Construction of polynomial spline spaces over quadtree and octree T-meshes

M. Brovka J.I. López J.M. Escobar
J.M. Cascón, R. Montenegro

October, 2014

Abstract

We present a new strategy for constructing tensor product spline spaces over quadtree and octree T-meshes. The proposed technique includes some simple rules for inferring local knot vectors to define spline blending functions. These rules allows to obtain for a given T-mesh a set of cubic spline functions that span a space with nice properties: it can reproduce cubic polynomials, the functions are C^2-continuous, lineally independent, and spaces spanned by nested T-meshes are also nested. In order to span spaces with these properties applying the proposed rules, the T-mesh should fulfill the only requirement of being a *0-balanced* quadtree or octree. The straightforward implementation of the proposed strategy and the simplicity of tree structures can make it attractive for its use in geometric design and isogeometric analysis. In this paper we give a detailed description of our technique and illustrate some examples of its application in isogeometric analysis performing adaptive refinement for 2D and 3D problems.

1 Introduction

The main drawback of using B-splines and NURBS for geometric design is the impossibility to perform local refinement due to its tensor product structure. T-splines were introduced by Sederberg et al. [1] as an alternative to NURBS. Based on the idea of admitting meshes with T-junctions and inferring local knot vectors by traversing T-mesh edges, T-splines have provided a promising tool for geometric modeling that allows to perform local refinement without introducing a large number of superfluous control points. Later, in [2] T-splines were incorporated to the framework of isogeometric analysis. Isometric analysis (IGA) was introduced in 2005 by Hughes et al. in [3, 4]. It has arisen as an attempt to unify the fields of CAD and classical finite element methods. The main idea of IGA consists in using for analysis the same functions that are used in CAD representation of the geometry.

To use spline functions for numerical analysis and obtain a proper convergence behavior, these functions must meet some requirements: linear independence, polynomial reproduction property, local supports and possibility to perform local adaptive refinement. This issue was the object of numerous research works in recent years.

Analysis-suitable T-splines, proposed by Scott et al. in [5], is a class of T-splines defined over T-meshes that should meet certain topological restriction formulated in terms of T-junction extensions. Basis functions defined over an extended analysis-suitable T-mesh are lineally independent [6] and form partition of unity. The refinement algorithm allows to accomplish highly localized refinements and constructs nested T-spline spaces, but it presents an elevated implementation complexity and, as far we know, it is still an open question the generalization of the strategy to 3D cases.

Another approach to the problem of local enrichment of the approximation space is the hierarchical refinement, originally introduced by Forsey and Bartels in [7] and later developed in [8]. Recently, hierarchical refinement technique in the context of isogeometric analysis was described in [9, 10, 11]. This approach is based on a simple and natural idea to construct multilevel spaces by replacing coarse level functions with finer basis functions. Starting from an initial uniform mesh, hierarchical refinement scheme leads to sequential construction of nested spline spaces with linear independent basis functions. Simplicity of its implementation and straightforward generalization to 3D make it an attractive option for local refinement. However, a drawback of this strategy is the impossibility to define a spline space over a given arbitrary T-mesh as well as the presence of redundant basis functions due to the necessity to enlarge the refined area.

As another options for performing local refinement of spline spaces, C^1-continuous PHT-splines [12] or local refined splines (LR-splines) [13] can be used.

In the present paper we propose another possible alternative for the construction of spline functions that span spaces with nice properties. The main goal we pursue here is the simplicity and low computational cost of the implementation of our strategy, both in 2D and 3D. For that we have to assume a restriction on T-mesh type, namely, the technique we present here is designed for *0-balanced* quadtree and octree T-meshes. Tree data structure is frequently used in engineering due to its simplicity and we consider it an attractive tool for performing adaptive refinements. Balanced tree condition is usually imposed to have gradual transition from the coarse mesh to the finer zones and thus to guarantee a good quality of the approximation space constructed over the mesh. In addition, for our technique this condition is an obligatory prerequisite that the T-mesh should fulfill. Assuming this reasonable restriction over the T-mesh type allows us to define easily cubic spline functions that span spaces with desirable properties: linear independence, C^2-continuous, cubic polynomial reproduction property, nestedness of spanned spaces and straightforward implementation. The key of the strategy lies in some simple rules used for inferring the local knot vectors for each blending function.

The paper is organized as follows. General scheme of our strategy and the description of its main stages are given in the section 2. In the section 3 we explain in details the key of our technique, that is the rules used for the modification of the function supports in order to span spaces with desirable properties. Computational examples of performing adaptive refinement for 2D and 3D Poisson problems are presented in the Section 4. Conclusions are given in Section 5.

2 Strategy for construction of polynomial spline spaces over quadtree and octree T-meshes

In this section we describe our strategy to define bivariate (trivariate) spline functions over quadtree (octree) T-meshes. The strategy we propose has some similarity with T-splines insomuch as we define the blending functions from local knot vectors that are inferred by traversing the T-mesh edges. Some additional rules and requirements are imposed for the local knot vectors in order to obtain spline spaces with nice properties. The process of spline space construction for a given T-mesh can be divided in the following three steps:

1. *Mesh pretreatment (0-balancing)*
2. *Inferring local knot vectors*
3. *Modification of local knot vectors*

Next, we give a description of each step of the process.

2.1 Mesh pretreatment. 0-balanced quadtree and octree meshes.

Due to its simplicity, quadtree and octree are attractive tools for performing adaptive refinement in IGA and geometric modeling. To guarantee a good quality of the approximation space constructed over a mesh, it is preferable to have a gradual transition from the coarse mesh to the finely refined zone. That is why it is common to work with balanced quadtree and octree meshes. The strategy we propose in this paper is designed exclusively for the *0-balanced* T-meshes. A mesh with tree structure is said to be *0-balanced* if for any k, no cell at level k shares a vertex (0-face) with a cell at level greater than $k+1$. To obtain a *0-balanced* quadtree, a standard balancing procedure is applied. Note that the refinements performed during the *0-balancing* procedure do not propagate, see [14].

It should be highlighted that *0-balancing* the T-mesh is an essential prerequisite for the construction of spline spaces by means of our technique. In general, if the T-mesh is not *0-balanced*, our rules for inferring local knot vectors do not lead to polynomial spaces.

Imposing this mild restriction for T-meshes allows us to define easily spline spaces with the desirable properties.

2.2 Inferring local knot vectors

Let us consider a T-mesh T of the squared parametric domain $\Omega := [0,1]^d$. We call *regular node* the node of the mesh that is not a T-junction. We associate a blending function only to *regular nodes* of the mesh, as it is usual in classical finite element methods when working with hanging nodes. The *skeleton* of a d-dimensional mesh T is the geometric set of points composed of the union of all $(d-1)$-faces of the mesh and it is denoted by $\mathrm{skt}(T)$. That is, for a 2D space, the mesh *skeleton* is the union of all edges of the mesh and the *skeleton* of a 3D mesh is the union of all its faces.

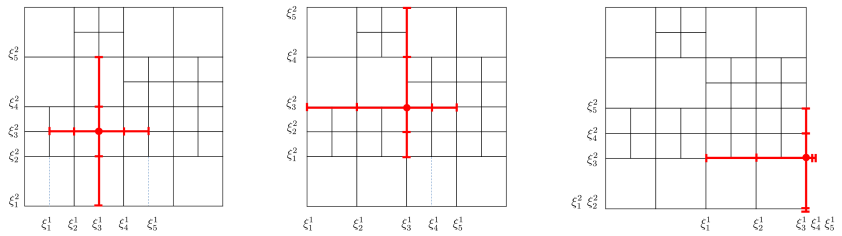


Figure 1: Inferring local knot vectors for a bivariate function by traversing the T-mesh edges. (a) No T-junction is skipped; (b) two T-junctions are skipped in each direction; (c) two T-junctions are skipped and the knots are repeated when the boundary is reached.

To define our cubic tensor product spline blending functions over a given d-dimensional T-mesh, a local knot vector for d parametric directions should be assigned to each function N_α: $\Xi^j_\alpha = \left(\xi^j_1, \xi^j_2, \xi^j_3, \xi^j_4, \xi^j_5\right)$, $j = 1, ..., d$. Similarly to [1], these knot vectors are inferred by traversing the T-mesh *skeleton*. For simplicity, let us describe this procedure for the two-dimensional T-mesh. Starting from the central knot (ξ^1_3, ξ^2_3), i.e. the anchor of the function, we walk across the T-mesh until intersect perpendicularly with a mesh edge. According to our strategy, we should skip over the T-junctions where the missing edge is perpendicular to the direction of our marching, see Fig. 1. When the boundary of the parametric domain is reached while walking across the mesh, we repeat knots creating an open knot vector structure along the boundary, see Fig. 1(c). Note that all interior knots have multiplicity 1. Thus, we obtain for the mesh T a set of blending functions $\{N_\alpha\}_{\alpha \in A_T}$, where A_T is the index set. Any interior *regular node* of the mesh has exactly one function associated to it and the boundary nodes have more than one function associated due to the open knot vector structure.

The process of inferring of local knot vectors can be resumed as follows:

- *Blending functions are associated only to regular nodes of the mesh.*
- *Local knot vectors are inferred by walking across the mesh until intersect the mesh skeleton. This intersection should not coincide with a T-junction perpendicular to the direction of our marching.*
- *Boundary knots are repeat to create an open knot vector structure along the boundary.*

Next, in order to span a spline space with good properties, some function supports should be modified. This issue is addressed in the next subsection.

2.3 Modification of local knot vectors

The key of our strategy lies in some simple rules used for the modification of the function supports that leads to the construction of a polynomial spline space over a given *0-balanced* T-mesh. In order to describe the idea, let us introduce

some notation. For the local knot vectors $\Xi_\alpha^j = \left(\xi_1^j, \xi_2^j, \xi_3^j, \xi_4^j, \xi_5^j\right),\ j = 1, ..., d$ let us denote the length of each knot interval as $\Delta_i^j = \xi_{i+1}^j - \xi_i^j,\ j = 1, ..., d$ and $i = 1, ..., 4$.

The support of a d-variate blending function N_α is a d-dimensional rectangular *box*: $[\xi_1^1, \xi_5^1] \times \cdots \times [\xi_1^d, \xi_5^d]$. We are going to call *frame* of a function support to the union of all $(d-2)$-faces of this *box* and we will denote it by $\mathrm{frm}(\mathrm{supp}\, N_\alpha)$. That is, for the rectangular support of a bivariate function, the *frame* is the union of the four vertices of this rectangle. For the cuboidal support of a trivariate function, its *frame* is composed of the union of the twelve edges of this cuboid.

The goal of the knot vector modification is to achieve that knot vectors $\Xi_\alpha^j,\ j = 1, ..., d$ of each blending function N_α fulfill the following conditions:

Condition 1: *Local knot vectors of the d-variate function* N_α *verify*[1]

$$\Delta_1^j \geqslant \Delta_2^j = \Delta_3^j \leqslant \Delta_4^j, \quad j = 1, ..., d, \tag{1}$$

Condition 2: *The frame of the function support should be situated over the mesh skeleton*:

$$\mathrm{frm}(\mathrm{supp}\, N_\alpha) \in \mathrm{skt}(T). \tag{2}$$

Thus, the function supports that do not meet the Conditions 1 or 2 should be modified. This modification consists in to extend the original knot vector by skipping over some knot intervals and changing some knots value until the resulting support satisfies both conditions.

In the next section we give a detailed description of this procedure for 2D and 3D cases.

3 Support modification

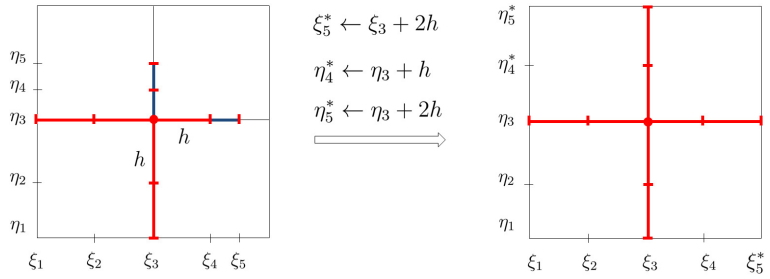
Here, we present our strategy based on some simple support extension rules to obtain local knot vectors that fulfill the Condition 1 and 2 formulated in the previous section. We proceed as follows. First, if after traversing the T-mesh *skeleton* the local knot vectors of a function do not satisfy Condition 1, we modify some of their knots in order to meet the Condition 1. Then, the fulfillment of the Condition 2 is checked and, if it is not satisfied, another appropriate modification of the support is carried out. As result of these modifications we obtain a new extended support of a function with local knot vectors which verify both conditions. These modifications are easily implemented taking into account the balanced tree structure of the mesh. Let see in details this procedure.

To simplify the notation, in the rest of the paper we denote the parametric coordinates as (ξ, η, ζ) and it is related to the previous notation as $(\xi^1, \xi^2, \xi^3) = (\xi, \eta, \zeta)$. Consequently, $(\Xi^1, \Xi^2, \Xi^3) = (\Xi, \mathcal{H}, \mathcal{Z})$ and $(\Delta_i^1, \Delta_i^2, \Delta_i^3)$
$= (\Delta_i^\xi, \Delta_i^\eta, \Delta_i^\zeta)$.

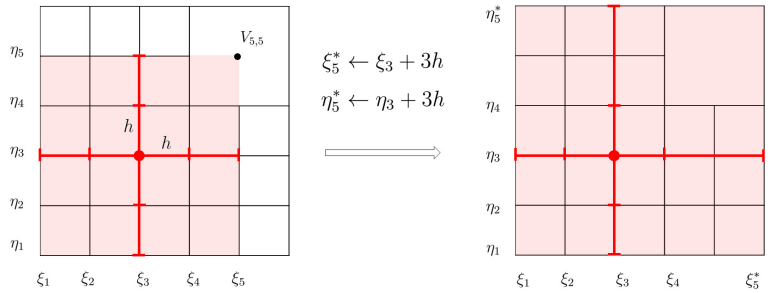
3.1 Support modification for 2D meshes

In order to facilitate the description and illustration of the strategy, some concepts and notation introduced in section 2 have to be particularized to the 2D

[1] Except the cases involving repeated knots that are explained at the end of the section 3.1.



(a) Condition 1 is not satisfied because $\Delta_4^\xi < \Delta_3^\xi$ and $\Delta_3^\eta < \Delta_2^\eta$.



(b) Condition 2 is not satisfied because $V_{5,5} \notin skt(T)$.

Figure 2: Support modification according to the Condition 1 and 2. (a) An example of modification of the local knot vectors of a bivariate function to satisfy the Condition 1; (b) an example of modification of the local knot vectors to satisfy the Condition 2.

case. The *skeleton* $\mathrm{skt}(T)$ of a two-dimensional mesh T is the union of all edges of the mesh. For a bivariate function let us denote the vertices of its rectangular support as $V_{1,1} = (\xi_1, \eta_1)$, $V_{5,1} = (\xi_5, \eta_1)$, $V_{5,5} = (\xi_5, \eta_5)$ and $V_{1,5} = (\xi_1, \eta_5)$. Then, the *frame* of a function support is the union of its four vertices, i.e., $\mathrm{frm}(\operatorname{supp} N_\alpha) = \{V_{n,m},\ n, m \in \{1, 5\}\}$.

Condition 1 for the local knot vectors Ξ and $\mathcal{H}$ of a bivariate function is trivial and does not need any clarification. The Condition 2 adapted to 2D case is formulated as follows: *The four vertices of a function support should be situated over the mesh edges.*

If, for example, the local knot vector Ξ of a function does not satisfy the Condition 1, we modify this vector by skipping over the minimal number of knots until $\Delta_1^\xi \geqslant \Delta_2^\xi = \Delta_3^\xi \leqslant \Delta_4^\xi$ is verified, and analogously, for $\mathcal{H}$. This modification is made independently for each parametric direction applying certain simple extension rules. Let see an example of support extension for a bivariate function. Leftmost function support shown in Fig. 2(a) does not meet the Condition 1. For the knot vector Ξ we have $\Delta_3^\xi > \Delta_4^\xi$, so the knot interval Δ_4^ξ should be modified. Let us denote $h = \max(\Delta_2^\xi, \Delta_3^\xi) = \max(\Delta_2^\eta, \Delta_3^\eta)$. Note that both maxima coincide due to the quadtree structure and the fact that the T-junctions are skipped. Then, the fifth knot ξ_5 is redefined as $\xi_5^* \leftarrow \xi_3 + 2h$. For the local knot vector $\mathcal{H}$ we have $\Delta_2^\eta > \Delta_3^\eta$, so the knots η_4 and η_5 should be modified as $\eta_4^* \leftarrow \eta_3 + h$, $\eta_5^* \leftarrow \eta_3 + 2h$.

Once the Condition 1 is satisfied, in order to fulfill the Condition 2, we check weather the vertices of the function support are situated over the mesh edges. If not, we modify the knot vectors by skipping over a knot for both parametric directions and placing this vertex over the mesh edges. Figure 2(b) illustrates this procedure. The checking is performed independently for each of the four quadrants of the function support. Note that, for our *0-balanced* quadtree, we should make this checking only for some functions. For example, without loss of generality, the support vertex $V_{5,5} = (\xi_5, \eta_5)$ must be checked only if $\Delta_3^\xi = \Delta_4^\xi = \Delta_3^\eta = \Delta_4^\eta$. An example of a function support violating the Condition 2 is illustrated in Fig. 2(b). The vertex $V_{5,5} = (\xi_5, \eta_5)$ of this support is not situated over a mesh edge, so the fifth knots for both parametric directions should be redefined as $\xi_5^* \leftarrow \xi_3 + 3h$, $\eta_5^* \leftarrow \eta_3 + 3h$, and thus, the new vertex $V_{5,5}$ is placed over the mesh edges.

The extension of any other function support is completely analogous to these two examples. In all possible cases, the extension of a function support implies to change one or two knot intervals by duplicating its size.

A detailed algorithm for the extension rules used to modify a bivariate function support according to the Condition 1 and 2 are given in Algorithms 1 and 2.

Figure 3 shows some examples of support modifications. Functions that satisfy both conditions and should not be modified are given in Fig. 3(a). Examples of support modifications according to the Condition 1 are shown in Fig. 3(b), (c), (d) and (e). And Fig. 3(f), (g) and (h) illustrate support modifications according to the Condition 2 or both.

Note that an exception for the Condition 1 is a knot vector that contains knot interval of length 0 due to the open knot vector structure along the boundary. In this case, a knot vector should fulfill the inequality (1) not taking into account the knot intervals of length 0. Consequently, an exception for the application of the extension rules is the case when the boundary of the parametric domain is reached traversing the T-mesh edges, see Fig. 3(c), (d) and (e).

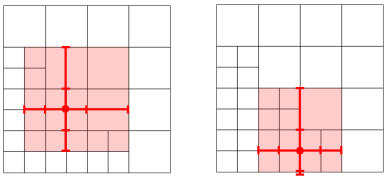
3.2 Support modification for 3D meshes

In this section we give a description and illustration of the proposed strategy for defining trivariate spline functions over *0-balanced* octree T-meshes.

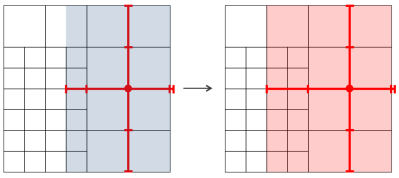
The *skeleton* $\mathrm{skt}(T)$ of a three-dimensional mesh T is the union of all faces of the mesh. For a trivariate function let us denote the vertices of its support as $V_{n,m,k} = (\xi_n, \eta_m, \zeta_k)$ where $n, m, k \in \{1, 5\}$. And the edge of a support formed by the two vertices $V_{n,m,k}$ and $V_{p,q,r}$ is denoted as $E_{(n,m,k),(p,q,r)}$. Then, the *frame* $\mathrm{frm}(\operatorname{supp} N_\alpha)$ of a trivariate function support is the union of its twelve edges.

The formulation of the Condition 1 for the local knot vectors of a trivariate function is analogues to 2D case. The Condition 2 adapted to 3D meshes is stated as follows: *Edges of the cuboidal function support should be situated over the mesh faces.*

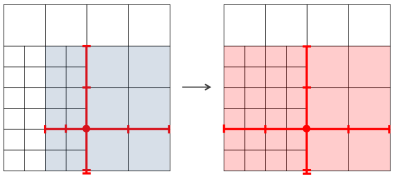
The implementation of the strategy for 3D is similar to 2D case. To satisfy the Condition 1, extension rules are applied to each of three local knot vectors of a function analogously to 2D case using the Algorithms 1. In order to fulfill the Condition 2, we check weather the edges of a function support are situated over the mesh faces. If not, the two knot vectors perpendicular to this edge



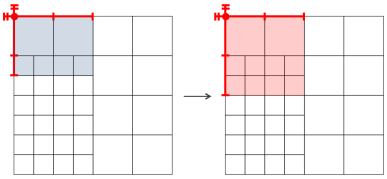
(a) Initial supports that satisfy both conditions and should not be modified.



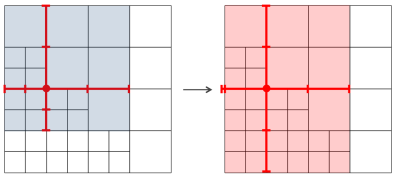
(b) Condition 1 is not satisfied because $\Delta_1^\xi < \Delta_2^\xi$, so $\xi_1^* \leftarrow \xi_3 - 2h$.



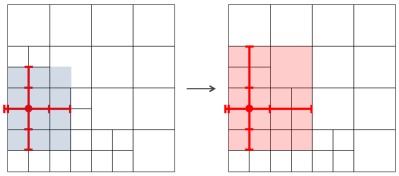
(c) Condition 1 is not satisfied because $\Delta_2^\xi < \Delta_3^\xi$, so $\xi_1^* \leftarrow \xi_3 - 2h$ and $\xi_2^* \leftarrow \xi_3 - h$.



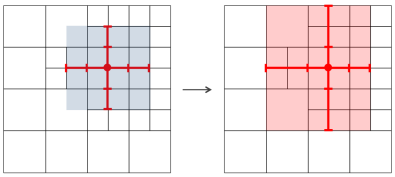
(d) Condition 1 is not satisfied because $\Delta_1^\eta < \Delta_2^\eta$, so $\eta_1^* \leftarrow \eta_3 - 2h$.



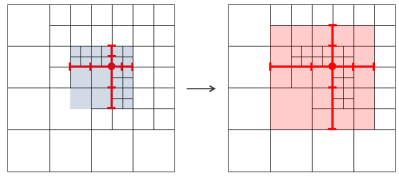
(e) Condition 1 is not satisfied because $\Delta_2^\xi < \Delta_3^\xi$ and $\Delta_2^\eta < \Delta_3^\eta$, so $\xi_2^* \leftarrow \xi_3 - h$, $\eta_2^* \leftarrow \eta_3 - h$ and $\eta_1^* \leftarrow \eta_3 - 2h$.



(f) Condition 2 is not satisfied because $V_{5,5} \notin skt(T)$, so $\xi_5^* \leftarrow \xi_3 + 3h$ and $\eta_5^* \leftarrow \eta_3 + 3h$.



(g) Condition 2 is not satisfied because $V_{1,1}$ and $V_{1,5} \notin skt(T)$, so $\xi_1^* \leftarrow \xi_3 - 3h$, $\eta_1^* \leftarrow \eta_3 - 3h$ and $\eta_5^* \leftarrow \eta_3 + 3h$.



(h) Condition 1 and 2 are not satisfied because $\Delta_2^\xi > \Delta_3^\xi$, $\Delta_2^\eta > \Delta_3^\eta$ and $V_{1,1} \notin skt(T)$, so $\xi_4^* \leftarrow \xi_3 + h$, $\xi_5^* \leftarrow \xi_3 + 2h$, $\eta_4^* \leftarrow \eta_3 + h$, $\eta_5^* \leftarrow \eta_3 + 2h$, $\xi_1^* \leftarrow \xi_3 - 3h$, $\eta_1^* \leftarrow \eta_3 - 3h$.

Figure 3: Examples of function support modifications to fulfill the imposed Conditions 1 and 2. Initial support is marked in blue and corrected support in red.

Algorithm 1: Knot vector modifications to fulfill the Condition 1.

Input: A knot vector $\Xi = (\xi_1, \xi_2, \xi_3, \xi_4, \xi_5)$, $\xi_i \in [0, 1]$.

Function ***Modify1***(Ξ)
- $\Xi^* \leftarrow \Xi$
- $h = \max(\Delta_2, \Delta_3)$
- **if** $\Delta_2 < \Delta_3$ *and* $\xi_2 > 0$ **then**
  - $\xi_2^* \leftarrow \xi_3 - h$
  - $\xi_1^* \leftarrow \xi_2^*$
  - **if** $\xi_1^* > 0$ **then** $\xi_1^* \leftarrow \xi_3 - 2h$
- **if** $\Delta_1 < \Delta_2$ *and* $\xi_1 > 0$ **then**
  - $\xi_1^* \leftarrow \xi_3 - 2h$
- **if** $\Delta_2 > \Delta_3$ *and* $\xi_4 < 1$ **then**
  - $\xi_4^* \leftarrow \xi_3 + h$
  - $\xi_5^* \leftarrow \xi_4^*$
  - **if** $\xi_5^* < 1$ **then** $\xi_5^* \leftarrow \xi_3 + 2h$
- **if** $\Delta_4 < \Delta_3$ *and* $\xi_5 < 1$ **then**
  - $\xi_5^* \leftarrow \xi_3 + 2h$
- **return** Ξ^*

Output: A corrected knot vector Ξ^* that satisfies the Condition 1.

Algorithm 2: Support modification to fulfill the Condition 2 in 2D.

Input: A *0-balanced* mesh T and a pair of local knot vectors $S = \{\Xi, \mathcal{H}\}$.

Function ***Modify2***$(T,\ S)$
- $S^* \leftarrow S$
- $h = \max(\Delta_2^\xi, \Delta_3^\xi)$
- **for** $n \in \{1, 5\}$ **do**
  - **for** $m \in \{1, 5\}$ **do**
    - **if** $(\xi_n, \eta_m) \notin skt(T)$ **then**
      - $\xi_n^* \leftarrow \xi_3 + 3h\,\mathrm{sgn}(\xi_n - \xi_3)$
      - $\eta_m^* \leftarrow \eta_3 + 3h\,\mathrm{sgn}(\eta_m - \eta_3)$
- **return** S^*

Output: A modified support $S^* = \{\Xi^*, \mathcal{H}^*\}$ that satisfies the Condition 2.

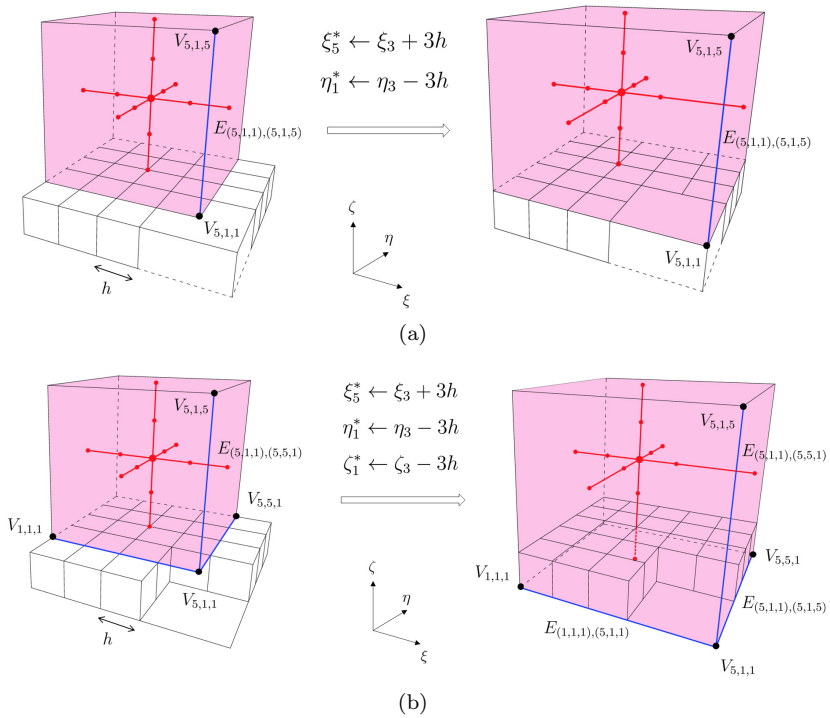


Figure 4: Support modification of a trivariate function according to the Condition 2. (a) Only one edge (in blue) violating the Condition 2. The node $V_{5,1,1}$ is sited in the center of the face of size $2h$. The support is extended in two directions; (b) three edges violating Condition 2. The node $V_{5,1,1}$ is sited in the center of the cell of size $2h$. The support is extended in three directions.

should be modified by skipping over a knot for both parametric direction and placing this edge over the mesh faces. This checking is performed independently for each of the eight quadrants of the function support and, in each quadrant, three edges should be checked.

Figure 4 illustrates the support extension procedure for the quadrant of the vertex $V_{5,1,1}$. Due to the octree structure only two cases can take place: (i) the quadrant contains one edge that is not situated over the mesh faces or (ii) the quadrant contains three edges and a vertex that are not situated over the mesh *skeleton*. We now study each case.

(i) If a quadrant contains one edge that does not fulfill the Condition 2, then two knot vectors perpendicular to this edge are modified, see Fig. 4(a). For the function support shown in Fig. 4(a) left, the edge $E_{(5,1,1),(5,1,5)}$ is not situated over the mesh faces. Therefore its two knot vectors Ξ and $\mathcal{H}$ perpendicular to this edge are modified in order to place the edge over the mesh faces, namely, the knots ξ_5 and η_1 are redefined as $\xi_5^* \leftarrow \xi_3 + 3h$ and $\eta_1^* \leftarrow \eta_3 - 3h$, where $h = \max(\Delta_2^\xi, \Delta_3^\xi) = \max(\Delta_2^\eta, \Delta_3^\eta) = \max(\Delta_2^\zeta, \Delta_3^\zeta)$.

(ii) If a quadrant includes three edges that are not situated over the mesh faces, then the three knot vectors are modified by skipping over a knot for each of the three parametric directions, see Fig. 4(b). The vertex $V_{5,1,1}$ and the three edges connected to it are not situated over the mesh *skeleton*, so all three knot vectors are modified to place the three edges over the mesh faces: $\xi_5^* \leftarrow \xi_3 + 3h,\ \eta_1^* \leftarrow \eta_3 - 3h,\ \zeta_1^* \leftarrow \zeta_3 - 3h$.

Algorithm 3 explains the extension rule used to modify a trivariate function support according to the Condition 2.

Algorithm 3: Support correction to fulfill the Condition 2 in 3D.

Input: A *0-balanced* T-mesh T and three local knot vectors $S = \{\Xi, \mathcal{H}, \mathcal{Z}\}$.

Function ***Modify2*** $(T,\ S)$
- $S^* \leftarrow S$
- $h = \max(\Delta_2^\xi, \Delta_3^\xi)$
- $mov(i) := i + 4\,\mathrm{sgn}(3 - i)$
- **for** $n \in \{1, 5\}$ **do**
  - **for** $m \in \{1, 5\}$ **do**
    - **for** $k \in \{1, 5\}$ **do**
      - **if** $E_{(n,m,k),(mov(n),m,k)} \notin skt(T)$ **then**
        - $\eta_m^* \leftarrow\ \eta_3 + 3h\,\mathrm{sgn}(\eta_m - \eta_3)$
        - $\zeta_k^* \leftarrow\ \zeta_3 + 3h\,\mathrm{sgn}(\zeta_k - \zeta_3)$
      - **if** $E_{(n,m,k),(n,mov(m),k)} \notin skt(T)$ **then**
        - $\xi_n^* \leftarrow\ \xi_3 + 3h\,\mathrm{sgn}(\xi_n - \xi_3)$
        - $\zeta_k^* \leftarrow\ \zeta_3 + 3h\,\mathrm{sgn}(\zeta_k - \zeta_3)$
      - **if** $E_{(n,m,k),(n,m,mov(k))} \notin skt(T)$ **then**
        - $\xi_n^* \leftarrow\ \xi_3 + 3h\,\mathrm{sgn}(\xi_n - \xi_3)$
        - $\eta_m^* \leftarrow\ \eta_3 + 3h\,\mathrm{sgn}(\eta_m - \eta_3)$
- **return** S^*

Output: A modified support $S^* = \{\Xi^*, \mathcal{H}^*, \mathcal{Z}^*\}$ that satisfies the Condition 2.

3.3 Properties

Here, we summarize the properties of the spaces constructed by means of our method.

For any *0-balanced* mesh T a set of blending functions defined according to our strategy spans a space $S_T := \operatorname{span}\{N_\alpha : \alpha \in A_T\}$ with the following properties:

- *Functions* $\{N_\alpha\}_{\alpha \in A_T}$ *are* C^2*-continuous.*
- *Functions* $\{N_\alpha\}_{\alpha \in A_T}$ *are lineally independent.*
- *Polynomial reproduction property:* $\mathbb{P}_3(\Omega) \in S_T$*.*
- *Spaces spanned by nested T-meshes are also nested:* $T_1 \subset T_2 \;\Rightarrow\; S_{T_1} \subset S_{T_2}$*.*

Unfortunately, for now, we can not provide a rigorous theoretical framework for our strategy. We plan to do this in our future work. We have carried out numerical experiments for an enormous number of different scenarios of mesh refinements and we have verified for all of them the nestedness of the spaces and the linear independence.

4 Computational examples

In this section we present computational examples of the application of our technique in problems involving adaptive refinement. The proposed method is tested by resolution of 2D and 3D Poisson problems using IGA.

4.1 Poisson problem on a complex domain

Here, we present the result of the resolution of a Poisson problem on a complex domain using isogeometric analysis. Let us consider the problem

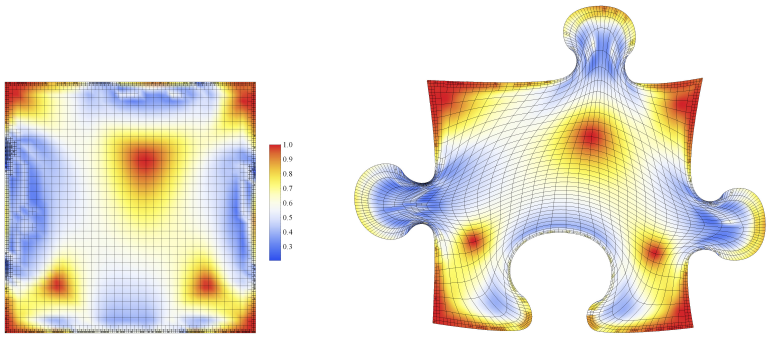
$$\begin{aligned} -\triangle u &= f \qquad \text{in } \Omega, \\ u &= g \qquad \text{on } \partial\Omega. \end{aligned} \tag{3}$$

The problem is set up in such a way that the analytical solution is a function with steep wave front given by

$$u(r) = \arctan(\alpha(r - r_0)),$$

where $r = \sqrt{(x - x_c)^2 + (y - y_c)^2}$, the parameter α determines the steepness of the wave front and r_0 is its location. In this example $\alpha = 200$ and $r_0 = 0.6$. The center of the wave front $(x_c, y_c) = (0, 0)$ is situated outside our computational domain, so the function is smooth in Ω.

The parameterization of the computational domain is performed using the algorithm described in our previous work [15]. This technique, based on a T-mesh untangling and optimization procedure, allows us to obtain a good



(a) (b)

Figure 5: Parameterization of the computational domain for the problem of the section 4.1. (a) Colormap of the mean ratio Jacobian represented in the parametric domain; (b) colormap of the mean ratio Jacobian represented in the physical domain.

quality parameterization suitable for application of IGA. The procedure is an extension of the ideas presented in our works [16, 17, 18].

The mean ratio Jacobian is used to evaluate the quality of the parameterization in the sense of its orthogonality and uniformity. Figures 5(a) and (b) show the colormap of the parameterization quality for our parametric and physical domains. It should be mentioned that for all numerical examples the parameterization of the computational domain is performed using the same blending functions that are used for the solution approximation, so isoparametric concepts holds during isogeometric analysis.

We perform an adaptive refinement based on a posteriori error indicator to improve the quality of the numerical solution. We have chosen a simple residual-type error estimator given by

$$\eta(\Omega_e)^2 = \|h\,(f + \Delta u_h)\,\|^2_{0,\Omega_e} = \int_{\Omega_e} h^2\,(f + \Delta u_h)^2\,\mathrm{d}\Omega,$$

where h is the diameter of the cell Ω_e. The estimator is jump free because of the smoothness of the isogeometric approximation. A cell Ω_e is marked to be refined if $\eta(\Omega_e) > \gamma\,\max_i\{\eta(\Omega_i)\}$, being $\gamma \in [0, 1]$.

The numerical solution of the problem and the mesh corresponding to the final refinement iteration is shown in Fig. 6. As expected, the error estimator have marked to refine in the zone of the wave front. The evolution of the exact error in L^2-norm and H^1-seminorm are shown in Fig. 7.

4.2 Poisson problem on 3D domain

The next computational example is the resolution of a 3D Poisson problem using IGA. The computational domain is a spline approximation of a sphere portion, see Fig. 8. The initial uniform mesh was composed by $4 \times 4 \times 4$ cells. The approximation is constructed using our spline blending function. The Poisson problem with Dirichlet boundary condition is set up so that the

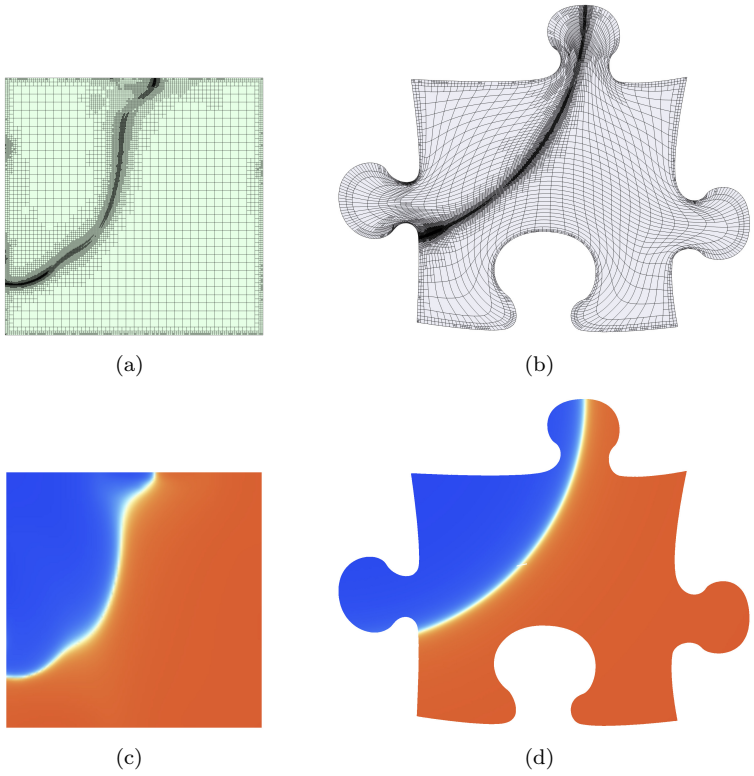


Figure 6: Results of the adaptive refinement for the Poisson problem of the section 4.1. (a) Final refinement in the parametric domain; (b) final refinement in the physical domain; (c) numerical solution in the parametric domain; (d) numerical solution in the physical domain.

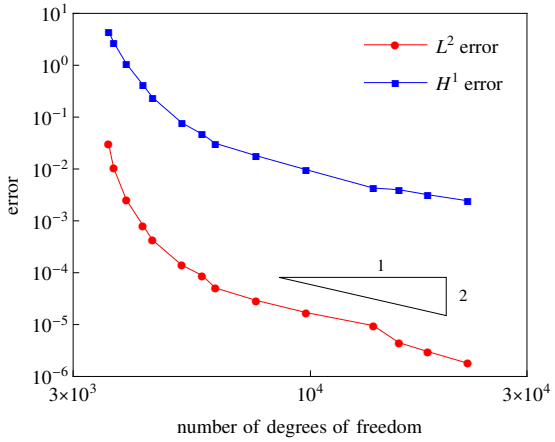


Figure 7: Convergence of L^2-norm and H^1-seminorm error for the Poisson problem of the section 4.1, for $\gamma = 0.1$.

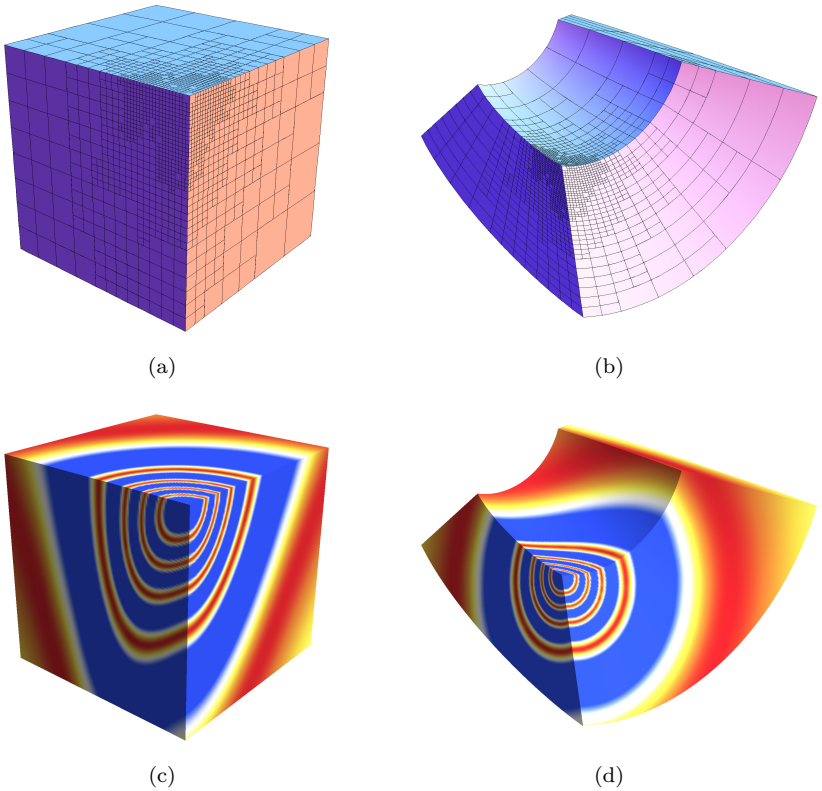


Figure 8: Mesh and numerical solution in the sixth adaptive refinement step for the 3D Poisson problem of the section 4.2. (a) Parametric mesh; (b) physical mesh; (c) numerical solution in the parametric domain; (d) numerical solution in the physical domain.

analytical solution of the problem is

$$u(r) = \sin\left(\frac{1}{\alpha + r}\right),$$

where $r = \sqrt{x^2 + y^2}$ and parameter $\alpha = 1/10\pi$. This is a smooth function with an oscillation near the origin. Results of the final refinement iteration are shown in Fig. 8. The convergence behavior of the adaptive refinement is illustrated in Fig. 9.

5 Conclusions and future research

In this paper we have proposed a strategy for defining tensor product spline spaces over quadtree and octree T-meshes. We only demand these T-meshes to be *0-balanced* and this requirement can be easily satisfied by using a standard balancing procedure. The proposed strategy includes simple instructions used for inferring local knot vectors to define blending functions. The resulting

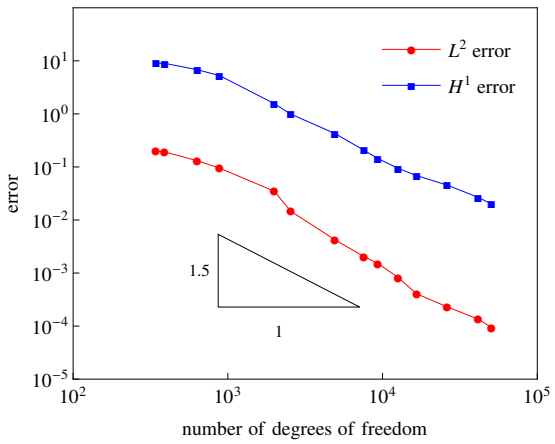


Figure 9: Convergence of L^2-norm and H^1-seminorm error for the Poisson problem of the section 4.2, for $\gamma = 0.1$.

spline spaces have good properties: lineal independence, C^2-continuity, ability for reproducing cubic polynomials and the characteristic that spaces spanned by nested T-meshes are also nested. The above mentioned properties were verified in numerical experiments, but we can not provide for now a complete theoretical framework for our method. We plan to do that in our future works.

The implementation of our technique is straightforward taking into account the balanced tree structure of the mesh. Examples of adaptive refinement using IGA for 2D and 3D Poisson problems have been presented. In all of them, optimal rates of convergence have been obtained. We believe that the simplicity of our technique can make it an attractive tool for its application in IGA and geometric design.

Acknowledgements

This work has been supported by the Spanish Government, "Secretaría de Estado de Universidades e Investigación," "Ministerio de Economía y Competitividad,", "Programa de FPU del Ministerio de Educación, Cultura y Deporte", "Programa de FPI propio de la ULPGC" and FEDER, grant contracts: CGL2011-29396-C03-01 and CGL2011-29396-C03-03; "Junta Castilla León," grant contract: SA266A12-2. It has been also supported by CONACYT-SENER ("Fondo Sectorial CONACYT SENER HIDROCARBUROS," grant contract: 163723).

References

[1] T. W. Sederberg, J. Zheng, A. Bakenov, A. Nasri, T-splines and T-NURCCSs, ACM Trans. Graph. 22 (2003) 477–484.

[2] Y. Bazilevs, V. M. Calo, J. A. Cottrell, J. A. Evans, T. J. R. Hughes,

S. Lipton, M. A. Scott, T. W. Sederberg, Isogeometric analysis using T-splines, Comput. Meth. Appl. Mech. Eng. 199 (2010) 229–263.

[3] T. J. R. Hughes, J. A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Comput. Meth. Appl. Mech. Eng. 194 (2005) 4135–4195.

[4] J. A. Cottrell, T. J. R. Hughes, Y. Bazilevs, Isogeometric analysis: Toward integration of CAD and FEA, John Wiley & Sons, Chichester, 2009.

[5] M. A. Scott, X. Li, T. W. Sederberg, T. J. R. Hughes, Local refinement of analysis-suitable T-splines, Comput. Meth. Appl. Mech. Eng. 213-216 (2012) 206–222.

[6] X. Li, J. Zheng, T. W. Sederberg, T. J. R. Hughes, M. A. Scott, On linear independence of T-spline blending functions, Comput. Aid. Geom. Design 29 (2012) 63–76.

[7] D. R. Forsey, R. H. Bartels, Hierarchical B-spline refinement, Computer Graphics 22(4) (1988) 205–212.

[8] R. Kraft, Surface Fitting and Multiresolution Methods, volume 2, Vanderbilt University Press, 1997, pp. 209–218.

[9] A. V. Vuong, C. Giannelli, B. Jüttler, B. Simeon, A hierarchical approach to adaptive local refinement in isogeometric analysis, Comput. Meth. Appl. Mech. Eng. 200 (2011) 3554–3567.

[10] D. Schillinger, L. Debé, M. Scott, J. A. Evans, M. J. Borden, E. Rank, T. J. R. Hughes, An isogeometric design-through-analysis methodology based on adaptive hierarchical refinement of nurbs, immersed boundary methods, and T-spline CAD surfaces, Comput. Meth. Appl. Mech. Eng. 249–252 (2012) 116–150.

[11] P. B. Bornemann, F. Cirak, A subdivision-based implementation of the hierarchical b-spline finite element method, Comput. Meth. Appl. Mech. Eng. 253 (2013) 584–598.

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

[13] T. Dokken, T. Lyche, K. F. Pettersen, Polynomial splines over locally refined box-partitions, Comput. Aid. Geom. Design 30(3) (2013) 331–356.

[14] D. Moore, The cost of balancing generalized quadtrees, in: Proceedings of the Third ACM Symposium on Solid Modeling and Applications, SMA '95, ACM, New York, NY, USA, 1995, pp. 305–312. doi:10.1145/218013.218078.

[15] M. Brovka, J. I. López, J. M. Escobar, J. M. Cascón, R. Montenegro, A new method for T-spline parameterization of complex 2D geometries, Engineering with Computers (2013) 1–17, published online. doi:10.1007/s00366–013–0336–8.

[16] J. M. Escobar, J. M. Cascón, E. Rodríguez, R. Montenegro, A new approach to solid modeling with trivariate T-splines based on mesh optimization, Comput. Meth. Appl. Mech. Eng. 200 (2011) 3210–3222.

[17] J. M. Escobar, R. Montenegro, E. Rodríguez, J. M. Cascón, The meccano method for isogeometric solid modeling and applications, Engineering with Computers 30 (2014) 331–343.

[18] J. M. Cascón, E. Rodríguez, J. M. Escobar, R. Montenegro, Comparison of the meccano method with standard mesh generation techniques, Engineering with Computers (2013) 1–14, published online. doi:10.1007/s00366–013–0338–6.