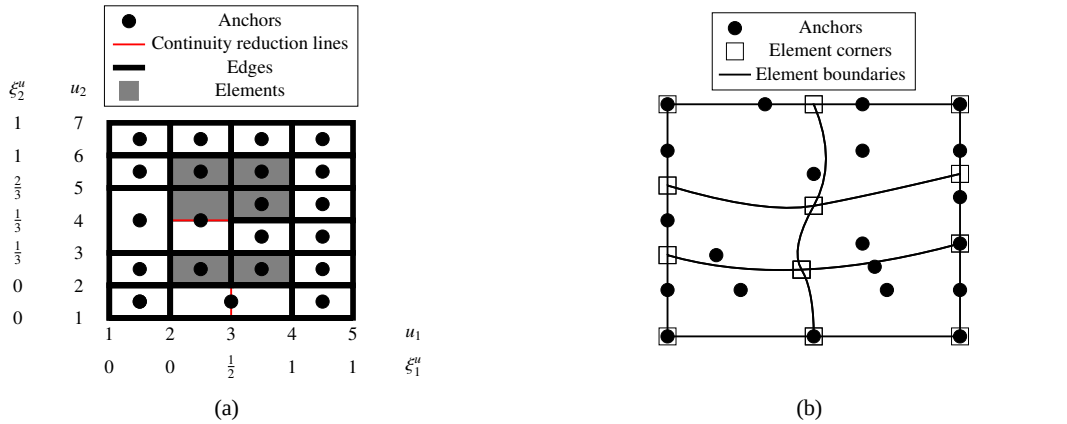


Figure 15. Refined quadratic T-spline mesh of Figure 11 in (a) the index domain and (b) the physical domain. This T-spline mesh is standard and nested with the initial T-spline mesh of Figure 11.

5.4.2. Example 2: Removing linear dependencies

Initial refinement

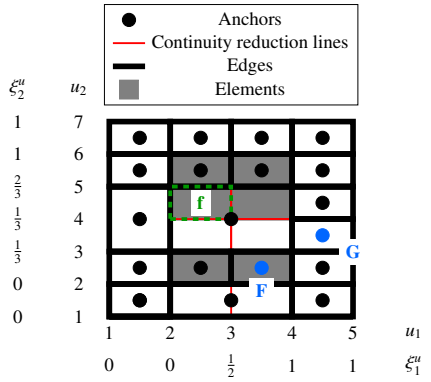
As a next example, the initial quadratic T-spline mesh in Figure 11 is now refined as shown in Figure 16(a).

Removing linear dependencies

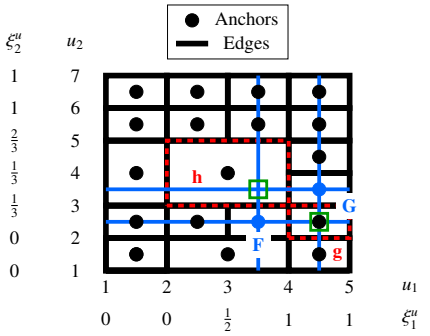
The T-spline mesh of Figure 16(a) is non-standard using Equation (38). Furthermore, the necessary condition Equation (51) is not fulfilled. Transforming Equation (31) into row echelon form yields the dependency $N^F(\underline{\boldsymbol{\xi}}) - N^G(\underline{\boldsymbol{\xi}}) = 0$ in element f. In order to break this dependence, new anchors need to be inserted. In the following, it will be shown how to identify potential locations for these new anchors and how to select the ideal one.

Extension lines (solid blue) are drawn between the anchors F and G as depicted in Figure 16(b). These extension lines intersect at the location of the green squares. These squares are located in the rectangles g and h (dashed red line). Rectangle g cannot be further subdivided, but rectangle h can, as shown in Figure 17.

DOI: 10.1002/nme

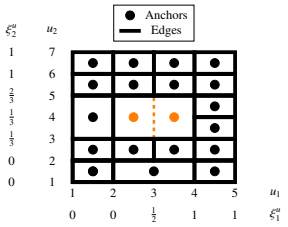


(a) The T-spline mesh is locally linearly dependent – the row echelon version of Equation (31) gives the dependency $N^F(\underline{\boldsymbol{\xi}}) - N^G(\underline{\boldsymbol{\xi}}) = 0$ in element f.

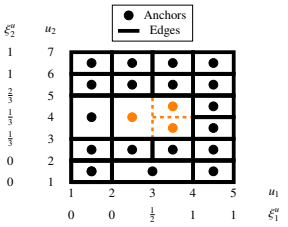


(b) Extension lines (solid blue) for the anchors F and G intersect at the location of the green squares. The rectangles g and h (dashed red line) contain the green squares. Rectangle g cannot be further subdivided. All options for subdividing rectangle h are given in Figure 17.

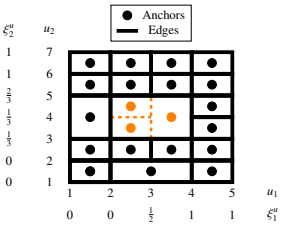
Figure 16. Refined (non-standard) quadratic T-spline mesh from Figure 11(a) in the index domain.



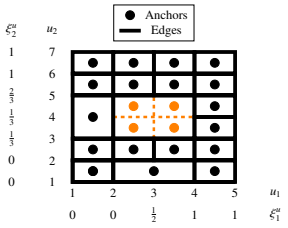
(a) Standard, not nested.



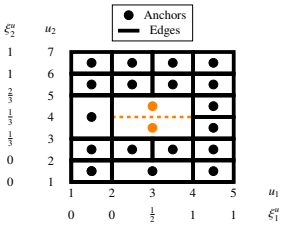
(b) Standard, nested.



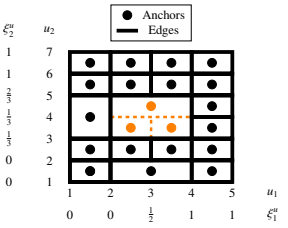
(c) Standard, not nested.



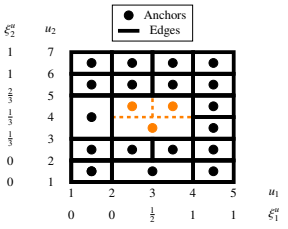
(d) Standard, nested.



(e) Non-standard.



(f) Non-standard.



(g) Non-standard.

Figure 17. All possible subdivisions for the rectangle h in Figure 16(b): the dashed orange lines indicate the new edges to be inserted, the orange points denote the location of the new anchors.

Table I gives a summary of the number of pairs of anchors with linearly dependent blending functions, number of non-square matrices $\underline{\underline{\mathbf{C}}}_e$, nestedness and number of additionally inserted

DOI: 10.1002/nme

anchors for the options in Figure 17. This information can be used in order to determine the best location and optimum number of additional anchors.

Table I. Summary of the number of pairs of anchors with linearly dependent blending functions, number of non-square matrices $\underline{\underline{\mathbf{C}}}_e$, nestedness and number of additionally inserted anchors for the options in Figure 17.

Figure	17(a)	17(b)	17(c)	17(d)	17(e)	17(f)	17(g)
Number of pairs of anchors with linearly dependent blending functions	0	0	0	0	0	0	0
Number of non-square matrices $\underline{\underline{\mathbf{C}}}_e$	0	0	0	0	1	2	2
Nestedness	✗	✓	✗	✓	✗	✗	✗
Number of additional anchors	2	3	3	4	2	3	3

According to Figure 17 and Table I, only the options (b) and (d) are suitable for refinement of the T-spline mesh in Figure 11 since they are standard and nested with the initial mesh. From an implementational point of view, one could select the option which introduces the smallest amount of new anchors, i. e. option (b).

In case that no refinement option results in a standard and nested T-spline mesh, one can select either the option with the smallest number of pairs of anchors with linearly dependent blending functions or the option with the smallest number of non-square matrices $\underline{\underline{\mathbf{C}}}_e$ and then continue with the next refinement step until a standard and nested mesh is obtained, see Appendix C.

5.5. Summary for the local refinement of standard T-splines

The examples for the local refinement of standard T-spline meshes by adding anchors demonstrate that the Bézier extraction operator allows to:

- enforce the necessary condition in Equation (51) for standard T-spline meshes:
  - when the T-spline mesh is locally linearly independent but we do not have a square matrix $\underline{\underline{\mathbf{C}}}_e$ for each element e, the Bézier extraction operator shows, which element does not have enough anchors with a support (Figure 13(a));
  - when there are local linear dependencies, the Bézier extraction operator shows, where new anchors and edges need to be inserted (Figure 16(b))
- pinpoint for which blending functions two T-spline meshes are not nested (Figure 14).

We have found that when the necessary condition in Equation (51) is fulfilled and the refinement matrix $\underline{\underline{\mathbf{M}}}$ in Equation (57) can be computed, we always obtain a nested standard T-spline mesh. We have not experienced a single case where this resulted in a nested semi-standard T-spline mesh. However, should such a case arise, one can pinpoint for which anchors $\beta_R^i \neq 1$ using Equation (59) and insert an additional anchor in the supported domain of these anchors.

The local refinement of standard T-spline meshes of odd degree is treated in Appendix A. Furthermore, Appendix B demonstrates that also non-standard T-splines can be refined locally when nestedness exists.

6. HIERARCHICAL REFINEMENT OF STANDARD, SEMI-STANDARD AND NON-STANDARD T-SPLINES USING THE RECONSTRUCTION OPERATOR

In [15] another refinement strategy was introduced based on the reconstruction operator. Instead of adding new anchors to the mesh as was proposed in the previous section, the method is based on the division of elements while an invertible elemental Bézier extraction operator $\underline{\underline{\mathbf{C}}}_e$ is needed for the reconstruction operator. The hierarchical refinement method in [15] has been derived for

Prepared using nmeauth.cls
DOI: 10.1002/nme

analysis-suitable T-splines. Here, we show how the idea of this concept can also be applied to standard, semi-standard and non-standard T-spline meshes.

6.1. Splitting elements

The hierarchical refinement algorithm based on the reconstruction operator requires local linear independence. Moreover, it requires that $\underline{\underline{\mathbf{C}}}_e$ is a square matrix for the element e that is subdivided,

$$\operatorname{rank}(\underline{\underline{\mathbf{C}}}_e) = \prod_{\ell=1}^{d} p_\ell + 1 \tag{70}$$

since the reconstruction operator, defined as

$$\underline{\underline{\mathbf{R}}}_e = \underline{\underline{\mathbf{C}}}_e^{-1}, \tag{71}$$

is needed. Therefore, for this hierarchical refinement algorithm the Bézier extraction operator plays again a key role: when Equation (70) is satisfied for element e, this element can be refined hierarchically. Thus, this algorithm can be applied to standard, semi-standard and non-standard T-spline meshes.

Consider an element with range $[-1, 1]$ and suppose that we want to split it in half: $[-1, 0]$ and $[0, 1]$. The first Bernstein basis B_1^1 with the knot vector $\{-1, -1, -1, 1\}$ (black curve) in Figure 18 in the element $[-1, 1]$ can be defined in the two sub-elements $[-1, 0]$ and $[0, 1]$ as a linear combination of the Bernstein polynomials in the two sub-elements: the Bernstein basis functions for the left part of the element with support in $[-1, 0]$ are given by the local knot vectors

$$B_{1\,l}^1 \text{ for } \{-1, -1, -1, 0\}, \quad B_{1\,l}^2 \text{ for } \{-1, -1, 0, 0\}, \quad B_{1\,l}^3 \text{ for } \{-1, 0, 0, 0\}. \tag{72}$$

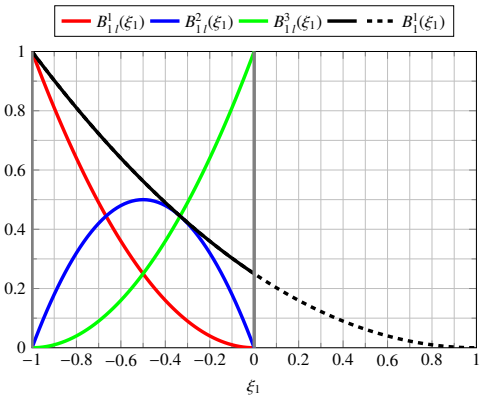


Figure 18. The Bernstein polynomial B_1^1 with support over the element $[-1, 1]$ can be expressed in the sub-element e_l with range $[-1, 0]$ as a linear combination of the Bernstein polynomials $B_{1\,l}^a$: $B_1^1(\xi_1) = B_{1\,l}^1(\xi_1) + \frac{1}{2}B_{1\,l}^2(\xi_1) + \frac{1}{4}B_{1\,l}^3(\xi_1)$.

The Bernstein polynomial B_1^1 in the left part of the element (solid black line) can now be expressed as a linear combination of the Bernstein polynomials $B_{1\,l}^i$ as follows

$$B_1^1(\xi_1) = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{4} \end{bmatrix} \begin{bmatrix} B_{1\,l}^1(\xi_1) \\ B_{1\,l}^2(\xi_1) \\ B_{1\,l}^3(\xi_1) \end{bmatrix}. \tag{73}$$

The coefficients in Equation (73) can either be obtained using the algorithm in [7] for the knot vector $\{-1, -1, -1, 1\}$ with an interior knot (causing a discontinuity) at $\xi_1 = 0$ or, alternatively, using the relations in [21]. Applying the same procedure to B_1^2, B_1^3, and on the right part of the element (with

Prepared using nmeauth.cls
DOI: 10.1002/nme

the range $[0,1]$) gives

$$\underbrace{\begin{bmatrix} B_1^1(\xi_1) \\ B_1^2(\xi_1) \\ B_1^3(\xi_1) \end{bmatrix}}_{\underline{\boldsymbol{B}}_1(\xi_1)} = \underbrace{\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{4} \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & \frac{1}{4} \end{bmatrix}}_{\underline{\underline{\mathbf{A}}}_{1\,l}} \underbrace{\begin{bmatrix} B_{1\,l}^1(\xi_1) \\ B_{1\,l}^2(\xi_1) \\ B_{1\,l}^3(\xi_1) \end{bmatrix}}_{\underline{\boldsymbol{B}}_{1\,l}(\xi_1)} + \underbrace{\begin{bmatrix} \frac{1}{4} & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{4} & \frac{1}{2} & 1 \end{bmatrix}}_{\underline{\underline{\mathbf{A}}}_{1\,r}} \underbrace{\begin{bmatrix} B_{1\,r}^1(\xi_1) \\ B_{1\,r}^2(\xi_1) \\ B_{1\,r}^3(\xi_1) \end{bmatrix}}_{\underline{\boldsymbol{B}}_{1\,r}(\xi_1)}. \tag{74}$$

Hence, the Bernstein polynomials $\underline{\boldsymbol{B}}_1$ over one element e with the span $[-1,1]$ can be expressed as a linear combination of the Bernstein polynomials $\underline{\boldsymbol{B}}_{1\,l}$ and $\underline{\boldsymbol{B}}_{1\,r}$ over the two smaller elements e_l with the span $[-1,0]$ and e_r with the span $[0,1]$. Extending Equation (74) into more dimensions gives

$$\underline{\boldsymbol{B}}(\underline{\boldsymbol{\xi}}) = \underline{\underline{\mathbf{A}}}_l \underline{\boldsymbol{B}}_l(\underline{\boldsymbol{\xi}}) + \underline{\underline{\mathbf{A}}}_r \underline{\boldsymbol{B}}_r(\underline{\boldsymbol{\xi}}). \tag{75}$$

We next assume that the Bézier extraction operator is known for the original, single element $\underline{\underline{\mathbf{C}}}_e$ and for the two sub-elements $\underline{\underline{\mathbf{C}}}_{e\,l}$ and $\underline{\underline{\mathbf{C}}}_{e\,r}$. Then, we can express the weighted curve $\underline{\boldsymbol{T}}_{w\,e}$ with the weighted control points $\underline{\boldsymbol{P}}_{w\,e}^i$ over element e using Equation (75)

$$\underline{\boldsymbol{T}}_{w\,e}(\underline{\boldsymbol{\xi}}) = \sum_{i=1}^{n_e} \underline{\boldsymbol{C}}_e^{i\,T} \underline{\boldsymbol{B}}(\underline{\boldsymbol{\xi}}) \underline{\boldsymbol{P}}_{w\,e}^i = \sum_{i=1}^{n_e} \underline{\boldsymbol{C}}_e^{i\,T} \left(\underline{\underline{\mathbf{A}}}_l \underline{\boldsymbol{B}}_l(\underline{\boldsymbol{\xi}}) + \underline{\underline{\mathbf{A}}}_r \underline{\boldsymbol{B}}_r(\underline{\boldsymbol{\xi}}) \right) \underline{\boldsymbol{P}}_{w\,e}^i \tag{76}$$

and over the two sub-elements

$$\underline{\boldsymbol{T}}_{w\,e}(\underline{\boldsymbol{\xi}}) = \underline{\boldsymbol{T}}_{w\,e\,l}(\underline{\boldsymbol{\xi}}) + \underline{\boldsymbol{T}}_{w\,e\,r}(\underline{\boldsymbol{\xi}}) = \sum_{j=1}^{n_{e\,l}} \underline{\boldsymbol{C}}_{e\,l}^{j\,T} \underline{\boldsymbol{B}}_l(\underline{\boldsymbol{\xi}}) \underline{\boldsymbol{P}}_{w\,e\,l}^j + \sum_{k=1}^{n_{e\,r}} \underline{\boldsymbol{C}}_{e\,r}^{k\,T} \underline{\boldsymbol{B}}_r(\underline{\boldsymbol{\xi}}) \underline{\boldsymbol{P}}_{w\,e\,r}^k \tag{77}$$

with the weighted control points $\underline{\boldsymbol{P}}_{w\,e\,l}$ and $\underline{\boldsymbol{P}}_{w\,e\,r}$ for element e_l and e_r, respectively. Comparing Equation (76) and Equation (77) results in

$$\sum_{i=1}^{n_e} \underline{\boldsymbol{C}}_e^{i\,T} \underline{\underline{\mathbf{A}}}_l \underline{\boldsymbol{B}}_l(\underline{\boldsymbol{\xi}}) \underline{\boldsymbol{P}}_{w\,e}^i = \sum_{j=1}^{n_{e\,l}} \underline{\boldsymbol{C}}_{e\,l}^{j\,T} \underline{\boldsymbol{B}}_l(\underline{\boldsymbol{\xi}}) \underline{\boldsymbol{P}}_{w\,e\,l}^j \tag{78}$$

$$\sum_{i=1}^{n_e} \underline{\boldsymbol{C}}_e^{i\,T} \underline{\underline{\mathbf{A}}}_r \underline{\boldsymbol{B}}_r(\underline{\boldsymbol{\xi}}) \underline{\boldsymbol{P}}_{w\,e}^i = \sum_{k=1}^{n_{e\,r}} \underline{\boldsymbol{C}}_{e\,r}^{k\,T} \underline{\boldsymbol{B}}_r(\underline{\boldsymbol{\xi}}) \underline{\boldsymbol{P}}_{w\,e\,r}^k \tag{79}$$

or in vector-matrix form

$$\underline{\underline{\mathbf{C}}}_e^{\,T} \underline{\underline{\mathbf{A}}}_l \underline{\boldsymbol{P}}_{w_e} = \underline{\underline{\mathbf{C}}}_{e\,l}^{\,T} \underline{\boldsymbol{P}}_{w_{e\,l}}, \tag{80}$$

$$\underline{\underline{\mathbf{C}}}_e^{\,T} \underline{\underline{\mathbf{A}}}_r \underline{\boldsymbol{P}}_{w_e} = \underline{\underline{\mathbf{C}}}_{e\,r}^{\,T} \underline{\boldsymbol{P}}_{w_{e\,r}}. \tag{81}$$

Hence, the weighted coordinates of the two sub-elements are obtained with the reconstruction operator in Equation (71) as

$$\underline{\boldsymbol{P}}_{w_{e\,l}} = \underline{\underline{\mathbf{R}}}_{e\,l}^{\,T} \underline{\underline{\mathbf{C}}}_e^{\,T} \underline{\underline{\mathbf{A}}}_l \underline{\boldsymbol{P}}_{w_e}, \qquad \underline{\boldsymbol{P}}_{w_{e\,r}} = \underline{\underline{\mathbf{R}}}_{e\,r}^{\,T} \underline{\underline{\mathbf{C}}}_e^{\,T} \underline{\underline{\mathbf{A}}}_r \underline{\boldsymbol{P}}_{w_e}. \tag{82}$$

6.2. Example

As an example we consider the quadratic non-standard T-spline mesh of Figure 19 which is globally linearly independent but locally linearly dependent.

The dashed green element b is now divided vertically into two sub-element b_l and b_r with range $[0,\frac{1}{4}]\times[\frac{2}{4},\frac{3}{4}]$ and $[\frac{1}{4},\frac{1}{2}]\times[\frac{2}{4},\frac{3}{4}]$, i. e. the knot value $\xi_1=\frac{1}{4}$ is inserted in element b. Element b can be subdivided since Equation (70) holds for it. In order to obtain the weighted control points $\underline{\boldsymbol{P}}_{w_{e\,l}}$ and $\underline{\boldsymbol{P}}_{w_{e\,r}}$ in Equation (82), the reconstruction operators $\underline{\underline{\mathbf{R}}}_{e\,l}$ and $\underline{\underline{\mathbf{R}}}_{e\,r}$, which follow from the Bézier extraction operators $\underline{\underline{\mathbf{C}}}_{e\,l}$ and $\underline{\underline{\mathbf{C}}}_{e\,r}$, respectively, are needed for element b. These Bézier extraction operators are based on the modified local knot vectors of the sub-elements b_l and b_r

DOI: 10.1002/nme

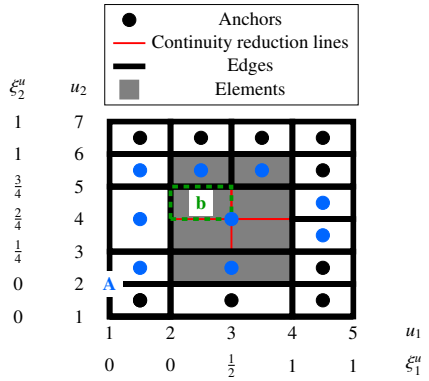
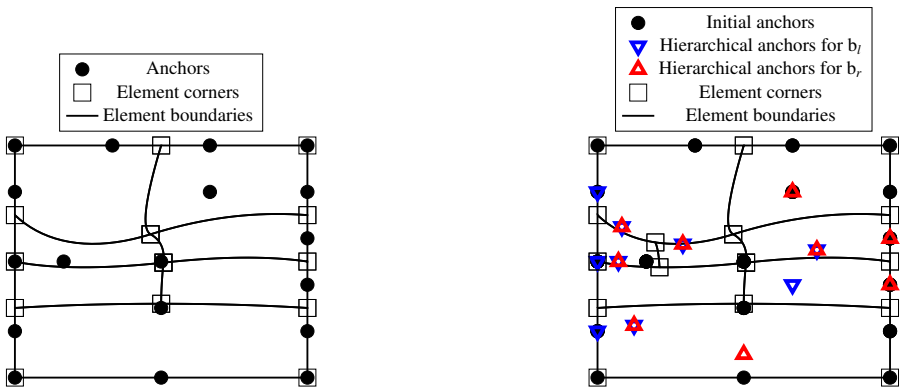


Figure 19. Non-standard, globally linearly independent T-spline mesh in the index domain (from Figure 7(b)). Element b (dashed green line) with range $[0, \frac{1}{2}] \times [\frac{2}{4}, \frac{3}{4}]$ is split vertically into two sub-elements b_l and b_r with range $[0, \frac{1}{4}] \times [\frac{2}{4}, \frac{3}{4}]$ and $[\frac{1}{4}, \frac{1}{2}] \times [\frac{2}{4}, \frac{3}{4}]$, respectively. Each local knot vector associated to a blue anchor (i. e. those having a support in element b) needs to be modified. For instance, the anchor A with $\Xi_1^A = \{0, 0, 0, 1\}$ becomes $\Xi_{1\,l}^A = \{0, 0, 0, \frac{1}{4}\}$ in element b_l and $\Xi_{1\,r}^A = \{0, 0, \frac{1}{4}, 1\}$ in element b_r. $\Xi_1^A = \{0, 0, 0, 1\}$ remains unchanged for the other elements. The modified local knot vectors for the other blue anchors are given in Appendix D.

which are obtained as follows. We pick from each blue anchor i which has a support over element b in Figure 19 the local knot vector Ξ_1^i. Then we insert into this local knot vector the knot value $\xi_1 = \frac{1}{4}$ and split the resulting knot vector into two knot vectors of length $p_\ell + 2$, where one knot vector contains the first $p_\ell + 2$ entries and the other one the last $p_\ell + 2$ entries. For instance, taking the anchor A in Figure 19 gives the local knot vector $\Xi_1^A = \{0, 0, 0, 1\}$. The local knot vectors for the elements b_l and b_r are then $\Xi_{1\,l}^A = \{0, 0, 0, \frac{1}{4}\}$ and $\Xi_{1\,r}^A = \{0, 0, \frac{1}{4}, 1\}$. We note that the local knot vector Ξ_1^A is modified only in the elements b_l and b_r, while for the other elements Ξ_1^A remains unchanged. The local knot vectors for the blue anchors in the sub-elements b_l and b_r are given in Appendix D.

The initial non-standard T-spline mesh and the hierarchically refined non-standard T-spline mesh in the physical domain are depicted in Figure 20. Both physical meshes represent the same geometry.



(a) (b)

Figure 20. (a) Initial and (b) hierarchically refined non-standard T-spline mesh from Figure 19 in the physical domain.

DOI: 10.1002/nme

7. DISCUSSION AND CONCLUSIONS

We have classified T-spline meshes of arbitrary degree according to linear independence and the partition of unity property. We have also shown how to refine standard T-spline meshes by adding anchors, such that the initial and the refined T-spline mesh are nested. It has been demonstrated that non-standard meshes can also be refined by adding anchors when nestedness exists. All methods exploit the Bézier extraction operator, which appears to play a central role. Furthermore, it has been shown that hierarchical refinement of standard, semi-standard and non-standard T-spline meshes using the reconstruction operator basically also involves the Bézier extraction operator, since the reconstruction operator is just its inverse.

We finally note that the term "analysis-suitable" might cause confusion. Analysis can be performed with standard, semi-standard and non-standard T-spline meshes. In our view, the requirements for a T-spline mesh to be suitable for analysis are

- the blending functions N^i are globally linearly independent (Equation (25) holds)
- the partition of unity property holds in order to satisfy the affine transformation and the patch test.

We note that the local/global linear independence of the blending functions N^i results in the local/global linear independence of the rational blending functions R^i. Hence, globally linearly independent semi-standard and non-standard T-spline meshes which employ the rational blending functions R^i in Equation (32) can be used for analysis since the rational blending functions R^i always form a partition of unity. It has also been demonstrated that semi-standard and non-standard T-spline meshes can be refined locally by either adding new anchors, or by splitting existing elements. Refining semi-standard and non-standard T-spline meshes by adding anchors requires nestedness, while the hierarchical refinement for semi-standard and non-standard meshes requires the satisfaction of Equation (70).

A. LOCAL REFINEMENT OF STANDARD T-SPLINES OF ODD DEGREE BY ADDING ANCHORS

Herein it is explained how the necessary condition for standard T-spline meshes of Equation (51) and nestedness for meshes of odd degree can be enforced using the Bézier extraction operator.

A.1. Example 1: Ensuring that $\underline{\underline{\mathbf{C}}}_e$ is a square matrix and nestedness

Initial refinement
We start with the cubic standard T-spline mesh depicted in Figure 21 which is refined as in Figure 22(a).

Ensuring that $\underline{\underline{\mathbf{C}}}_e$ is a square matrix
The mesh in Figure 22(a) is locally linearly independent, but non-standard and $\underline{\underline{\mathbf{C}}}_e$ is not a square matrix for all elements. For instance, we have $\mathrm{rank}(\underline{\underline{\mathbf{C}}}_e) = n_e$ in element b as there are only fifteen anchors (blue) with a support, Figure 22(b). Hence, an additional anchor needs to be inserted. Each local knot vector of the blue anchors in Figure 22(b) contains the sub-parameter values of the boundaries of element b – $[0, \frac{1}{2}] \times [\frac{1}{3}, \frac{2}{3}]$ in the ξ_1 direction and the ξ_2 direction, respectively – except for the anchors A and B in Figure 23(a). The local knot vectors of the anchors A and B in the ξ_1 direction do not contain the sub-parameter value $\xi_1 = \frac{1}{2}$. Therefore, an additional anchor needs to be inserted at the location of the red point c. This results in the standard mesh in Figure 23(b).

Nestedness
Unfortunately, the initial mesh in Figure 21(a) and the refined mesh in Figure 23(b) are not nested. Transforming Equation (57) into row echelon form gives no solution for the anchors C, D, E and

DOI: 10.1002/nme

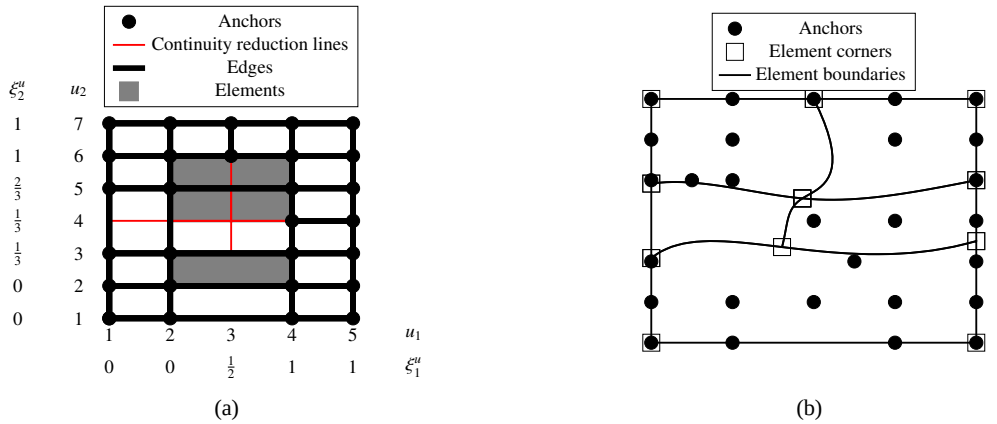


Figure 21. Initial cubic standard T-spline mesh in (a) the index domain and (b) the physical domain.

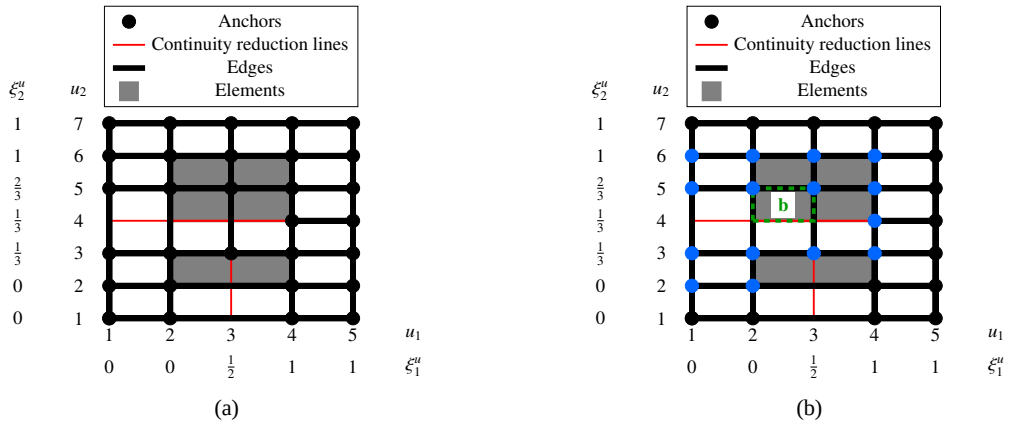


Figure 22. (a) Refined cubic non-standard T-spline mesh from Figure 21(a) in the index domain; (b) the T-spline mesh is locally linearly independent, but in element b (dashed green line) are only fifteen anchors (blue) with a support and therefore $\underline{\underline{\mathbf{C}}}_e$ is not a square matrix for this element.

F in the initial mesh, see Figure 24, i. e. the blending functions associated to these anchors in the initial T-spline mesh cannot be represented as a linear combination of the blending functions of the refined T-spline mesh in Figure 23(b).

DOI: 10.1002/nme

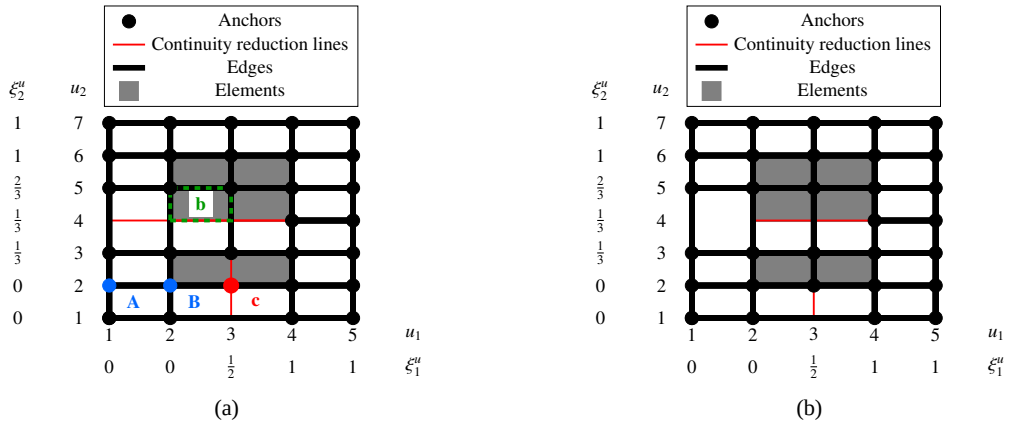


Figure 23. Determination of the location of a new anchor when $\underline{\underline{\mathbf{C}}}^e$ is not a square matrix for a cubic T-spline mesh. (a) The local knot vectors in the ξ_1 direction of the blue anchors A and B do not contain the sub-parameter value $\xi_1 = \frac{1}{2}$, which is a boundary of element b (dashed green). Therefore, an anchor is required at the location of the red point c. (b) The resulting standard T-spline mesh. This T-spline mesh and the initial T-spline mesh in 21(a) are not nested.

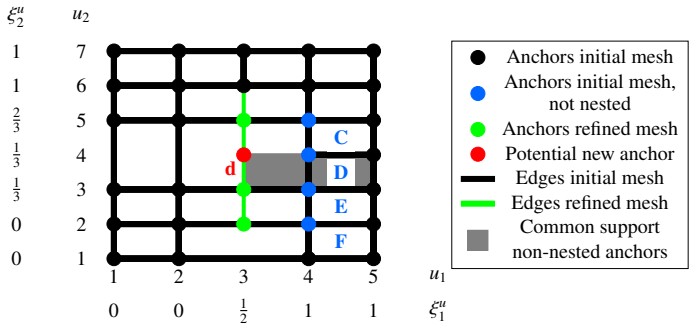


Figure 24. Superposition of the initial T-spline mesh in the index domain from Figure 21(a) and the refined mesh in Figure 23(b). The row echelon form of Equation (57) gives no results for the anchors C, D, E and F (blue) and therefore, the meshes in Figure 21(a) and Figure 23(b) are not nested. Edges and anchors from the refined mesh in Figure 23(b), which were added during refinement, are inserted in the initial mesh from Figure 21 and marked with green. In the grey domain all four anchors C, D, E and F have a common support, while the grey domain is bounded by the newly inserted green edges. Within the grey domain, no anchor is at the position of the red point d. In order to obtain a refined mesh which is standard *and* nested with the initial mesh in Figure 21, the anchor d needs to be inserted into the mesh of Figure 23(b), see also Figure 25.

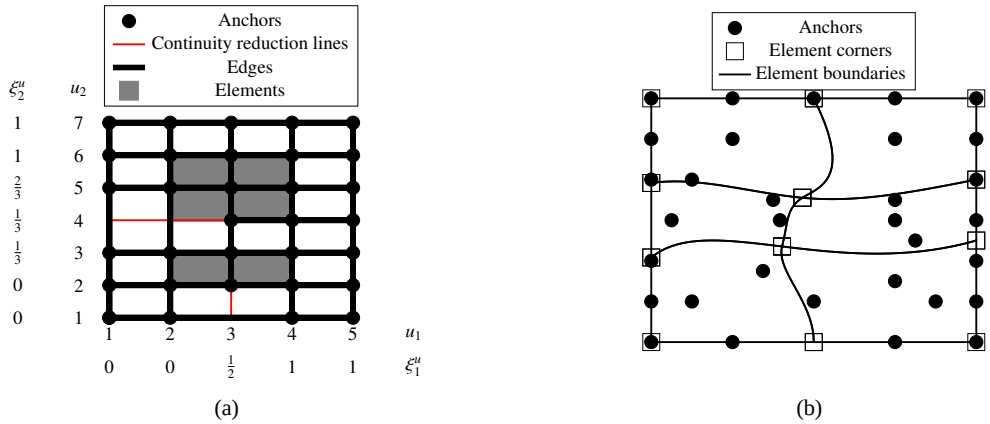
Therefore, an additional anchor has to be inserted. We draw the new edges and anchors of the refined T-spline mesh of Figure 23(b) in the initial T-spline mesh of Figure 21(a) as illustrated with solid green lines and points in Figure 24. Then, the domain where all four anchors C, D, E and F have a common support is drawn while this domain needs to be cut by the green edge. This domain is indicated with a grey colour. It can be observed that within the grey domain no anchor is sitting at the location of the red point d. Therefore, the red point d represents the location of an anchor which has to be inserted into the T-spline mesh. The resulting T-spline mesh is depicted in Figure 25(a). This T-spline mesh is standard. Furthermore, the initial T-spline mesh in Figure 21(a) and the refined T-spline mesh in Figure 25(a) are nested.

Refined physical mesh

After obtaining a standard and nested T-spline mesh in the index domain we can now consider the computation of the physical mesh. The location of the weighted control points for the refined

DOI: 10.1002/nme

mesh $\underline{\boldsymbol{P}}_{wR}$ can be determined using Equation (69). The physical mesh after refinement is shown in Figure 25(b) which represents the same geometry as the physical mesh in Figure 21(b).



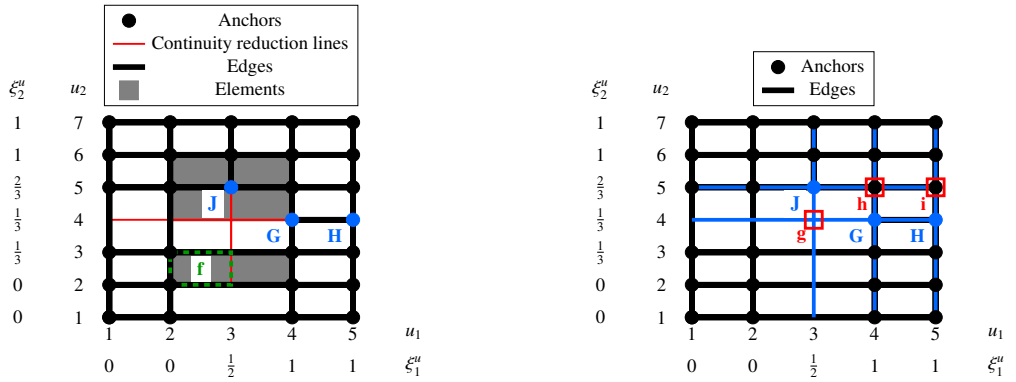
(a) (b)

Figure 25. Refined cubic T-spline mesh of Figure 21 in (a) the index domain and (b) the physical domain. This T-spline mesh is standard and nested with the initial T-spline mesh of Figure 21.

A.2. Example 2: Removing linear dependencies

Initial refinement

As a next example, the initial cubic standard T-spline mesh in Figure 21 is refined as shown in Figure 26(a).



(a) The T-spline mesh is locally linearly dependent – the row echelon form of Equation (31) results in the dependency $-2N^G(\underline{\boldsymbol{\xi}}) + 2N^H(\underline{\boldsymbol{\xi}}) + 3N^J(\underline{\boldsymbol{\xi}}) = 0$ in element f (dashed green).

(b) Illustration for the determination of the location of new anchors. Extension lines (solid blue) for the anchors with locally linearly dependent blending functions G, H and J are drawn. The extension lines intersect at the location of the red squares g, h and i. Only the square g represents a location for a new anchor (see Figure 27) since at h and i anchors are already located.

Figure 26. Refined (non-standard) cubic T-spline mesh from Figure 21(a) in the index domain.

Removing linear dependencies

The T-spline mesh in Figure 26(a) is non-standard using Equation (38). The row echelon version of Equation (31) yields the dependency $-2N^G(\underline{\boldsymbol{\xi}}) + 2N^H(\underline{\boldsymbol{\xi}}) + 3N^J(\underline{\boldsymbol{\xi}}) = 0$ in element f. Therefore, an additional anchor needs to be inserted. This will be done in a manner similar to Section 5.4: extension lines (solid blue) are drawn between the anchors with locally linearly dependent blending

DOI: 10.1002/nme

functions G, H and J, Figure 26(b). The intersections of the extension lines are marked with the red squares. These squares denote possible positions for a new anchor if there does not already exist one. It can be observed from Figure 26(b) that only the intersection at the red square g is a candidate for a new anchor. However, the T-spline mesh with a new anchor in Figure 27 is still locally linearly dependent and semi-standard, so that more anchors and edges need to be inserted by applying the aforementioned methods until a standard and nested T-spline mesh is obtained.

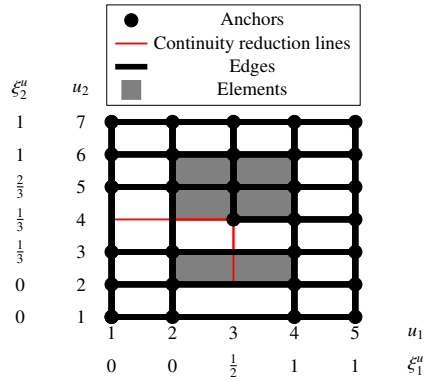
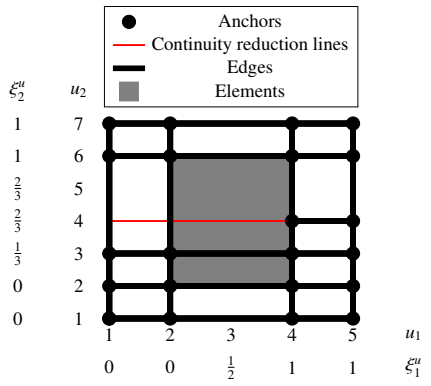


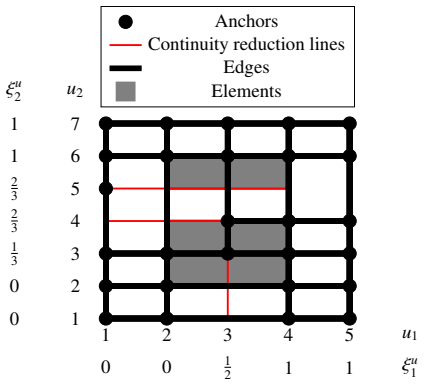
Figure 27. Refined cubic T-spline mesh from Figure 26(a). This T-spline mesh is semi-standard.

A.3. Example 3: Non-standard T-spline fulfils necessary condition for standard T-splines

In the examples considered so far, enforcing Equation (51) resulted in a standard T-spline mesh. However, this is not always the case as Equation (51) is not sufficient for obtaining a standard T-spline mesh – it is only a necessary condition for standard T-splines. Figure 28(b) presents a case where enforcing Equation (51) does not result in a standard T-spline mesh.



(a) Standard, $\mathrm{rank}(\underline{\underline{\mathbf{C}}}_e) = \prod_{\ell=1}^{d} p_\ell + 1$ for $e = 1 \dots E$.



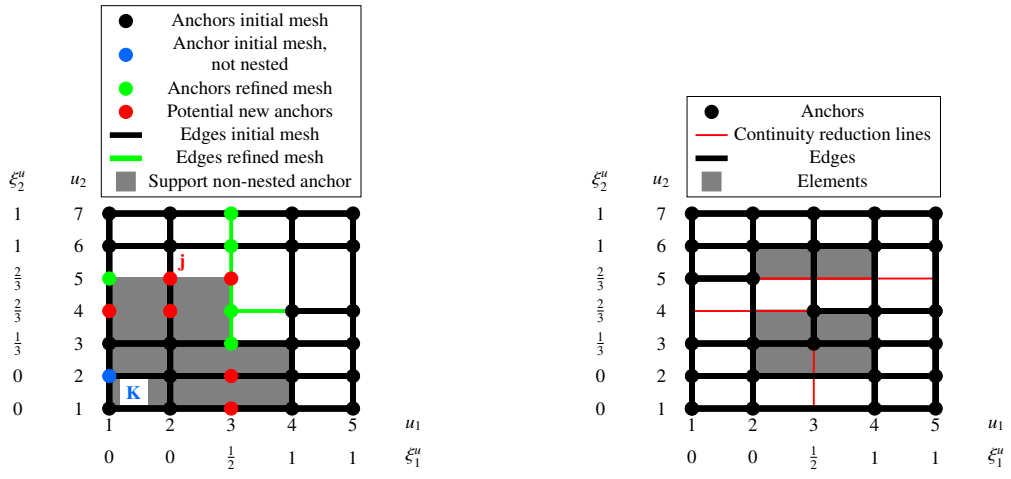
(b) Non-standard, $\mathrm{rank}(\underline{\underline{\mathbf{C}}}_e) = \prod_{\ell=1}^{d} p_\ell + 1$ for $e = 1 \dots E$.

Figure 28. (a) Initial standard T-spline mesh and (b) refined non-standard T-spline mesh. For both meshes Equation (51) holds, but both T-spline meshes are not nested as shown in Figure 29(a).

Both T-spline meshes in Figure 28 are locally linearly independent with a square matrix $\underline{\underline{\mathbf{C}}}_e$. The initial T-spline mesh, Figure 28(a), is standard, while the refined T-spline mesh, Figure 28(b), is non-standard. We recall, that a standard and a non-standard T-spline mesh cannot be nested according to Equation (59): evaluating the refinement matrix in Equation (57) gives no solution for the anchor

DOI: 10.1002/nme

K – and therefore both T-spline meshes are not nested. Applying the same procedure as previously explained (see also Figure 24) gives the possible locations for new anchors (red) as depicted in Figure 29(a).



(a) (b)

Figure 29. (a) Superposition of the initial T-spline mesh from Figure 28(a) and the newly inserted edges and anchors (green) from the refined T-spline mesh in Figure 28(b). The blending function of anchor K in the initial T-spline mesh cannot be represented as a linear combination of the blending functions of the refined T-spline mesh, i. e. both T-spline meshes are not nested since the row echelon form of Equation (57) gives no result for anchor K. The support of anchor K – bounded by the new green edges and anchors – is depicted with a grey domain. Within the grey domain, new anchors can be inserted at the location of the red points in order to obtain a standard and nested T-spline mesh. For instance, inserting the red anchor j would result in a standard and nested T-spline mesh as presented in (b).

B. LOCAL REFINEMENT OF NON-STANDARD T-SPLINES BY ADDING ANCHORS

In the previous examples in Section 5.4 and Appendix A, we demonstrated how to refine a standard T-spline mesh and obtain a standard mesh based on Algorithm 1. In this section we give an example that non-standard meshes can also be refined locally by adding anchors – the only requirement is that the initial and the refined mesh are nested.

Figure 30 shows the initial non-standard and Figure 31 the refined semi-standard quadratic T-spline mesh in the index and physical domain, respectively. Both meshes are nested, which allows the calculation of the weighted control points $\underline{\boldsymbol{P}}_{wR}$ in Equation (69).

DOI: 10.1002/nme

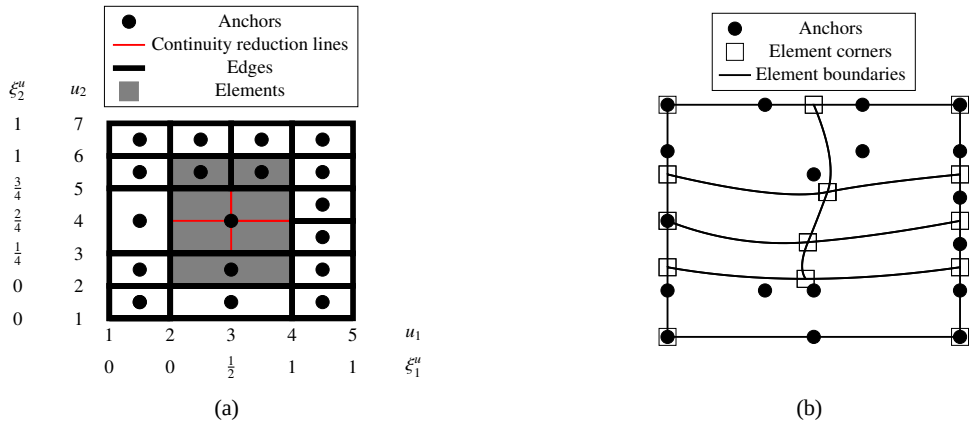


Figure 30. Initial (non-standard) quadratic T-spline mesh in (a) the index domain and (b) the physical domain.

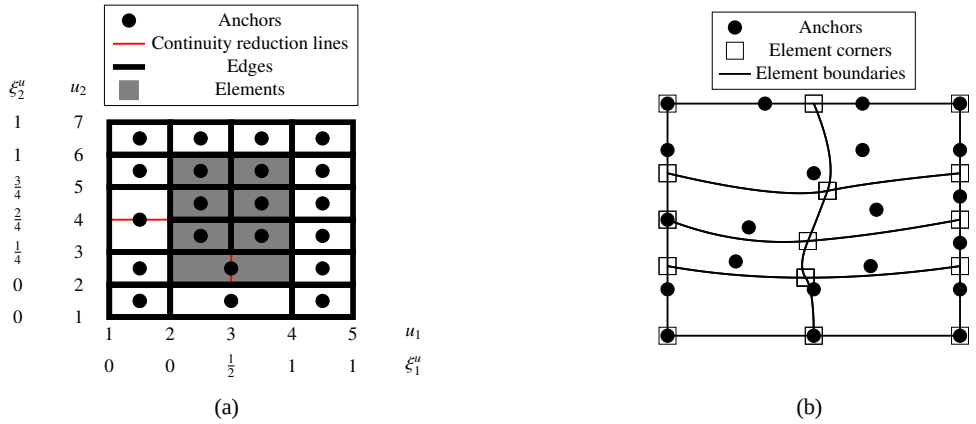


Figure 31. Refined (semi-standard) quadratic T-spline mesh of Figure 30 in (a) the index domain and (b) the physical domain.

C. OBTAINING THE OPTIMISED NUMBER OF ADDITIONALLY INSERTED ANCHORS

Consider the initial (standard) and refined (non-standard) T-spline mesh in Figure 32.

Figure 33 shows all the options where additional anchors can be inserted by applying the routines from Section 5.4, while due to symmetry only the options for the lower part of Figure 32(b) are considered.

DOI: 10.1002/nme

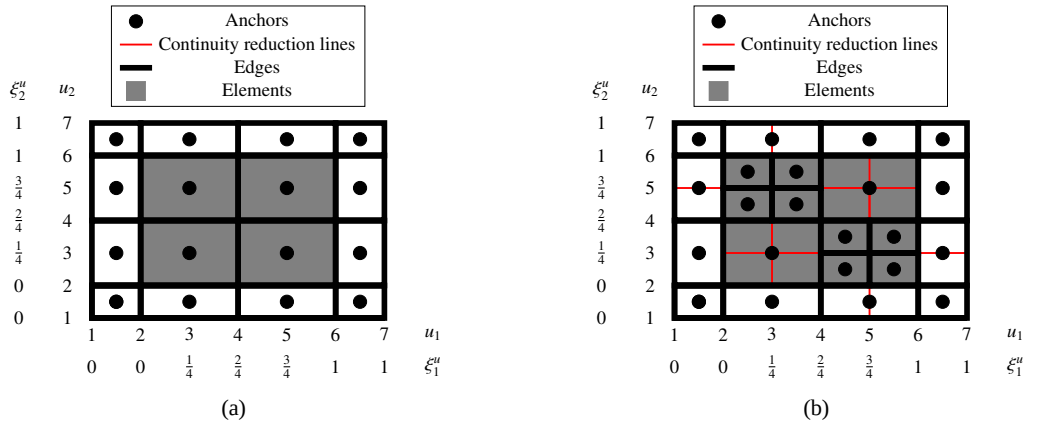


Figure 32. (a) Initial standard and (b) refined non-standard T-spline mesh in the index domain.

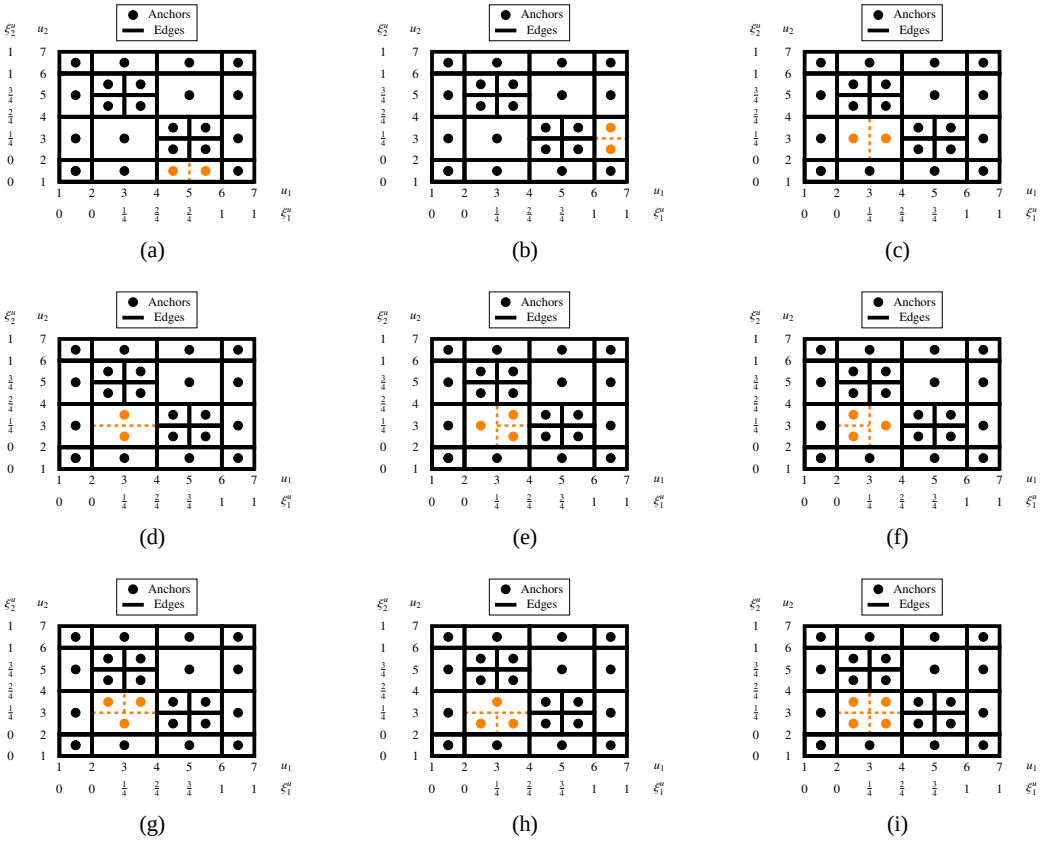


Figure 33. All possible subdivisions for Figure 32(b): the dashed orange lines indicate the new edges to be inserted, the orange points denote the location of the new anchors.

Table II gives the number of pairs of anchors with linearly dependent blending functions, number of non-square matrices $\underline{\underline{\mathbf{C}}}_e$, nestedness and number of additionally inserted anchors for the options in Figure 33.

According to Table II, the optimum option would be either Figure 33(e) or Figure 33(g) since they yield the smallest number of pairs of anchors with linearly dependent blending functions and number of non-square matrices $\underline{\underline{\mathbf{C}}}_e$. After inserting the additional anchors, again, the rectangles to be

DOI: 10.1002/nme

Table II. Summary of the number of pairs of anchors with linearly dependent blending functions, number of non-square matrices $\underline{\underline{\mathbf{C}}}_e$, nestedness and number of additionally inserted anchors for the options in Figure 33.

Figure	33(a)	33(b)	33(c)	33(d)	33(e)	33(f)	33(g)	33(h)	33(i)
Number of pairs of anchors with linearly dependent blending functions	5	5	6	6	5	6	5	6	6
Number of non-square matrices $\underline{\underline{\mathbf{C}}}_e$	8	8	8	8	6	6	6	6	6
Nestedness	✗	✗	✗	✗	✗	✗	✗	✗	✗
Number of additional anchors	2	2	2	2	3	3	3	3	4

subdivided are determined for the updated mesh and the optimum option is selected. This procedure needs to be repeated until a standard and nested T-spline mesh is obtained, see for example Figure 34 after six iterations.

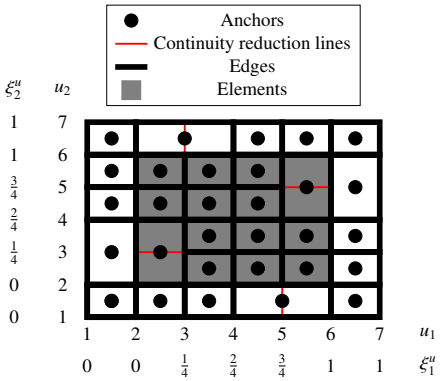


Figure 34. Standard and nested T-spline mesh after inserting additional anchors in six iterations into the refined T-spline mesh of Figure 32(b)

D. MODIFIED LOCAL KNOT VECTORS

Table III gives the local knot vector in element b and in the sub-elements b_l and b_r for each of the blue anchors in Figure 19.

DOI: 10.1002/nme

Table III. Local knot vectors Ξ_1 for the blue anchors in Figure 19 in element b and in the sub-elements b_l and b_r.

Coordinates in index domain	Local knot vector	Local knot vector in b_l	Local knot vector in b_r
$(1.5, 2.5)$	$\{0,0,0,1\}$	$\{0,0,0,\frac{1}{4}\}$	$\{0,0,\frac{1}{4},1\}$
$(3, 2.5)$	$\{0,0,1,1\}$	$\{0,0,\frac{1}{4},1\}$	$\{0,\frac{1}{4},1,1\}$
$(1.5, 4)$	$\{0,0,0,1\}$	$\{0,0,0,\frac{1}{4}\}$	$\{0,0,\frac{1}{4},1\}$
$(3, 4)$	$\{0,0,1,1\}$	$\{0,0,\frac{1}{4},1\}$	$\{0,\frac{1}{4},1,1\}$
$(4.5, 3.5)$	$\{0,1,1,1\}$	$\{0,\frac{1}{4},1,1\}$	$\{\frac{1}{4},1,1,1\}$
$(4.5, 4.5)$	$\{0,1,1,1\}$	$\{0,\frac{1}{4},1,1\}$	$\{\frac{1}{4},1,1,1\}$
$(1.5, 5.5)$	$\{0,0,0,\frac{1}{2}\}$	$\{0,0,0,\frac{1}{4}\}$	$\{0,0,\frac{1}{4},\frac{1}{2}\}$
$(2.5, 5.5)$	$\{0,0,\frac{1}{2},1\}$	$\{0,0,\frac{1}{4},\frac{1}{2}\}$	$\{0,\frac{1}{4},\frac{1}{2},1\}$
$(3.5, 5.5)$	$\{0,\frac{1}{2},1,1\}$	$\{0,\frac{1}{4},\frac{1}{2},1\}$	$\{\frac{1}{4},\frac{1}{2},1,1\}$

REFERENCES

1. Hughes TJR, Cottrell JA, Bazilevs Y. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering* 2005; **194**(39-41):4135–4195.
2. Sederberg TW, Zheng J, Bakenov A, Nasri A. T-splines and T-NURCCs. *ACM Transactions on Graphics* 2003; **22**(3):477–484.
3. Sederberg TW, Cardon DL, Finnigan GT, North NS, Zheng J, Lyche T. T-spline simplification and local refinement. *ACM Transactions on Graphics* 2004; **23**(3):276–283.
4. Dörfel MR, Jüttler B, Simeon B. Adaptive isogeometric analysis by local h-refinement with T-splines. *Computer Methods in Applied Mechanics and Engineering* 2010; **199**(5–8):264–275.
5. Bazilevs Y, Calo VM, Cottrell JA, Evans JA, Hughes TJR, Lipton S, Scott MA, Sederberg TW. Isogeometric analysis using T-splines. *Computer Methods in Applied Mechanics and Engineering* 2010; **199**(5–8):229–263.
6. Borden MJ, Scott MA, Evans JA, Hughes TJR. Isogeometric finite element data structures based on Bézier extraction of NURBS. *International Journal for Numerical Methods in Engineering* 2011; **87**(1–5):15–47.
7. Scott MA, Borden MJ, Verhoosel CV, Sederberg TW, Hughes TJR. Isogeometric finite element data structures based on Bézier extraction of T-splines. *International Journal for Numerical Methods in Engineering* 2011; **88**(2):126–156.
8. Buffa A, Cho D, Sangalli G. Linear independence of the T-spline blending functions associated with some particular T-meshes. *Computer Methods in Applied Mechanics and Engineering* 2010; **199**(23–24):1437–1445.
9. Li X, Zheng J, Sederberg TW, Hughes TJR, Scott MA. On linear independence of T-spline blending functions. *Computer Aided Geometric Design* 2012; **29**(1):63–76.
10. da Veiga LB, Buffa A, Cho D, Sangalli G. Analysis-suitable T-splines are dual-compatible. *Computer Methods in Applied Mechanics and Engineering* 2012; **249–252**:42–51.
11. da Veiga LB, Buffa A, Sangalli G, Vázquez R. Analysis-suitable T-splines of arbitrary degree: definition, linear independence and approximation properties. *Mathematical Models and Methods in Applied Sciences* 2013; **23**(11):1979–2003.
12. Li X, Scott MA. Analysis-suitable T-splines: characterization, refineability and approximation. *Mathematical Models and Methods in Applied Sciences* 2014; **24**(06):1141–1164.
13. Scott MA, Li X, Sederberg TW, Hughes TJR. Local refinement of analysis-suitable T-splines. *Computer Methods in Applied Mechanics and Engineering* 2012; **213–216**:206–222.
14. Evans EJ, Scott MA, Li X, Thomas DC. Hierarchical T-splines: analysis-suitability, Bézier extraction, and application as an adaptive basis for isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering* 2015; **284**:1–20.
15. Thomas DC, Scott MA, Evans JA, Tew K, Evans EJ. Bézier projection: a unified approach for local projection and quadrature-free refinement and coarsening of NURBS and T-splines with particular application to isogeometric design and analysis. *Computer Methods in Applied Mechanics and Engineering* 2015; **284**:55–105.
16. Wang A, Zhao G. An algorithm of determining T-spline classification. *Expert Systems with Applications* 2013; **40**(18):7280–7284.
17. Wang A, Zhao G, Li YD. Linear independence of the blending functions of T-splines without multiple knots. *Expert Systems with Applications* 2014; **41**(8):3634–3639.
18. Cox MG. The numerical evaluation of B-Splines. *IMA Journal of Applied Mathematics* 1972; **10**(2):134–149.
19. de Boor C. On calculating with B-splines. *Journal of Approximation Theory* 1972; **6**(1):50–62.
20. Piegl L, Tiller W. *The NURBS Book*. Springer, 1996.
21. Farouki RT, Neff CA. On the numerical condition of Bernstein-Bézier subdivision processes. *Mathematics of Computation* 1990; **55**(192):637–647.

DOI: 10.1002/nme