Comput Mech (2012) 50:65–84
DOI 10.1007/s00466-011-0674-6

ORIGINAL PAPER

Converting an unstructured quadrilateral/hexahedral mesh to a rational T-spline

Wenyan Wang · Yongjie Zhang · Guoliang Xu · Thomas J. R. Hughes

Received: 4 August 2011 / Accepted: 12 December 2011 / Published online: 27 December 2011

Abstract This paper presents a novel method for converting any unstructured quadrilateral or hexahedral mesh to a generalized T-spline surface or solid T-spline, based on the rational T-spline basis functions. Our conversion algorithm consists of two stages: the topology stage and the geometry stage. In the topology stage, the input quadrilateral or hexahedral mesh is taken as the initial T-mesh. To construct a gap-free T-spline, templates are designed for each type of node and applied to elements in the input mesh. In the geometry stage, an efficient surface fitting technique is developed to improve the surface accuracy with sharp feature preservation. The constructed T-spline surface and solid T-spline interpolate every boundary node in the input mesh, with C^2-continuity everywhere except the local region around irregular nodes. Finally, a Bézier extraction technique is developed and linear independence of the constructed T-splines is studied to facilitate T-spline based isogeometric analysis.

Keywords Quadrilateral mesh · Hexahedral mesh · Rational T-spline · Solid T-spline · Bézier extraction · Linear independence · Isogeometric analysis

W. Wang · Y. Zhang (✉)
Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA
e-mail: jessicaz@andrew.cmu.edu

G. Xu
Institute of Computational Mathematics, Academy of Mathematics and System Sciences, Chinese Academy of Sciences, Beijing 100190, China

T. J. R. Hughes
Institute for Computational Engineering and Sciences, The University of Texas at Austin, Austin, TX 78712, USA

1 Introduction

For the sake of integration of engineering design and analysis, isogeometric analysis was proposed [4,1] which utilizes NURBS (Non-Uniform Rational B-Spline) or T-splines as a basis. Over the 40 year history of commercial finite element analysis, there are a large number of polygonal meshes accumulated with the development of automatic mesh generation techniques and finite element analysis technology based on polygonal meshes. For example, Figure 1a shows an unstructured hexahedral mesh of a gear assembly. Hence, a solution to converting these polygonal meshes to T-splines is needed, which provides engineers with the opportunity to transition legacy bilinear quadrilateral surface meshes or trilinear hexahedral meshes to T-splines, analyze them using isogeometric analysis and compare the results with traditional finite element technology. In addition, the T-spline representation is a more compact way to represent geometry compared with polygonal meshes and has better continuity.

Previous approaches on converting meshes to spline representations involved approximating the data by determining the topology and choosing a parameterization. In [5,6], a conversion method from a triangle mesh of arbitrary topology into a T-spline surface was proposed based on periodic global parameterization. A polycube map, which mimics the input mesh in a topologically correct and geometrically meaningful manner, was utilized as parametric domain to construct T-splines [14].

In our earlier work, we developed an algorithm for converting any unstructured quadrilateral mesh to a standard T-spline, whose basis functions form a partition of unity [15]. In this method, many nodes need to be inserted in order to make the T-spline standard. To reduce the number of inserted nodes, as a follow-up we generalize the T-spline definition to the rational T-spline in this paper. The new rational T-spline

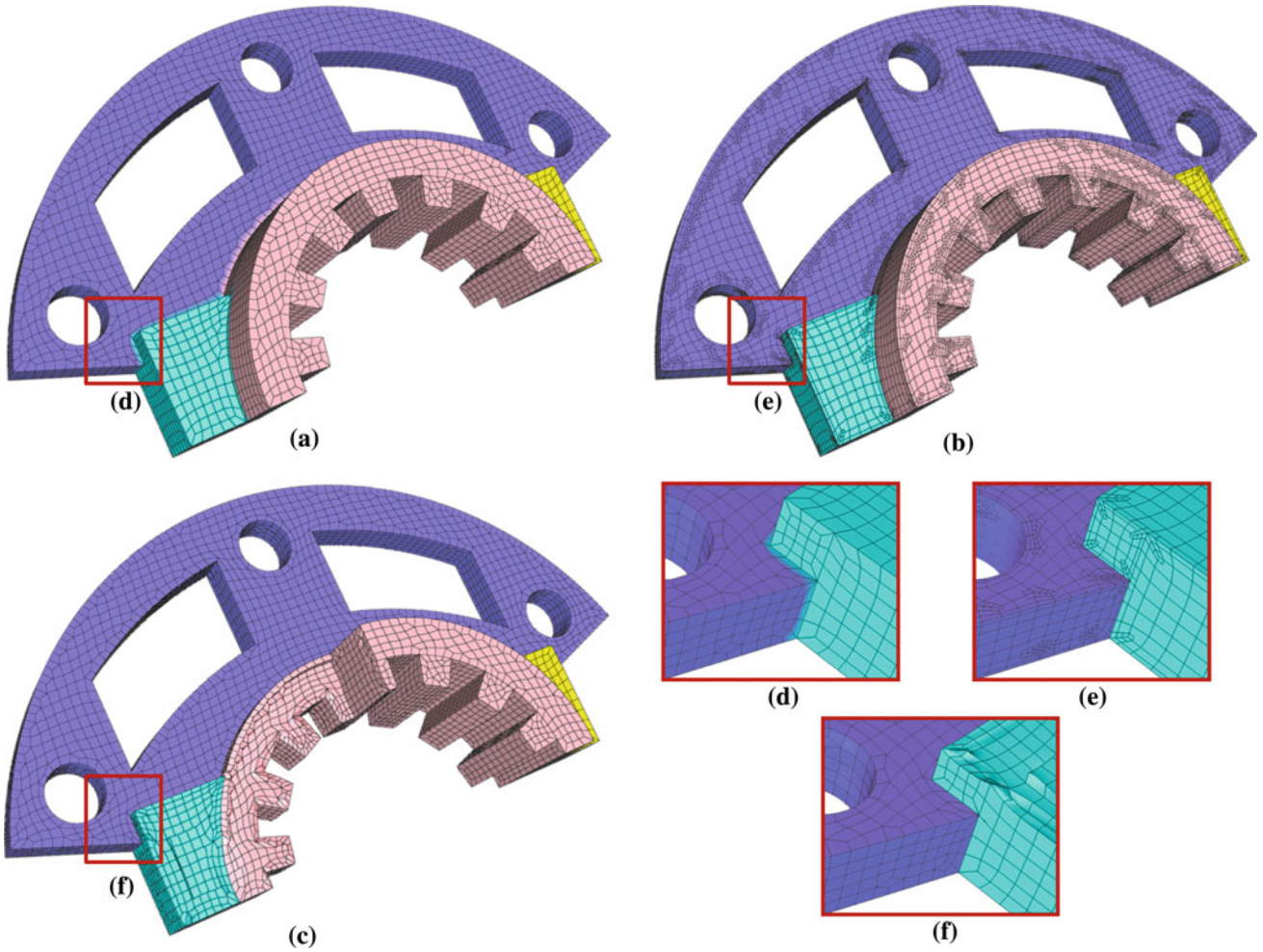


Fig. 1 The gear assembly. **a** The input unstructured hexahedral mesh; **b** the constructed solid T-spline and T-mesh; and **c** the extracted solid Bézier elements with some elements removed to show the interior mesh. **d–f** show details

basis functions have the property of a partition of unity not only for standard T-splines, but also for semi-standard and non-standard T-splines. For semi-standard and non-standard T-splines, basis functions do not satisfy a partition of unity based on the traditional T-spline definition. Here, we focus on converting an arbitrary unstructured quadrilateral or hexahedral mesh to a rational bicubic T-spline surface or tricubic solid T-spline. There are two main stages in the conversion algorithm: the topology stage and the geometry stage. We take the input mesh directly as the initial T-mesh, and the topology stage aims to make the initial T-mesh gap-free by designing templates for each type of node and applying them to elements. In the geometry stage, an efficient surface fitting technique is developed to improve surface accuracy. The constructed T-splines interpolate every boundary node in the input mesh, with C^2-continuity everywhere except the local region around irregular nodes. Finally, Bézier elements are extracted and linear independence of the constructed T-spline is studied to facilitate isogeometric analysis [2,10].

The remainder of this paper is organized as follows. Section 2 reviews T-splines and defines rational T-splines. Section 3 explains the converting algorithm in detail. Section 4 discusses sharp feature preservation and surface fitting. Section 5 describes a Bézier extraction technique to facilitate isogeometric analysis and studies the linear independence of T-splines. Section 6 presents results, and Sect. 7 draws conclusions.

2 Rational T-spline

T-splines [13] are generalized from NURBS [8], the prevailing industrial standard for surface modeling in Computer Aided Design (CAD) and Computer Aided Manufacturing (CAM). Given a set of $(m+1)\times(n+1)$ control points C_{ij} $(i=0,1,\cdots,m;\ j=0,1,\cdots,n)$, non-negative weights w_{ij} associated with C_{ij}, degree d and two *global* knot vectors, $\mathbf{u}=[u_0,u_1,\cdots,u_{m+d},u_{m+d+1}]$ and

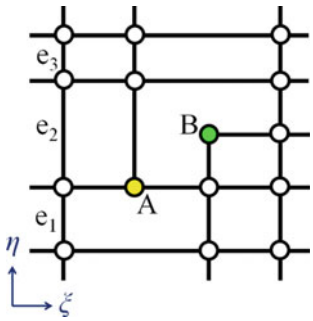


Fig. 2 One local region of a T-mesh. Node A is a T-junction and node B is an L-junction

$\mathbf{v} = [v_0, v_1, \cdots, v_{n+d}, v_{n+d+1}]$, the NURBS surface is defined as

$$S_N(u,v) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} C_{ij} w_{ij} N_{i,d}(u) N_{j,d}(v)}{\sum_{i=0}^{m}\sum_{j=0}^{n} w_{ij} N_{i,d}(u) N_{j,d}(v)}, \tag{1}$$

where $N_{i,d}(u)$ and $N_{j,d}(v)$ are B-spline basis functions defined by the two knot vectors. NURBS provides a unified geometry representation of standard analytical shapes (e.g., conics) and free-form shapes. Additionally, NURBS are invariant under affine as well as perspective transformations. However, NURBS have two drawbacks: they do not allow local refinement and all the control points must lie topologically in a rectangular grid. To overcome these two drawbacks, Sederberg et al. developed T-splines [13], which allow T-junctions and L-junctions in their control grid. A ***T-junction*** terminates a row or column of control points in the control grid, for example node A in Fig. 2. An ***L-junction*** terminates a row and a column of control points, like node B in Fig. 2, which is not permitted in analysis-suitable T-splines [7,2,11]. An analysis-suitable T-spline was defined as a T-spline for which no horizontal T-junction extension intersects a vertical T-junction extension [7]. A T-spline surface is defined by

$$S(\xi,\eta) = \frac{\sum_{i=0}^{n} w_i C_i B_i(\xi,\eta)}{\sum_{i=0}^{n} w_i B_i(\xi,\eta)}, \quad (\xi,\eta) \in \Omega, \tag{2}$$

where w_i is the weight for the control point C_i, $B_i(\xi,\eta) = N_i^{\xi}(\xi) N_i^{\eta}(\eta)$, N_i^{ξ} and N_i^{η} are B-spline basis functions defined by two *local* knot vectors, $\boldsymbol{\xi}_i = [\xi_{i0}, \xi_{i1}, \xi_{i2}, \xi_{i3}, \xi_{i4}]$ and $\boldsymbol{\eta}_i = [\eta_{i0}, \eta_{i1}, \eta_{i2}, \eta_{i3}, \eta_{i4}]$ when degree $d = 3$, and Ω is the local domain[1] of the T-spline in parameter space.

A T-mesh provides the connectivity of the control points and a knot interval is assigned to each edge in the T-mesh to indicate the parametric length of that edge. The local knot vectors for each node are inferred from the T-mesh. In [12], Sederberg et al. introduced three types of T-spline spaces: standard, semi-standard and non-standard, based on whether the basis functions or the weighted basis functions can provide a partition of unity. T-splines provide more flexibilities for modeling. However, there are several open problems which limit their application, such as how to characterize T-mesh configurations for a standard, semi-standard, or non-standard T-spline, and how to calculate the weights for a semi-standard T-spline.

In order to obtain basis functions satisfying a partition of unity, we choose the rational basis functions to construct T-splines. The rational T-spline surface is defined as

$$S(\xi,\eta) = \frac{\sum_{i=0}^{n} w_i C_i R_i(\xi,\eta)}{\sum_{i=0}^{n} w_i R_i(\xi,\eta)}, \quad (\xi,\eta) \in \Omega, \tag{3}$$

where

$$R_i(\xi,\eta) = \frac{N_i^{\xi}(\xi) N_i^{\eta}(\eta)}{\sum_{j=0}^{n} N_j^{\xi}(\xi) N_j^{\eta}(\eta)} \tag{4}$$

is the newly defined rational B-spline basis function, N_i^{ξ} and N_i^{η} are B-spline basis functions defined by the local knot vectors at node C_i, $\boldsymbol{\xi}_i = [\xi_{i0}, \xi_{i1}, \xi_{i2}, \xi_{i3}, \xi_{i4}]$ and $\boldsymbol{\eta}_i = [\eta_{i0}, \eta_{i1}, \eta_{i2}, \eta_{i3}, \eta_{i4}]$ when degree $d = 3$. Similarly, the formula for a rational solid T-spline is

$$S(\xi,\eta,\zeta) = \frac{\sum_{i=0}^{n} w_i C_i R_i(\xi,\eta,\zeta)}{\sum_{i=0}^{n} w_i R_i(\xi,\eta,\zeta)}, \quad (\xi,\eta,\zeta) \in \Omega, \tag{5}$$

where

$$R_i(\xi,\eta,\zeta) = \frac{N_i^{\xi}(\xi) N_i^{\eta}(\eta) N_i^{\zeta}(\zeta)}{\sum_{j=0}^{n} N_j^{\xi}(\xi) N_j^{\eta}(\eta) N_j^{\zeta}(\zeta)} \tag{6}$$

is the newly defined rational B-spline basis function, N_i^{ξ}, N_i^{η} and N_i^{ζ} are B-spline basis functions defined by the local knot vectors at node C_i, $\boldsymbol{\xi}_i = [\xi_{i0}, \xi_{i1}, \xi_{i2}, \xi_{i3}, \xi_{i4}]$, $\boldsymbol{\eta}_i = [\eta_{i0}, \eta_{i1}, \eta_{i2}, \eta_{i3}, \eta_{i4}]$ and $\boldsymbol{\zeta}_i = [\zeta_{i0}, \zeta_{i1}, \zeta_{i2}, \zeta_{i3}, \zeta_{i4}]$ when degree $d = 3$. It is obvious that the rational B-spline basis functions automatically satisfy $\sum_{i=0}^{n} R_i = 1$, for any (ξ,η) in 2D and (ξ,η,ζ) in 3D. In this way, we obtain one set of basis functions satisfying a partition of unity even for non-standard T-splines, and successfully avoid the difficulty of checking the type of T-splines from the T-mesh configuration and calculating the weights for semi-standard cases.

3 Converting algorithm

As shown in Fig. 3, there are two main stages in converting an unstructured quadrilateral or hexahedral mesh to a T-spline

[1] In this paper, "domain" refers to one parametric area in 2D or one parametric volume in 3D and "patch" refers to the T-spline surface or solid T-spline defined on one domain.

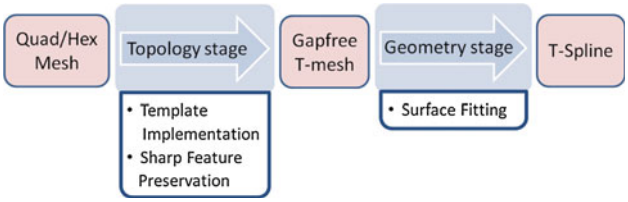


Fig. 3 An overview of the algorithm to convert an unstructured quadrilateral or hexahedral mesh to a T-spline surface or solid T-spline

surface or solid T-spline: the topology stage and the geometry stage. We take the input mesh as the initial T-mesh and the topology stage aims to make the initial T-mesh gap-free by designing templates for each type of node and applying them to elements. Additional nodes and edges are inserted to preserve sharp features in the input mesh. We assign a unit knot interval to each edge in the input quadrilateral or hexahedral mesh. All the new edges inserted in the topology stage have either zero or unit parametric edge length. As a result, a ***quasi-uniform T-mesh*** is constructed, which contains edges with only zero or unit knot interval. Edges with zero knot interval are called ***zero-length edges***. In the geometry stage, the goal is to minimize the error between the input mesh and the output T-spline. Then a T-spline surface or solid T-spline is constructed based on the obtained T-mesh. In addition, Bézier elements are extracted and linear independence is studied in order to facilitate isogeometric analysis. Here are some definitions which will be needed in the following algorithm description.

Definition 3.1 A pair of ***reflection edges*** are two adjacent edges with one common node and all the elements sharing one edge are topologically symmetric with all the elements sharing the other one, with respect to a line of symmetry in 2D or a plane of symmetry in 3D. The line of symmetry in 2D is formed by all the adjacent edges of the shared node except for these two edges. The plane of symmetry in 3D is formed by the adjacent quadrilaterals of the shared node which do not contain any of these two edges. In a pair of reflection edges, one edge is also called the reflection edge of the other one about the shared node and vice versa.

Let us take node A and its adjacent edges AB and AC in Fig. 4a as an example. The blue line DE is the symmetry line, formed by the adjacent edges of A except for AB and AC. The two quadrilaterals adjacent to AB and the two adjacent to AC are topologically symmetric with respect to the symmetry line DE, hence AC is the reflection edge of AB about node A and vice versa. For node A and its two adjacent edges AB and AC in Fig. 4d–e, the blue face is the symmetry plane, formed by the quadrilateral elements adjacent to node A but not containing AB or AC. The hexahedral elements adjacent to AB and all the elements adjacent to AC are topologically symmetric with respect to the blue symmetry plane, hence AB and AC are a pair of reflection edges.

Definition 3.2 A ***regular node*** is a node about which each adjacent edge has a reflection edge.

In Fig. 4a, each edge adjacent to A has a reflection edge. By definition, node A is a regular node in 2D. Node A in Fig. 4d is a regular node as well, because each edge adjacent to A has a reflection edge about A in 3D. A regular node always has a valence of four in 2D (for a regular node on the boundary in 3D, its valence is also always four), and a valence of eight in 3D. However in 3D, a node with a valence of four (on the boundary) or eight may not be a regular node in general.

Definition 3.3 A ***partial extraordinary node*** is an irregular node about which some but not all of its adjacent edges have reflection edges.

Let us take node A in Fig. 4e as an example, its two adjacent edges, AB and AC, are a pair of reflection edges. However, the other adjacent edges do not have reflection edges. Therefore, by definition node A is a partial extraordinary node. Partial extraordinary nodes can only be found in 3D.

Definition 3.4 An ***extraordinary node*** is an irregular node about which none of its adjacent edges has a reflection edge.

For example, node A in Fig. 4b, c is an extraordinary node. In 2D, an extraordinary node has a valence other than four. Figures 4f, g show two extraordinary nodes in 3D, and for the two local regions, none of the edges adjacent to A has a reflection edge. In 2D, we can use the valence number to identity whether a node is regular or extraordinary. However in 3D, situations are more complicated and there is no such direct relationship. In 3D, a regular node has a valence of eight, but a valence-eight node may not be regular. For example, node A in Fig. 4g has a valence of eight but it is an extraordinary one.

We classify all the nodes in the input mesh into three categories: regular, partial extraordinary and extraordinary nodes, and treat them differently during template design. Regular nodes do not introduce gaps in the T-spline or decrease the surface continuity. If all the nodes in the input mesh are regular, for example a structured mesh, the initial T-mesh is topologically correct and gap-free. Unlike regular nodes, partial extraordinary nodes and extraordinary nodes need to be handled properly, otherwise they may introduce gaps in the T-spline model. Hence, a rule is defined for the template design: for each partial extraordinary node or extraordinary node, the template should ensure the constructed T-mesh is gap-free.

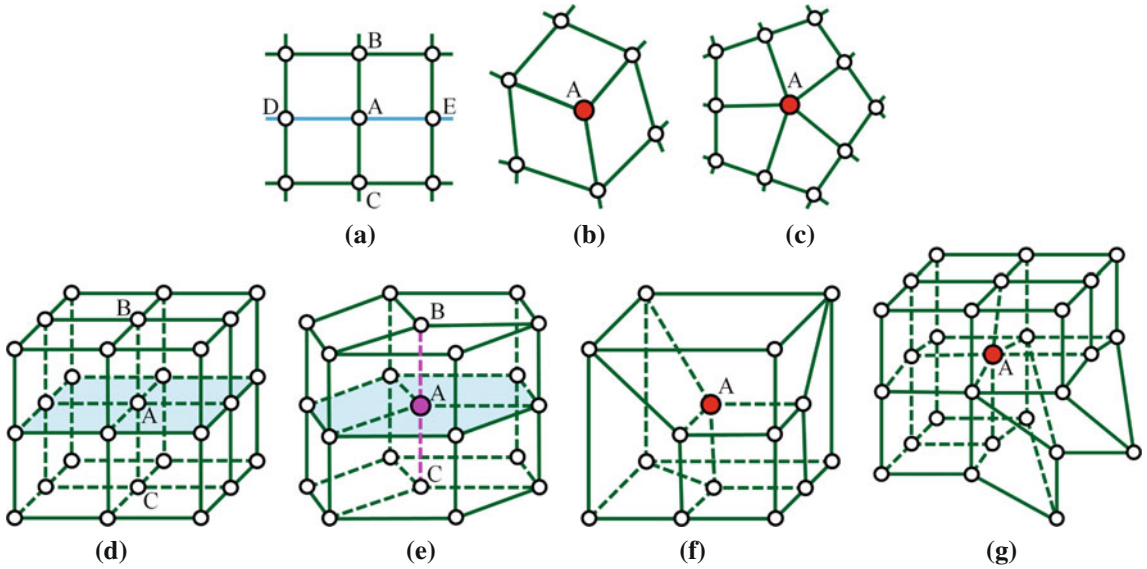


Fig. 4 Regular node, partial extraordinary node and extraordinary node in 2D and 3D. Node *A* rendered in white is a regular node (**a**, **b**), the magenta node is a partial extraordinary node (**e**), and red nodes are extraordinary nodes (**b**, **c**, **f**, **g**)

3.1 Converting a quadrilateral mesh to a T-spline surface

Figure 5 shows three general templates for an extraordinary node in 2D. (a) was derived using T-NURCCs (Non-Uniform Rational Catmull-Clark Surfaces with T-junctions) [13], (b) and (c) are two simplified templates based on (a) with fewer newly inserted nodes and edges. Based on these templates, we design four sets of templates for each quadrilateral element type, see Table 1. There are six types of elements in the initial T-mesh classified by the number of extraordinary nodes: elements with none, one, two (neighboring or diagonal), three and four extraordinary nodes. In the following, we will discuss how to derive these four sets of templates and which set is optimal (with the fewest inserted nodes).

First, let us take set 1 as an example to see how to derive these templates in Table 1 from the template in Fig. 5a. Figure 6 demonstrates the derivation process for element type 2. We first apply the template to the extraordinary node *A*

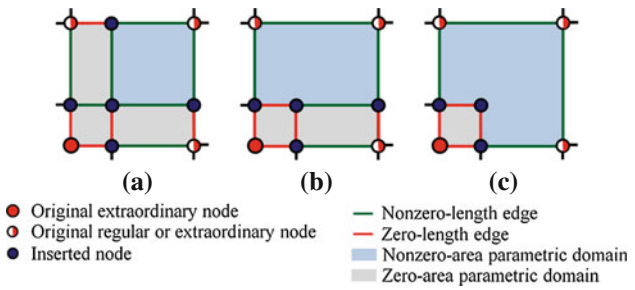
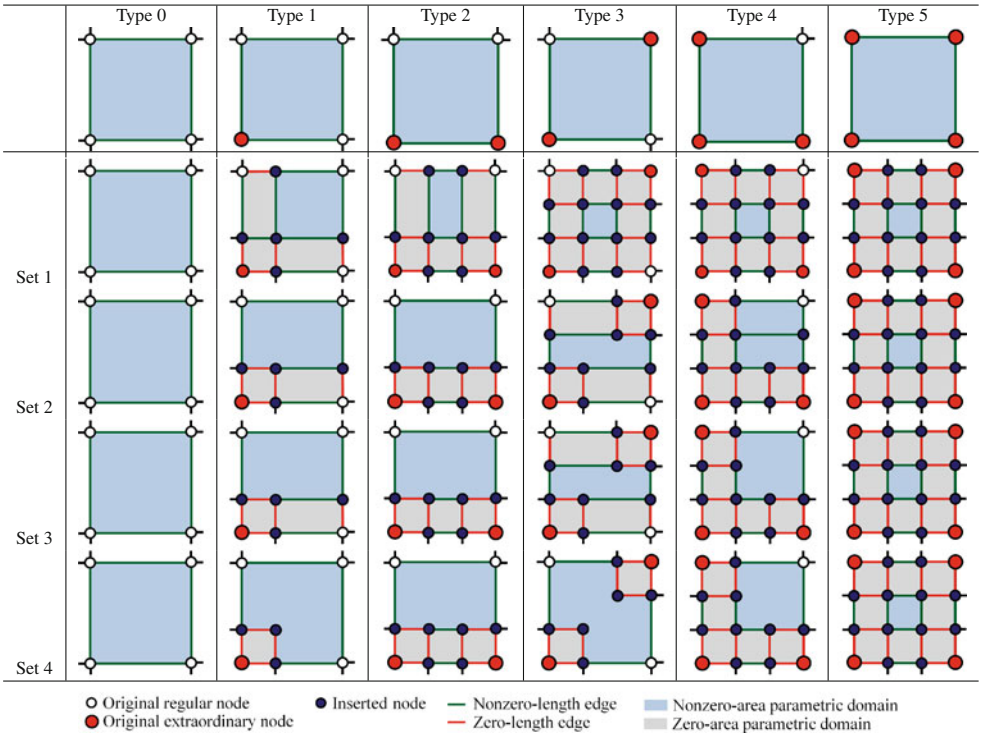


Fig. 5 The general templates for an extraordinary node in 2D, which are used to design templates for each type of element for set 1 (**a**), sets 2 and 3 (**b**), and set 4 (**c**) in Table 1

to obtain the result in Fig. 6b, and then apply the template to the other ordinary node *B* to get the final result. The other templates for set 1 can be derived in the same manner. For sets 2, 3 and 4, we can proceed similarly to derive templates for each type of element using the general templates in Fig. 5b–c. Note that the template in Fig. 5b has two possible orientations. We always choose the one which introduces a minimum number of newly inserted nodes and edges. Figure 7 shows three possible results for element type 2 (sets 2 and 3) after applying the template in Fig. 5b by choosing different orientations. Obviously, (a) is the best result because it introduces the minimum number of new nodes. Figure 8 shows the template derivation process for element type 2 (set 4) from the template in Fig. 5c. Figure 8a is obtained by applying the template to the extraordinary nodes *A* and *B*. The blue edge in Fig. 8b is added according to a T-spline rule [13]: if a T-junction or L-junction on one edge of a face can "legally" be connected to a T-junction or L-junction on an opposing edge of this face, the two T-junctions or L-junctions must be connected in the T-mesh. "Legal" here means that the sum of knot intervals on opposing edges of any face must always be equal.

Sets 1 and 2 were given in [15], and set 2 was simplified from set 1 in order to insert fewer nodes and get better continuity. In [15], the designed templates, together with a T-mesh standardization algorithm, were used to convert an unstructured quadrilateral mesh to a standard T-spline surface. In this paper, we choose the rational basis functions and do not need to get a standard T-spline. We also simplify the template of type 4 elements in set 2 to obtain set 3. Sets 1, 2 and 3 guarantee a gap-free T-spline surface as proved in [15]: for a T-mesh without L-junctions, if the region formed by all the

Table 1 Templates for six quadrilateral element types



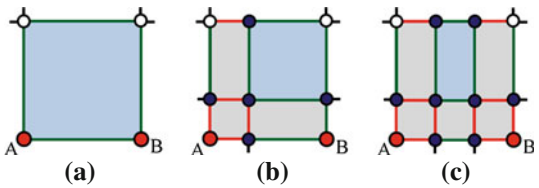


Fig. 6 The template derivation process for element type 2 (set 1) from the template in Fig. 5a. **a** The quadrilateral element with two extraordinary nodes; **b** the result after applying the template to the extraordinary node A; and **c** the final result after applying the template to the other extraordinary node B

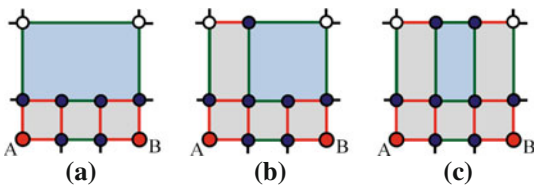


Fig. 7 Three possible results for element type 2 (sets 2 and 3) after applying the template in Fig. 5b by choosing different orientations

adjacent elements of one extraordinary node does not contain any T-junctions and all the edges in it have zero knot intervals, the local region around this extraordinary node is gap-free. These three sets work for any T-mesh. Since we focus on quasi-uniform T-meshes in this paper, we simplify set 3 further and obtain set 4, which is specially designed for quasi-uniform T-meshes and with L-junctions involved.

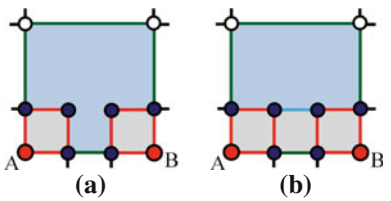


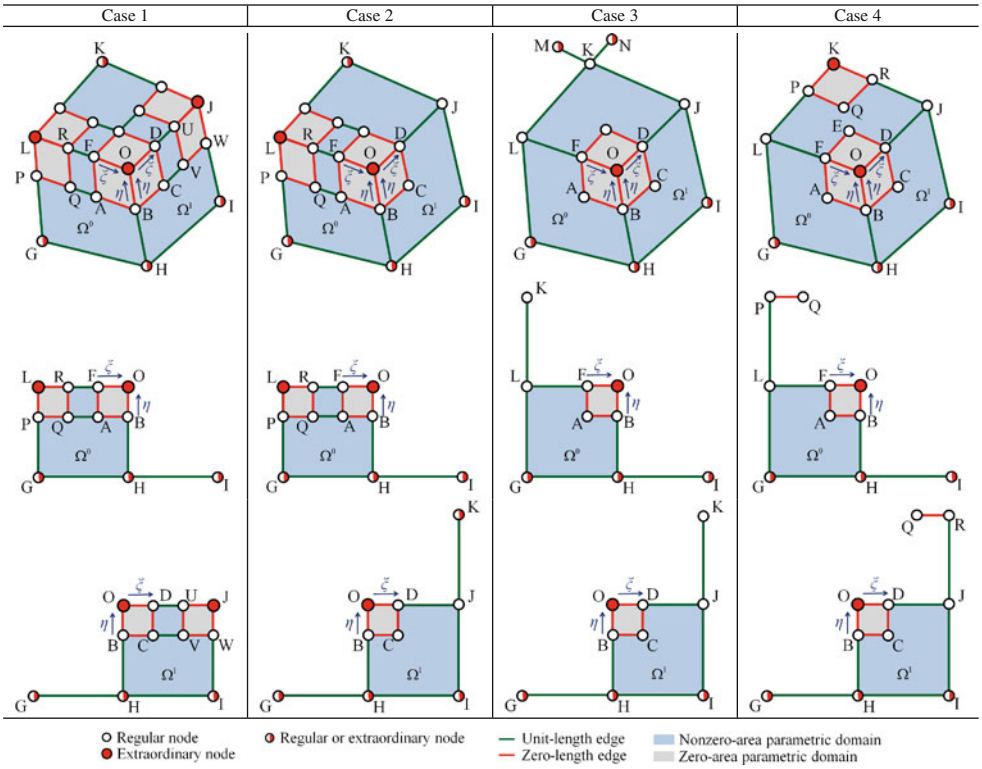
Fig. 8 The template derivation process for element type 2 (set 4) from the template in Fig. 5c. **a** The result after applying the template to the extraordinary nodes A and B; and **b** the final result after adding the blue edge according to a T-spline rule

Set 4 can also generate gap-free T-splines as stated in the following lemma.

Lemma 1 *For any input unstructured quadrilateral mesh, the quasi-uniform T-mesh obtained by applying template set 4 is gap-free.*

Proof We first prove this lemma for a valence-3 node and then extend it to other extraordinary nodes. As shown in Table 2, there are three non-zero domains around a valence-3 extraordinary node O and we first prove the patch defined by Ω^0 is continuous or gap-free with the patch defined by Ω^1. In other words, we need to prove that all the nodes share the same basis function value at the boundary of the two neighboring domains Ω^0 and Ω^1. In Table 2, the top row pictures show the local region of the T-mesh around the extraordinary node O. The second and third rows show the nodes around

Table 2 Local region around one valence-3 extraordinary node



the extraordinary node with non-zero basis function values and their parametric position for Ω^0 and Ω^1, respectively. Node O is the parametric origin. There are four cases for domains Ω^0 and Ω^1: Nodes L and J are both extraordinary nodes (case 1); either L or J is an extraordinary node (case 2); nodes L, J and K are all regular (case 3); and nodes L and J are regular while node K is extraordinary (case 4). The type of nodes G, H, I will not influence the continuity of the two patches defined by Ω^0 and Ω^1.

In case 1, we can observe that except for node O, all the other nodes with non-zero basis function values in Ω^0 and Ω^1 share the same knot vectors. In other words, they share the same basis function in these two domains. In addition, although node O has different basis functions for these two domains, it shares the same basis function value at the shared boundary of the two domains ($\xi = 0, \eta \in [-1, 0]$), because $N^{\xi}_{O,\Omega^0}(0) = N^{\xi}_{O,\Omega^1}(0) = 1$ and $N^{\eta}_{O,\Omega^0} = N^{\eta}_{O,\Omega^1}$ when $\eta \in [-1, 0]$. Therefore, we can conclude that for case 1, the two patches defined by Ω^0 and Ω^1 are gap-free.

In case 2, except for nodes O and K, all the other nodes with non-zero basis function values in Ω^0 and Ω^1 share the same knot vectors. Similar to case 1, node O shares the same basis function value at the shared boundary. Node K has zero basis function value in Ω^0, and although it has non-zero basis function value in Ω^1, the function value is zero at the shared boundary. Hence again, we can conclude that for case 2, the two patches are gap-free.

In case 3, nodes O and K do not share the same knot vectors in Ω^0 and Ω^1 and the other nodes share the same knot vectors. Again, node O shares the same basis function value at the shared boundary. Then we only need to prove node K has the same basis function or the same basis function value at the shared boundary. Suppose nodes M and N are the other two adjacent nodes of node K, besides L and J. There are three possibilities for this case classified by the node types of M and N: both of them are regular; both of them are extraordinary; one is regular and the other is extraordinary.

- When M and N are both regular, the two knot vectors of node K for Ω^0 are $\boldsymbol{\xi}_{K,\Omega_0} = [-3, -2, -1, 0, 1]$, $\boldsymbol{\eta}_{K,\Omega_0} = [-1, 0, 1, 2, 3]$ and the two knot vectors for Ω^1 are $\boldsymbol{\xi}_{K,\Omega_1} = [-1, 0, 1, 2, 3]$, $\boldsymbol{\eta}_{K,\Omega_1} = [-1, 0, 1, 2, 3]$. Since $N^{\xi}_{K,\Omega_0}(0) = N^{\xi}_{K,\Omega_1}(0) = \frac{1}{6}$, we have $N^{\xi}_{K,\Omega_0}(0) N^{\eta}_{K,\Omega_0}(\eta) = N^{\xi}_{K,\Omega_1}(0) N^{\eta}_{K,\Omega_1}(\eta)$ when $\eta \in [-1, 0]$.
- When M and N are both extraordinary, the two knot vectors of node K for Ω^0 are $\boldsymbol{\xi}_{K,\Omega_0} = [-2, -2, -1, 0, 1]$, $\boldsymbol{\eta}_{K,\Omega_0} = [-1, 0, 1, 2, 2]$ and the two knot vectors for Ω^1 are $\boldsymbol{\xi}_{K,\Omega_1} = [-1, 0, 1, 2, 2]$, $\boldsymbol{\eta}_{K,\Omega_1} = [-1, 0, 1, 2, 2]$. Since $N^{\xi}_{K,\Omega_0}(0) = N^{\xi}_{K,\Omega_1}(0) = \frac{1}{6}$, we have $N^{\xi}_{K,\Omega_0}(0) N^{\eta}_{K,\Omega_0}(\eta) = N^{\xi}_{K,\Omega_1}(0) N^{\eta}_{K,\Omega_1}(\eta)$ when $\eta \in [-1, 0]$.
- Suppose M is extraordinary and N is regular. The two knot vectors of node K for Ω^0 are $\boldsymbol{\xi}_{K,\Omega_0} = [-2, -2, -1, 0, 1]$, $\boldsymbol{\eta}_{K,\Omega_0} = [-1, 0, 1, 2, 3]$ and the

two knot vectors for Ω^1 are $\boldsymbol{\xi}_{K,\Omega_1} = [-1, 0, 1, 2, 3]$, $\boldsymbol{\eta}_{K,\Omega_1} = [-1, 0, 1, 2, 2]$. Since $N^{\xi}_{K,\Omega_0}(0) = N^{\xi}_{K,\Omega_1}(0) = \frac{1}{6}$, and using the Oslo knot insertion algorithm [3], we have $N^{\eta}_{K,\Omega_0}(\eta) = N^{\eta}_{K,\Omega_1}(\eta)$, $\eta \in [-1, 0]$. Hence we can obtain $N^{\xi}_{K,\Omega_0}(0)N^{\eta}_{K,\Omega_0}(\eta) = N^{\xi}_{K,\Omega_1}(0)N^{\eta}_{K,\Omega_1}(\eta)$ when $\eta \in [-1, 0]$.

In conclusion, for all the three possibilities, node K always has the same basis function values at the shared boundary of the two adjacent domains. Therefore, the two patches defined by Ω^0 and Ω^1 are gap-free.

In case 4, except for nodes O, P, Q and R, all the other nodes with non-zero basis function values in Ω^0 and Ω^1 share the same knot vectors. In addition, node O has the same basis function value at the shared boundary of the two domains ($\xi = 0, \eta \in [-1, 0]$). Node P has non-zero basis function value in Ω^0, but the function value is zero at the shared boundary and also it has zero basis function value in Ω^1. Node R has the same situation with node P. The knot vectors of node Q are $\boldsymbol{\xi}_{Q,\Omega_0} = [-1, -1, -1, 0, 1]$, $\boldsymbol{\eta}_{Q,\Omega_0} = [-1, 0, 1, 1, 1]$ for Ω^0, and $\boldsymbol{\xi}_{Q,\Omega_1} = [-1, 0, 1, 1, 1]$, $\boldsymbol{\eta}_{Q,\Omega_1} = [-1, 0, 1, 1, 1]$ for Ω^1. Since $N^{\xi}_{Q,\Omega_0}(0) = N^{\xi}_{Q,\Omega_1}(0) = \frac{1}{4}$, we have $N^{\xi}_{Q,\Omega_0}(0)N^{\eta}_{Q,\Omega_0}(\eta) = N^{\xi}_{Q,\Omega_1}(0)N^{\eta}_{Q,\Omega_1}(\eta)$, $\eta \in [-1, 0]$. Therefore, we can conclude that for case 4, the two patches defined by Ω^0 and Ω^1 are gap-free.

In summary, for all these four cases, the two patches defined by Ω^0 and Ω^1 share the same curve when $\xi = 0$ and $\eta \in [-1, 0]$. In other words, the two patches are continuous or gap-free across the shared boundary. Due to symmetry, the surface is gap-free across the boundary shared by the two patches defined by Ω^1 and Ω^2 and likewise for Ω^2 and Ω^0. Hence the surface is gap-free for the local region of the quasi-uniform T-mesh around a valence-3 extraordinary node obtained by applying template set 4. Here, we utilize a valence-3 node in the proof, but the proof can be generalized to other valence numbers. Let us take one valence-5 node O in Fig. 9a as an example, if we treat nodes K, L and J in the same way as the nodes K, L and J in the valence-3 proof, the proof proceeds in the same way. □

Discussion In summary, sets 1 and 2 were designed in [15] to obtain a standard T-mesh, sets 3 and 4 are based on the rational T-spline basis functions. Sets 1, 2 and 3 work for arbitrary T-meshes and set 4 can only be used for quasi-uniform T-mesh. Template set 4 inserts many fewer nodes and zero-length edges, leading to a better surface continuity. For the rational T-spline, we can use all four sets of templates, and we do not need the standardization step. Thus we successfully avoid the propagation in [15], and the results have fewer nodes and better surface continuity.

Let us check the surface continuity around an extraordinary node after applying the four sets of templates. We define the ***p-ring neighborhood*** around an extraordinary node as follows: The one-ring neighborhood consists of all the T-mesh faces adjacent to the extraordinary node. The two-ring neighborhood consists of the one-ring neighborhood plus all the T-mesh faces adjacent to faces on the one-ring neighborhood. This process is repeated as many times as necessary to construct the p-ring neighborhood [15]. Figure 10 shows 1-ring and 2-ring neighborhoods after applying the four template sets. Using set 1, the surface continuity is C^0 at the knot coordinate corresponding to the extraordinary point O, and C^0 across the shared curve of the nonzero area parametric domain for each ring until the 4-ring neighborhood. Using set 2 or 3, the surface continuity is again C^0 at the knot coordinate corresponding to O, C^0 across the shared curve of the nonzero area parametric domain for each ring until the 3-ring neighborhood, and C^1 outside the 3-ring neighborhood until the 4-ring neighborhood. Using set 4, the surface continuity is C^0 at the knot coordinate corresponding to O, and C^0 across the shared curve of the nonzero area parametric domain for each ring until the 3-ring neighborhood. Beyond the 3-ring neighborhood, the surface is C^2-continuous. Therefore, set 4 is the best choice for us due to its better surface continuity and fewer nodes introduced.

Figure 11 shows a comparison of using template sets 1–4. (a) shows the rational T-mesh after applying set 1. The blue edges are added according to a T-spline rule as discussed earlier. After applying set 2 or 3, it is also possible that some

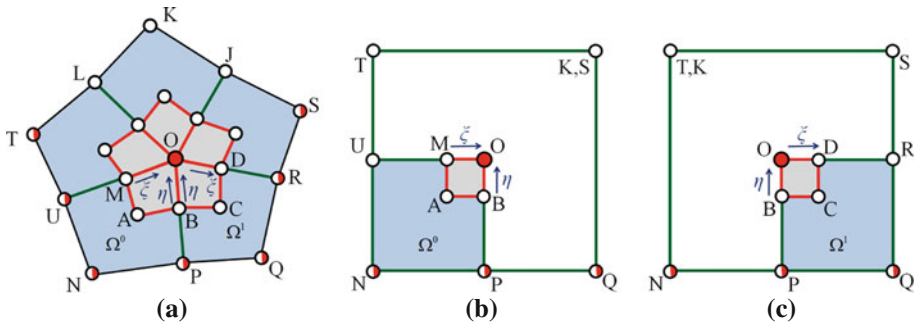


Fig. 9 A local region around one valence-5 extraordinary node O. **b**, **c** show the nodes around the extraordinary node with non-zero basis function values and their parametric position for domains Ω^0 and Ω^1

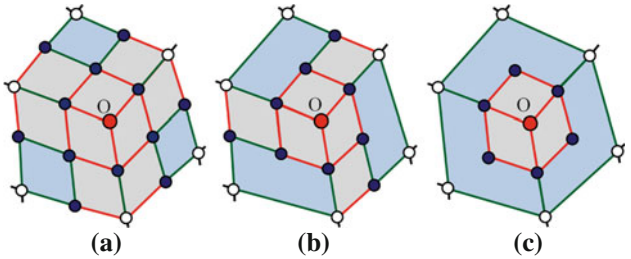


Fig. 10 1-ring and 2-ring neighborhoods around an extraordinary node O in the T-mesh after applying the four template sets in Table 1. **a** Set 1; **b** sets 2 and 3; and **c** set 4

additional edges must be inserted according to this rule. (b) shows a standard T-mesh using set 2, and the green edges are inserted during the standardization step. If we remove all the green edges, (b) becomes the result of a rational T-mesh after applying set 2. (c) and (d) show the results after applying sets 3 and 4, respectively. It is obvious that (d) contains many fewer nodes and zero-length edges, resulting in better surface continuity.

3.2 Converting a hexahedral mesh to a solid T-spline

Hexahedral meshes contain three different types of nodes: regular, partial extraordinary and extraordinary nodes. Here, we first design general templates for partial extraordinary and extraordinary nodes, and then apply them to each type of element. In 3D, one hexahedral element has eight nodes and each node has three possible types. In addition, one partial extraordinary node has three possible orientations. Hence, each node has a total of five possibilities and if we classify the elements as we did for quadrilateral meshes, there will be $5^8 = 390{,}625$ types of elements without considering symmetry and complementary. Therefore, instead of listing all the templates for each type of element as in Sect. 3.1 (Table 1), here we use several examples to explain how to apply the general templates to a certain element type.

Figure 12a–c show the general templates for a partial extraordinary node in hexahedral meshes. The magenta edge adjacent to the partial extraordinary node has a reflection edge about this node. Figure 12a–c are generalized from Fig. 5a–c, respectively. Similar to Fig. 5b, the template in

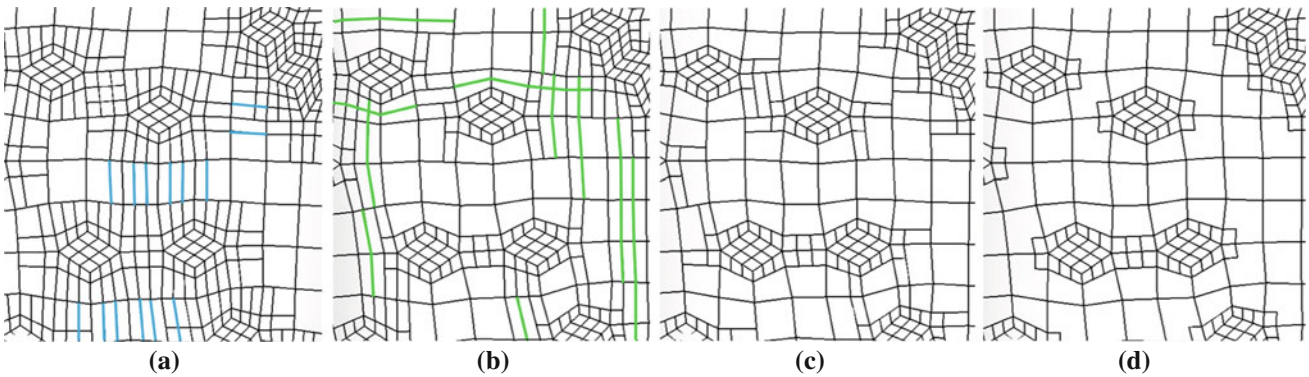


Fig. 11 A comparison of using template sets 1–4. **a** The rational T-mesh after applying set 1; **b** the standard T-mesh after applying set 2 and standardization; **c** the rational T-mesh after applying set 3; and **d** the rational T-mesh after applying set 4

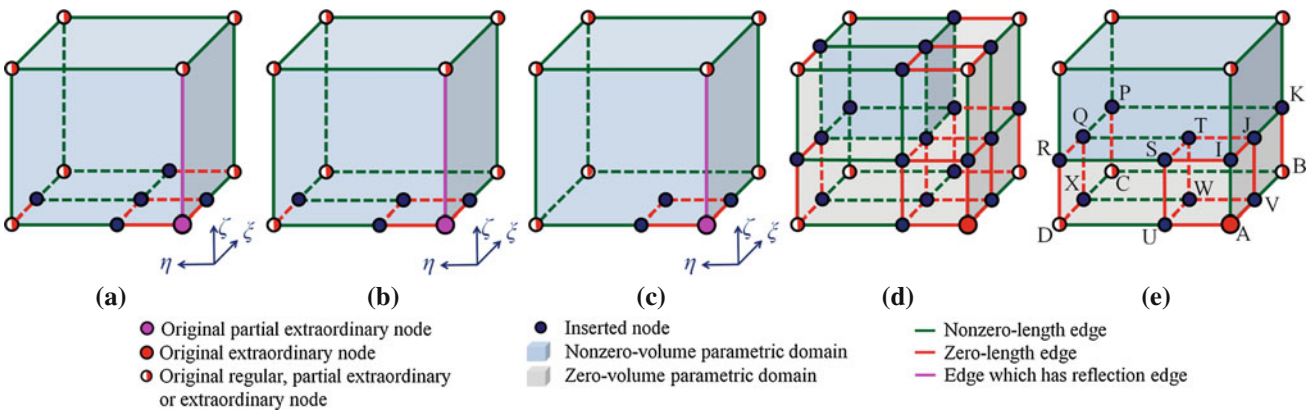


Fig. 12 The general templates for partial extraordinary nodes (**a**–**c**) and extraordinary nodes (**d**–**e**) of hexahedral meshes. **a**–**c** are generalized from templates in Fig. 5a–c, respectively; **d** is extended from the template in Fig. 5a; and **e** is a simplified version of **d**

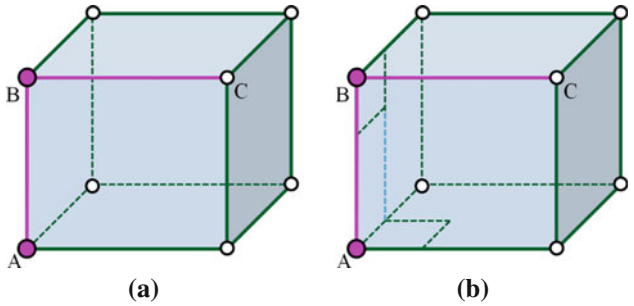


Fig. 13 One element with two partial extraordinary nodes, A and B. **a** The input hexahedral element in which edge AB has one reflection edge about A and edge BC has one reflection edge about B; and **b** the result after applying the template in Fig. 12c to nodes A and B

Fig. 12b also has two possible orientations. Again, we choose the one which introduces fewer new nodes and edges. Obviously, among these three templates, Fig. 12c is the best choice due to its minimum number of newly inserted nodes. Basically, we apply the 2D templates in Fig. 5 to the face perpendicular to the edge which has a reflection edge about this partial extraordinary node. It is no doubt that each node has one unique knot vector along the ζ direction. We have proved the geometry is gap-free on the iso-parametric ξ, η-plane in [15] and in Lemma 1. In other words, $N_i^{\xi}(\xi)N_i^{\eta}(\eta)$ for any node in one domain has the same function values with its neighboring domain at the shared boundary. Hence, $N_i^{\xi}(\xi)N_i^{\eta}(\eta)N_i^{\zeta}(\zeta)$ also has the same function value at the shared boundary of two neighboring domains, and the obtained geometry is gap-free after applying templates in Fig. 12a–c.

Figure 13 shows an example of one element with two partial extraordinary nodes. Nodes A and B are partial extraordinary nodes, edge AB has a reflection edge about A and edge BC has a reflection edge about B. For each partial extraordinary node, we apply the template in Fig. 12c to the face perpendicular to its adjacent edge which has one reflection edge about it. The blue dash edge in (b) is added according to the T-spline rule as discussed earlier.

Figure 12d–e give two general templates for an extraordinary node in hexahedral meshes, which are generalized from Fig. 5. The template in Fig. 12d has one single orientation and it can guarantee the obtained T-mesh gap-free as proved in the following lemma.

Lemma 2 *For the local region of any extraordinary node in the input unstructured hexahedral mesh, the T-mesh obtained by applying the template in Fig. 12d is gap-free.*

Proof To prove the T-mesh obtained is gap-free, we need to find out whether or not the two patches defined by any two neighboring domains around one extraordinary node share the same face. Let us take the local T-mesh around an extraordinary node O in Fig. 14 as an example, where node O is the parametric origin. (b) shows the T-mesh after applying the template in Fig. 12d. To prove the solid T-spline patch defined by the local domain is gap-free, we need to check whether or not the two patches defined by two neighboring domains, say Ω^0 and Ω^1, share the same face at the shared boundary. In other words, we need to check if we have $S^0(\xi, 0, \zeta) = S^1(\xi, 0, \zeta)$, where $\xi \in [0, e_2]$ and $\zeta \in [-e_4, 0]$.

In Ω^0 and Ω^1, only the nodes on the shared face $OGDC$ have non-zero basis function values at the shared boundary

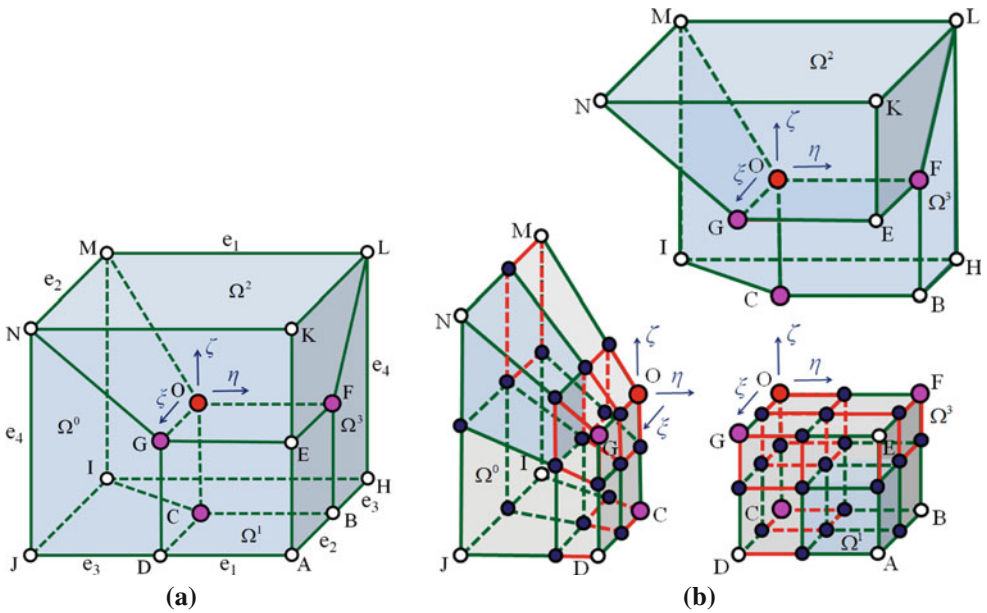


Fig. 14 One local region with an extraordinary node O. **a** shows the input hexahedral mesh; and **b** is the exploded view of the T-mesh after applying the template in Fig. 12d. Node O is the parametric origin

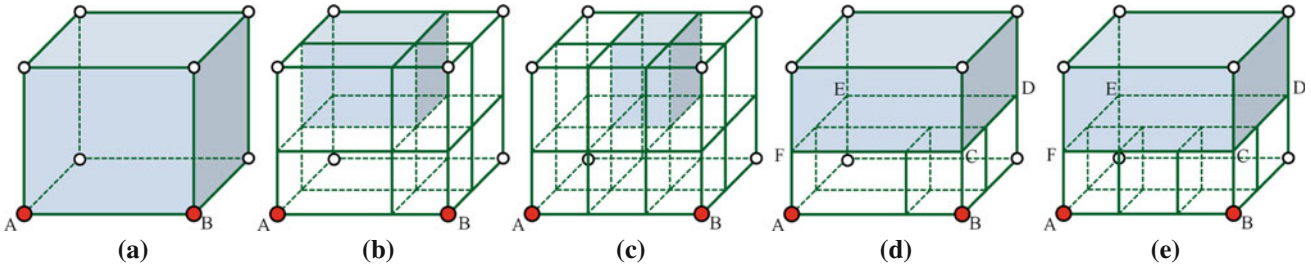


Fig. 15 An example demonstrating how to apply the templates in Fig. 12d–e to one element with two extraordinary nodes, A and B. **a** The hexahedral element with two extraordinary nodes; **b** the result after applying the template in Fig. 12d for node B; and **c** the final result after applying the template in Fig. 12d for node A. Similarly, **d–e** shows the two results after applying the template in Fig. 12e first for node B and then for node A

of these two domains. In addition, these nodes share the same knot vector along the η direction except for O, which is $\boldsymbol{\eta}_{\Omega_0,\Omega_1} = [-e_3, 0, 0, 0, e_1]$. The knot vectors of O are $\boldsymbol{\eta}_{O,\Omega_0} = [-e_3, 0, 0, 0, 0]$ and $\boldsymbol{\eta}_{O,\Omega_1} = [0, 0, 0, 0, e_1]$. We notice that $N[-e_3, 0, 0, 0, e_1](0) = N[-e_3, 0, 0, 0, 0](0) = N[0, 0, 0, 0, e_1](0) = 1$. Additionally, all the nodes with non-zero basis function value at the shared boundary of Ω^0 and Ω^1 share the same knot vectors along the ξ and ζ directions in these two domains. For instance, $\boldsymbol{\xi}_{O,\Omega_0,\Omega_1} = [0, 0, 0, 0, e_2]$ and $\boldsymbol{\zeta}_{O,\Omega_0,\Omega_1} = [-e_4, 0, 0, 0, 0]$. In other words, they have the same basis functions for these two parametric directions, $N^{\xi}_{i,\Omega^0}(\xi)N^{\zeta}_{i,\Omega^0}(\zeta) = N^{\xi}_{i,\Omega^1}(\xi)N^{\zeta}_{i,\Omega^1}(\zeta)$. Thus, we can conclude that $N^{\xi}_{i,\Omega^0}(\xi)N^{\zeta}_{i,\Omega^0}(\zeta)N^{\eta}_{i,\Omega^0}(\eta) = N^{\xi}_{i,\Omega^1}(\xi)N^{\zeta}_{i,\Omega^1}(\zeta)N^{\eta}_{i,\Omega^1}(\eta)$, which means that the two patches defined by Ω^0 and Ω^1 share the same boundary face when $\xi \in [0, e_2]$, $\eta = 0$ and $\zeta \in [-e_4, 0]$, and the two patches are continuous or gap-free across the shared boundary. Due to symmetry, this is also true for the pairs Ω^1–Ω^2, Ω^0–Ω^2, Ω^1–Ω^3, Ω^2–Ω^3 and for the local domains around the extraordinary node O. □

Compared to Fig. 12d, e is a simplified version and it was extended from Fig. 5b. The 2D template in Fig. 5b has one property: for one domain Ω surrounding one extraordinary node O, nodes in other surrounding domains which share only node O with Ω always have zero basis function value in Ω. When extending it to 3D, we want to design templates that inherit this property. Figure 12e is designed based on this principle. Basically, for one extraordinary node in a certain element we insert one cube whose edges all have zero knot intervals, extend the cube along one adjacent edge until the element boundary and insert one plane parallel to one adjacent plane. For example, for the extraordinary node A in Fig. 12e, we insert one cube $AVWU$-$IJTS$, extend the cube along one adjacent edge AD until the element boundary, and insert one plane $IKPR$ parallel to its adjacent face $ABCD$. In contrast with Fig. 12d, there are six possible variations for this template by choosing different adjacent edges for extending the cube and different adjacent faces for inserting the parallel plane. We always choose the orientation which brings the fewest newly inserted nodes. For one extraordinary node in a certain element, if one of its three adjacent nodes is extraordinary or partial extraordinary, the edge containing it will be chosen to extend the cube. Once we fix the orientation to extend the cube, there will be two options left for inserting the plane along the two faces sharing this edge. After we insert the cube, we only need to insert two additional nodes in order to form one plane. Here, we still choose the orientation which will insert fewer nodes. If none of its three adjacent nodes is extraordinary or partial extraordinary, we check the face-diagonal nodes. If one of its face-diagonal nodes is extraordinary or partial extraordinary, we choose the face containing that node to insert the plane.

Figure 15 gives one example demonstrating how to apply the two templates in Fig. 12d–e to one element with two extraordinary nodes, A and B. (b) shows the result after applying the template in Fig. 12d for node B, and then using the template again for node A to obtain the result shown in (c). Since we have one unique orientation, this process is quite straightforward. (d) shows the result after applying the template in Fig. 12e for node B. It is obtained by first inserting one cube around B, then extending the cube along its adjacent edge AB. Since node A is extraordinary, we choose edge AB to extend here. After that, we have two options to choose from to get one adjacent face for inserting one parallel plane, the front face or the bottom face. In this case, both of these options will end up with inserting two additional nodes, hence we can choose either one of them. Here the bottom face is chosen and plane $CDEF$ is inserted. Then the result shown in (d) is obtained. Similarly, since node A already has one plane and one extended cube, we only need to insert one cube around it. Then the result in (e) is obtained.

Figure 16 gives another example demonstrating how to apply the templates in Fig. 12c–e for one element with one extraordinary node B and one partial extraordinary node C. The magenta edge has one reflection edge about node C. (b) is the result after applying the template in Fig. 12d for

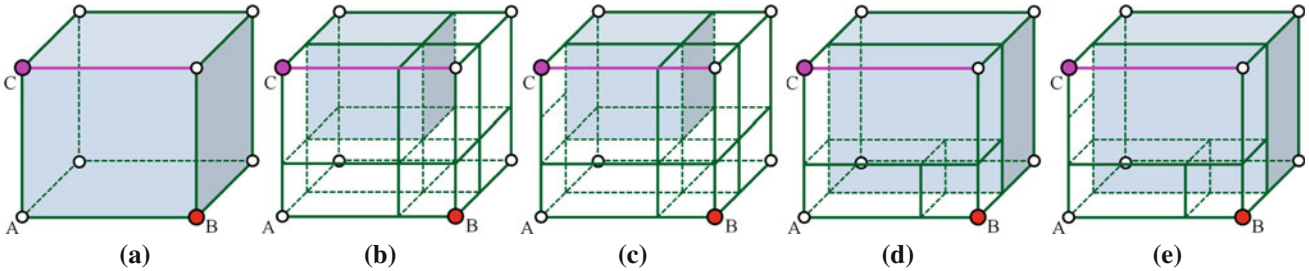


Fig. 16 An example demonstrating how to apply the templates in Fig. 12c–e to one element with one extraordinary node B and one partial extraordinary node C. **a** The hexahedral element in which the magenta edge has one reflection edge about node C; **b** the result after applying the template in Fig. 12d for node B; and **c** the final result after applying the template in Fig. 12c for node C. Similarly, **d–e** show the two results after applying the template in Fig. 12e first for node B and then the template in Fig. 12c for node C

extraordinary node B. Then we apply the template in Fig. 12c for node C to get (c). (d) is the result after applying the template in Fig. 12e for node B. Here we choose the front face to insert one plane because node C lies on this front face diagonal with node B. (e) shows the final result after applying the template in Fig. 12c for node C.

Discussion In summary, we design three general templates for a partial extraordinary node and two general templates for an extraordinary node. Among the three templates for a partial extraordinary node, Fig. 12c introduces many fewer newly inserted nodes. For extraordinary nodes, we prove that using the general template in Fig. 12d will produce a gap-free solid T-spline. Figure 12e is simplified from Fig. 12d, and due to its complexity we do not prove that it can always guarantee a gap-free solid T-spline, although our experience indicates it does.

In 3D, things are much more complicated and here we only discuss the continuity of the neighboring domains adjacent to one partial extraordinary or extraordinary node. In Fig. 12a–c, the patch defined by each domain is C^2-continuous with the patch defined by the non-zero domain sharing the bottom face with it, and C^0-continuous with the patches defined by the domains sharing the front or the right face. In Fig. 12d–e, the patch defined by each domain is C^0-continuous with the patches defined by all the non-zero domains sharing one face with it.

4 Sharp feature preservation and surface fitting

4.1 Sharp feature preservation

For T-spline surfaces, how to preserve sharp feature was described in [15]. The main idea is to use repeating knots to decrease the local boundary surface continuity to C^0, and zero-length edges are inserted across sharp edges and around sharp corners to preserve these sharp features. For each sharp

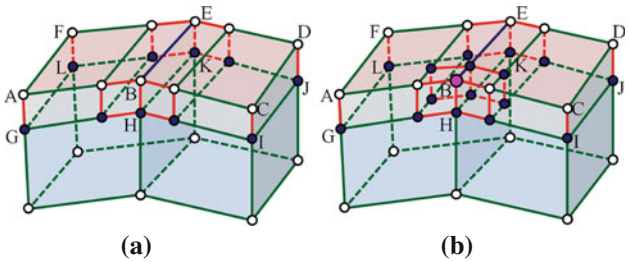
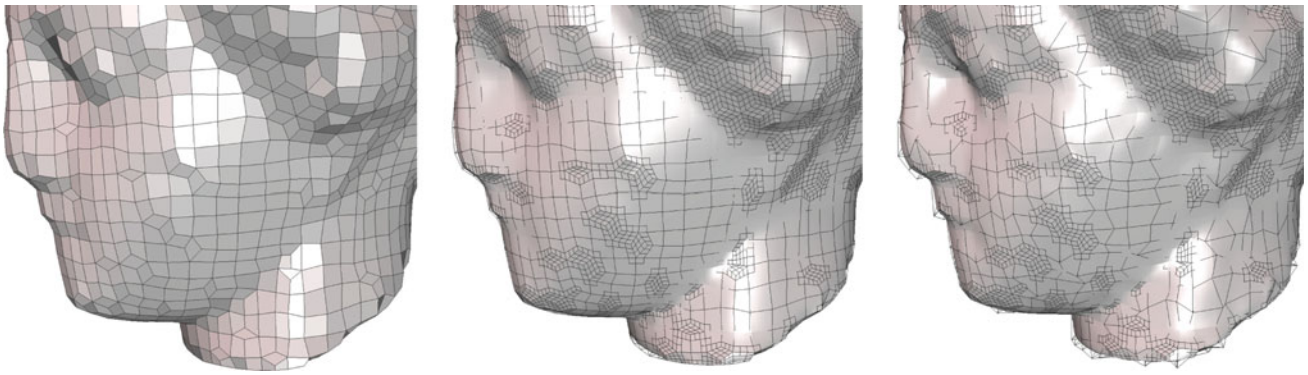


Fig. 17 Sharp feature preservation for solid T-spline. **a** Sharp edge preservation (*the blue edge*); and **b** sharp corner preservation (*the magenta point*)

edge, we duplicate it for each face and all the transverse edges have zero knot interval. Each sharp corner is treated as one extraordinary node.

For solid T-splines, the input hexahedral meshes in this paper were generated using an octree-based isocontouring algorithm together with a pillowing technique [9]. Pillowing is a sheet insertion method that refines the mesh boundary. After pillowing each element has at most one face lying on the boundary. For each boundary element we first insert one face parallel to the boundary face and the edges connecting them have zero knot intervals. In this way, only the boundary nodes have non-zero basis function value on the solid T-spline boundary. For example in Fig. 17, the pink faces are boundary faces and the two faces $GHKL$ and $HIJK$ are newly inserted faces. The edges shown in red between the boundary faces and the inserted faces have zero knot intervals.

To preserve sharp edges in solid T-splines, we insert one edge parallel to the sharp edge for each adjacent boundary face. In contrast with sharp feature preservation for 2D T-spline surfaces, we need to repeat inserting zero-length edges for the newly inserted faces. For example, in Fig. 17a, the blue edge is one sharp edge. We first insert two edges on the boundary faces $ABEF$ and $BCDE$. This step guarantees there are two zero-length edges across the sharp edge. In other words, there are three duplicated knots and the surface



(a) **(b)** **(c)**

Fig. 18 Surface fitting for a solid T-spline model. **a** The input hexahedral mesh; **b** the constructed solid T-spline and T-mesh before surface fitting; and **c** the constructed solid T-spline and T-mesh after surface fitting

continuity is C^0 across the sharp edge. After that, two additional edges are inserted on the newly inserted faces $GHKL$ and $HIJK$. The purpose of this step is to make the boundary face and the layer right below it have the same topology.

To preserve sharp corners in solid T-splines, the sharp corner node is treated as one partial extraordinary node. Its adjacent edge containing one interior node is treated in the same way as when it has one reflection edge. The templates in Fig. 12a–c can be used to handle the sharp corners. Again, the operation done for the boundary face needs to be repeated for the newly inserted faces. For example, in Fig. 17b the magenta node represents one sharp corner and the template in Fig. 12c is applied. Then we apply the same inserting operation to the pillowed faces $GHKL$ and $HIJK$. In this way, the sharp corner is preserved.

4.2 Surface fitting

In the surface fitting step, we aim to relocate T-mesh nodes so that the output T-spline interpolates all the boundary nodes in the input mesh. The surface fitting algorithm given here for 2D is the same as the method given in [15]. We take advantage of the local property of T-splines and fit each Bézier element to interpolate the four nodes of the corresponding quadrilateral element. For each nonzero domain of the T-spline, we set the positions of the four nodes, which come from the input mesh boundary and whose parametric coordinates correspond to the four corners of the domain, as unknown variables and fix all the other nodes. Then we use the interpolation condition to calculate the new position of the unknown variables. After that, all the nodes in the T-mesh which have the same parametric position with the unknown variables are updated. We loop over each non-zero domain and iterate until the interpolation error falls below a given tolerance.

For solid T-splines, the boundary nodes are relocated in the same way as for a T-spline surface. After each reposition of the boundary nodes, we also need to relocate the neighboring interior nodes to avoid tangling the local control mesh and to improve the quality of the Bézier elements. Here, we borrow some ideas from the subject of mesh quality improvement. The interior nodes which come from the input mesh are relocated using smoothing and optimization, and the newly inserted interior nodes are relocated correspondingly. In smoothing, each interior node is moved towards the mass center using its neighboring elements. If the smoothing operation cannot improve the quality, we use optimization for this node. The optimization method loops over all the adjacent hexahedral elements to compute their Jacobians, and then the element with the worst Jacobian is found and optimized, in which the objective function is the Jacobian at that vertex [16]. The Jacobian is used as a metric to measure the mesh quality. Note that relocating control points to improve the quality of Bézier elements is not straightforward. Here we use a mesh metric to relocate the control points, since intuitively a control mesh with better mesh quality will yield a T-spline with better Bézier elements. Figure 18 shows one surface fitting example for a solid T-spline, in which the boundary surface of the solid T-spline after surface fitting in Fig. 18c interpolates all the boundary nodes in the input mesh.

5 Bézier extraction and linear independence

To facilitate isogeometric analysis, Bézier elements are extracted from the constructed T-spline surface or solid T-spline [2,10]. For each nonzero parametric domain, we determine the nodes with nonzero basis function values in this domain and then calculate the transformation matrix M^e between the T-spline basis functions and the Bézier basis functions. In other words, we have

$$B_t^e = M^e B_b^e, \tag{7}$$

where B_t^e is the vector formed by the T-spline basis functions with nonzero function values, and B_b^e is the vector formed

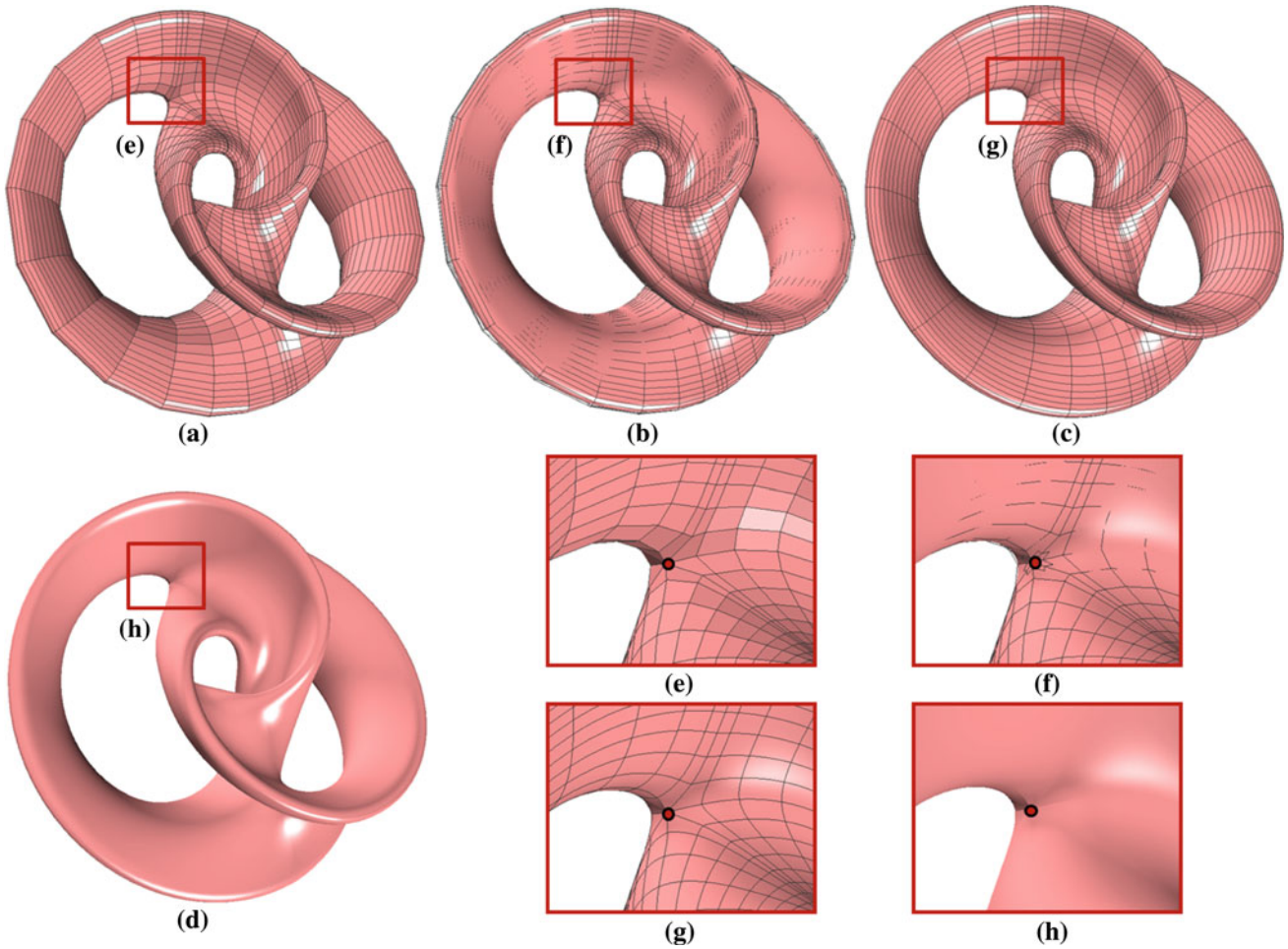


Fig. 19 One genus-3 model. **a** The input unstructured quadrilateral mesh; **b** the constructed T-spline surface and T-mesh; **c** the extracted Bézier elements; and **d** the T-spline surface. **e–h** show details

by the Bézier basis functions. M^e can be calculated using the Oslo knot insertion algorithm [3]. For a T-spline surface,

$$B_t^e = \left[N_0^\xi N_0^\eta, N_1^\xi N_1^\eta, \cdots, N_{n^e-2}^\xi N_{n^e-2}^\eta, N_{n^e-1}^\xi N_{n^e-1}^\eta \right]^T, \tag{8}$$

and

$$B_b^e = \begin{bmatrix} N[0,0,0,0,1](\xi)N[0,0,0,0,1](\eta) \\ N[0,0,0,1,1](\xi)N[0,0,0,0,1](\eta) \\ N[0,0,1,1,1](\xi)N[0,0,0,0,1](\eta) \\ \vdots \\ N[0,0,1,1,1](\xi)N[0,1,1,1,1](\eta) \\ N[0,1,1,1,1](\xi)N[0,1,1,1,1](\eta) \end{bmatrix}, \tag{9}$$

where n^e is the number of nodes with nonzero basis function values in this domain. Similarly, for a solid T-spline

$$B_t^e = \left[N_0^\xi N_0^\eta N_0^\zeta, N_1^\xi N_1^\eta N_1^\zeta, \cdots, N_{n^e-1}^\xi N_{n^e-1}^\eta N_{n^e-1}^\zeta \right]^T, \tag{10}$$

and

$$B_b^e = \begin{bmatrix} N[0,0,0,0,1](\xi)N[0,0,0,0,1](\eta)N[0,0,0,0,1](\zeta) \\ N[0,0,0,1,1](\xi)N[0,0,0,0,1](\eta)N[0,0,0,0,1](\zeta) \\ N[0,0,1,1,1](\xi)N[0,0,0,0,1](\eta)N[0,0,0,0,1](\zeta) \\ \vdots \\ N[0,0,1,1,1](\xi)N[0,1,1,1,1](\eta)N[0,1,1,1,1](\zeta) \\ N[0,1,1,1,1](\xi)N[0,1,1,1,1](\eta)N[0,1,1,1,1](\zeta) \end{bmatrix}. \tag{11}$$

The Bézier extraction technique can also be used to study the linear independence of a given T-mesh. In [7], the linear independence of a T-spline model is determined by computing the nullity of the T-spline-to-NURBS transform matrix. However, this method is not suitable for a T-spline with extraordinary nodes, since converting a T-spline with extraordinary nodes to NURBS will end up with multiple NURBS patches. The T-meshes in this paper have a large percentage of extraordinary nodes. In addition, we need to check the

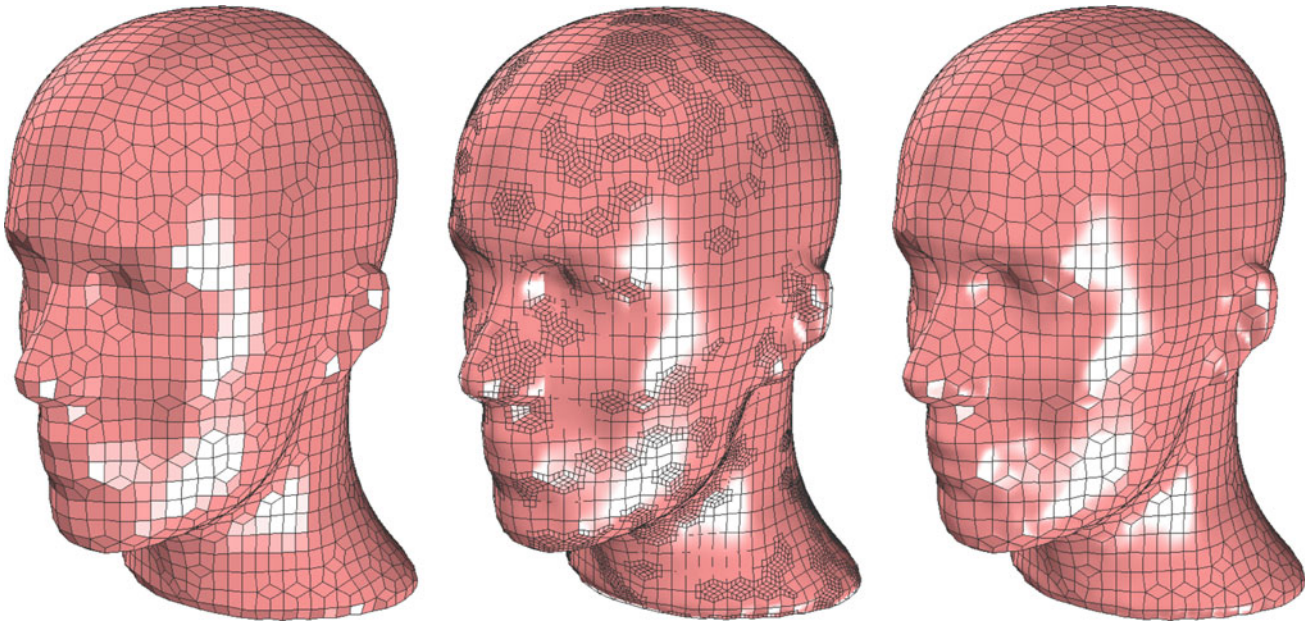


Fig. 20 Human head. **a** The input unstructured quadrilateral mesh; **b** the constructed T-spline surface and T-mesh; and **c** the extracted Bézier elements

linear independence for 3D solid T-splines. Here, in order to study the linear independence of T-splines we assemble all transformation matrices of the non-zero domains or Bézier elements in a similar way as for matrix assembly in the finite element method to get a global transformation matrix. In other words, we calculate the global transformation matrix, K, for the whole model from T-spline to Bézier elements. Then we can obtain

$$B_t = K B_b, \tag{12}$$

where

$$B_t = \left[N_0^\xi N_0^\eta, N_1^\xi N_1^\eta, \cdots, N_{n-2}^\xi N_{n-2}^\eta, N_{n-1}^\xi N_{n-1}^\eta\right]^T \tag{13}$$

is the vector formed by all the T-spline basis functions[2] of the T-mesh, and

$$\begin{aligned} B_b = \big[&B_b^0[0], B_b^0[1], \cdots, B_b^0[15], B_b^1[0], B_b^1[1], \\ &\cdots, B_b^1[15], \cdots, B_b^{m-1}[0], B_b^{m-1}[1], \\ &\cdots, B_b^{m-1}[15]\big]^T \end{aligned} \tag{14}$$

is formed by all the Bézier basis functions of the model. n is the number of nodes in the T-mesh and m is the number of non-zero domains or Bézier elements for the whole model. The matrix K is the global transformation matrix from T-spline to Bézier, with a size of $n \times 16m$ in 2D. In 3D,

$$B_t = \left[N_0^\xi N_0^\eta N_0^\zeta, N_1^\xi N_1^\eta N_1^\zeta, \cdots, N_{n-1}^\xi N_{n-1}^\eta N_{n-1}^\zeta\right]^T, \tag{15}$$

and

$$\begin{aligned} B_b = \big[&B_b^0[0], B_b^0[1], \cdots, B_b^0[63], B_b^1[0], B_b^1[1], \\ &\cdots, B_b^1[63], \cdots, B_b^{m-1}[0], B_b^{m-1}[1], \\ &\cdots, B_b^{m-1}[63]\big]^T. \end{aligned} \tag{16}$$

The matrix K is the global transformation matrix from solid T-spline to solid Bézier and the size of K is $n \times 64m$. Given one T-mesh, all the T-spline basis functions form a linear space. If all these functions are linearly independent, they form a basis of the space with dimension n. The following Lemma 3 provides a necessary and sufficient condition for linear independence of a T-spline.

Lemma 3 (Necessary and sufficient condition for linear independence) *A necessary and sufficient condition for a T-spline model to be linearly independent is that the global transformation matrix from T-spline to Bézier has full rank.*

Proof By definition, the T-spline basis functions are linearly independent if and only if there do not exist scalars, $\alpha = [\alpha_0, \cdots, \alpha_n]^T$, not all zero, such that

$$\alpha^T B_t = 0. \tag{17}$$

In other words,

$$\alpha^T K B_b = 0. \tag{18}$$

[2] The term "basis function" implies linear independence and it is more appropriate to use the terminology "blending function". However, since the majority of T-meshes are linearly independent, the term "basis function" is used here for simplicity.

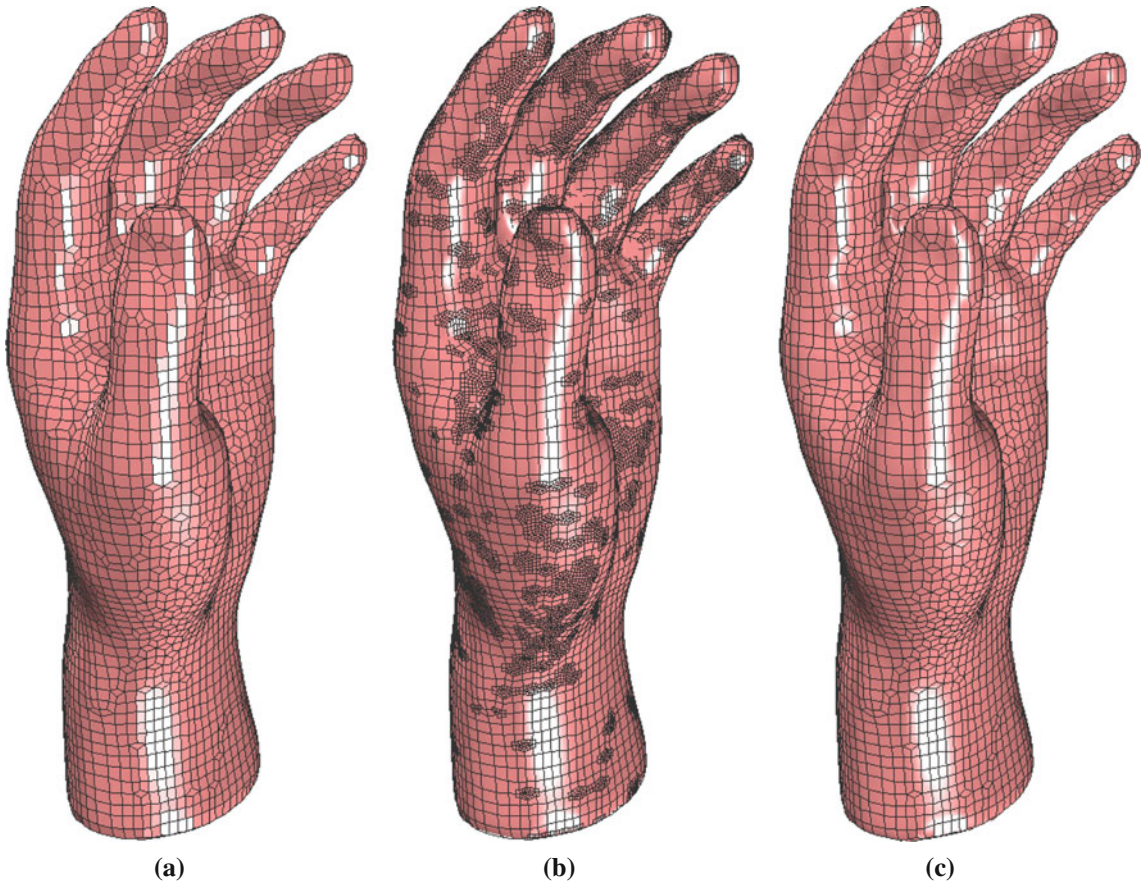


Fig. 21 Human hand. **a** The input unstructured quadrilateral mesh; **b** the constructed T-spline surface and T-mesh; and **c** the extracted Bézier elements

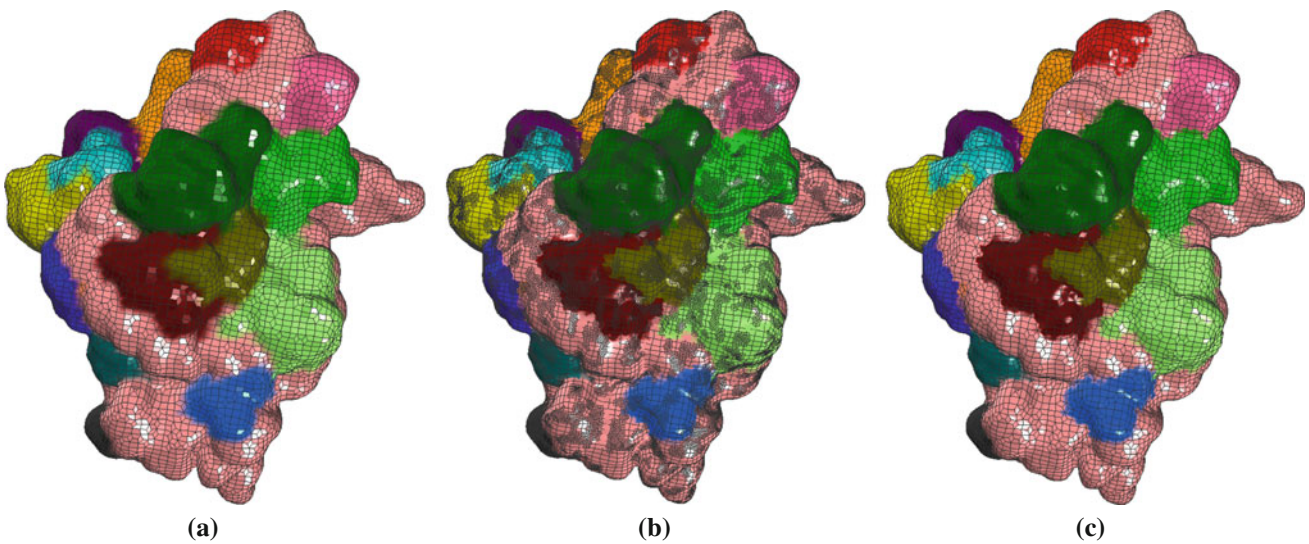


Fig. 22 Ribosome 30S. **a** The input unstructured quadrilateral mesh; **b** the constructed T-spline surface and T-mesh; and **c** the extracted Bézier elements

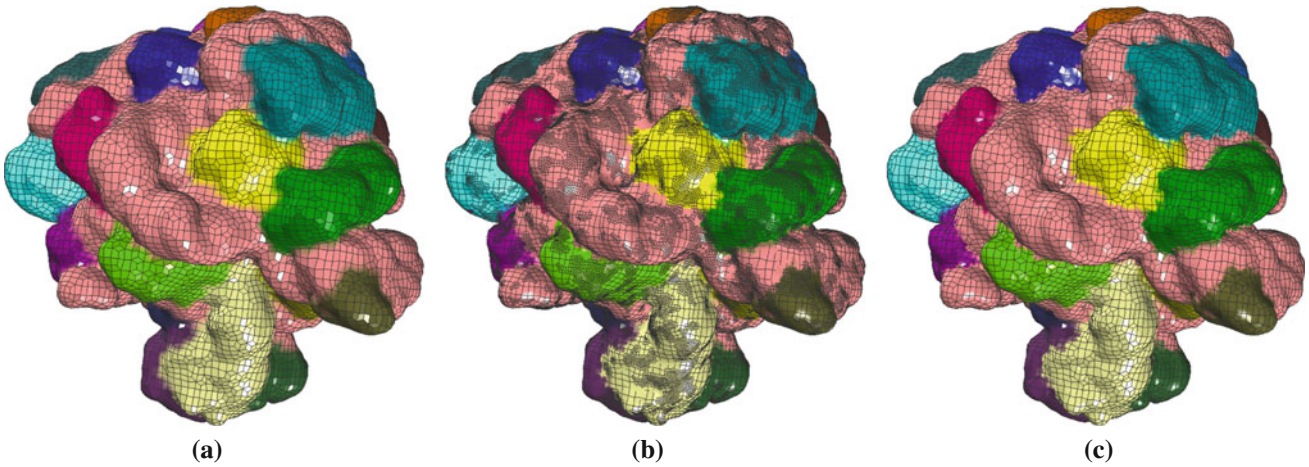


Fig. 23 Ribosome 50S. **a** The input unstructured quadrilateral mesh; **b** the constructed T-spline surface and T-mesh; and **c** the extracted Bézier elements

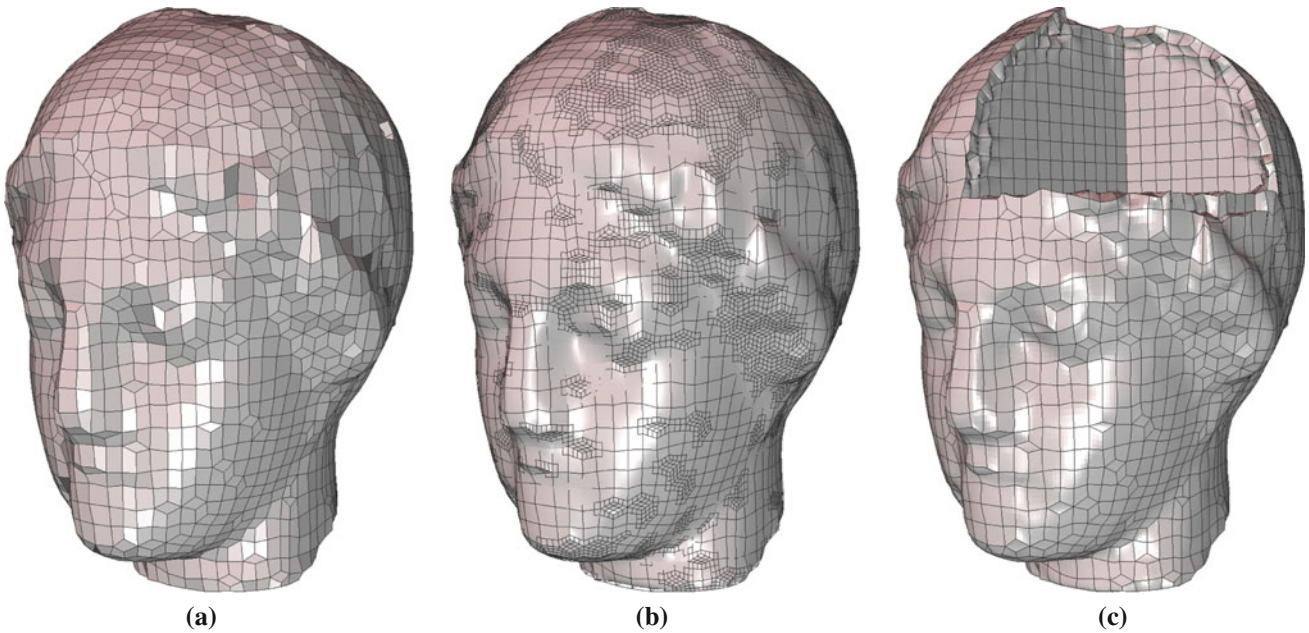


Fig. 24 One statue model. **a** The input unstructured hexahedral mesh; **b** the constructed solid T-spline and T-mesh; and **c** the extracted solid Bézier elements with some elements removed to show the interior mesh

Since the Bézier basis functions are linearly independent, the necessary and sufficient condition for linear dependence of the T-spline basis functions becomes

$$\alpha^T K = K^T \alpha = 0, \tag{19}$$

for $\alpha \neq 0$. This will only happen when K does not have full rank. □

Discussion Lemma 3 utilizes the transformation matrix from T-spline to Bézier elements to study the linear independence of a given T-spline. This technique is more general compared with the linear independence study using the T-spline-to-NURBS transformation matrix. It works for T-meshes with extraordinary nodes. However, generally the size of matrix K is very large, and checking its rank can be very time-consuming.

6 Results

We used template set 4 in Table 1 and applied the converting algorithm to several unstructured quadrilateral meshes (Figs. 19, 20, 21, 22, 23). If the input mesh contains few extraordinary nodes, for example Fig. 19, the output T-spline surface will be very smooth. We also applied our converting algorithm to hexahedral meshes of one statue model and

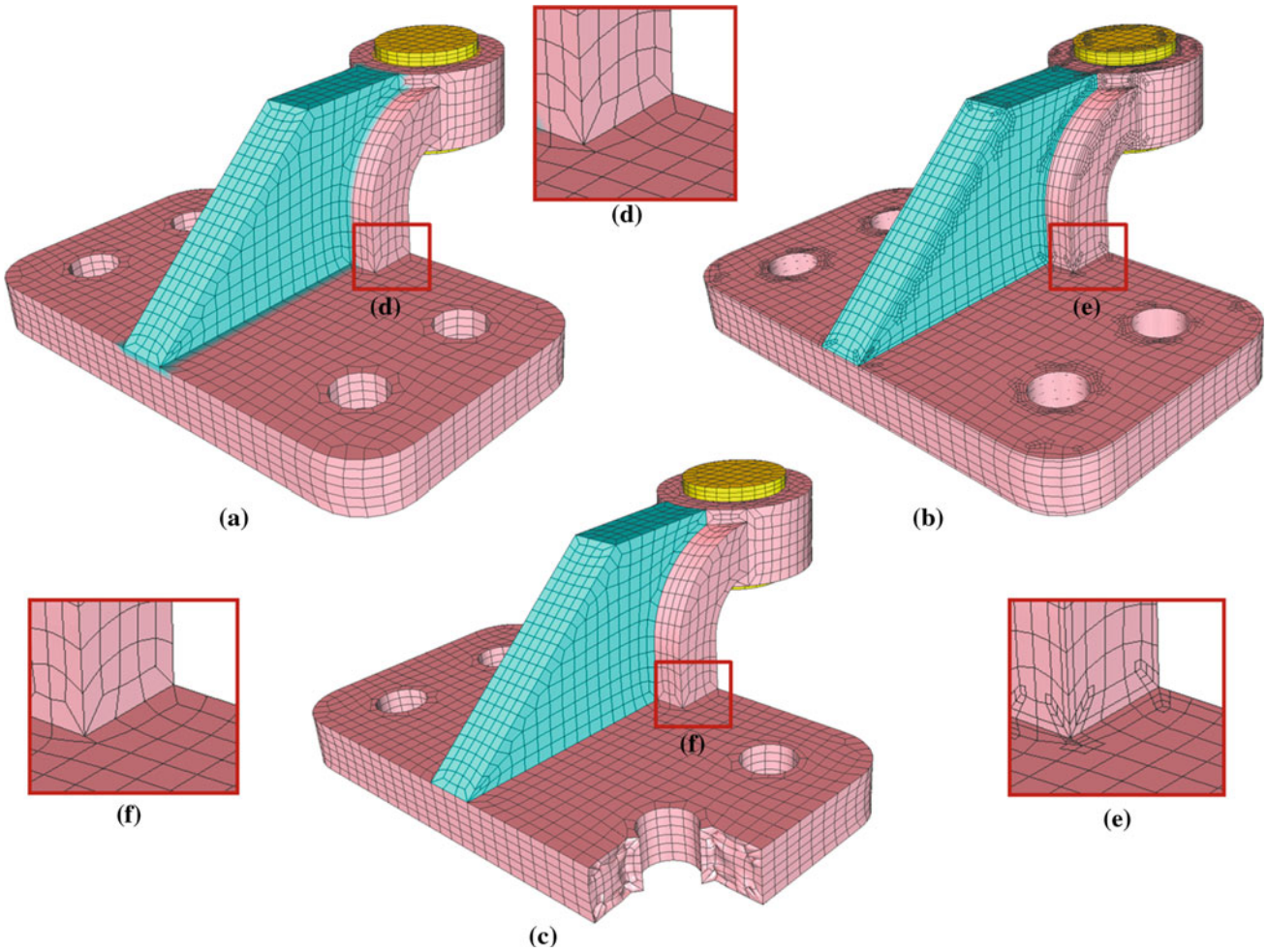


Fig. 25 The bearing assembly. **a** The input unstructured hexahedral mesh; **b** the constructed solid T-spline and T-mesh; and **c** the extracted solid Bézier elements with some elements removed to show the interior mesh. **d–f** show details

several CAD assembly models with sharp features (Figs. 24, 1, 25, 26). For these 3D hexahedral meshes, the templates in Fig. 12c, d are utilized for partial extraordinary nodes and extraordinary nodes, respectively. It is apparent that the constructed solid T-splines preserve all the sharp features in the input model. In Figs. 1, 25 and 26, different colors represent different components of the assembly model. These components have conformal boundaries in the output solid T-spline. The converting algorithm is efficient and all the results were computed on a PC equipped with an Intel Q6600 (4 cores, 2.4GHz) processor and 4GB main memory (DDR2, 800MHz).

Statistics for all the tested models are shown in Table 3. From the table, we can notice that the number of Bézier elements is the same as the number of elements in the input mesh. The reason for this is all the edges we inserted have zero knot interval. There is a subtlety here that is difficult to see. The input mesh consists of bilinear quadrilateral elements or trilinear hexahedral elements. The output T-spline surface is bicubic and C^2-continuous except in the vicinity of extraordinary nodes and the solid T-spline is tricubic and C^2-continuous except in the vicinity of partial extraordinary and extraordinary nodes. *The Bézier elements are embedded in the T-spline.* For all the tested models, the output T-spline surface or solid boundary interpolates all the nodes in the input quadrilateral mesh or hexahedral mesh boundary. The input hexahedral meshes of the three CAD models were generated from NURBS boundary representations in [9]. Although the output solid T-splines interpolate all the nodes on the mesh boundary, there are differences compared with the original NURBS model.

7 Conclusions

We have developed a novel algorithm for converting any unstructured quadrilateral or hexahedral mesh to a T-spline surface or solid T-spline, respectively. The T-spline definition is based on rational T-spline basis functions, with the partition of unity property. The converting algorithm has two

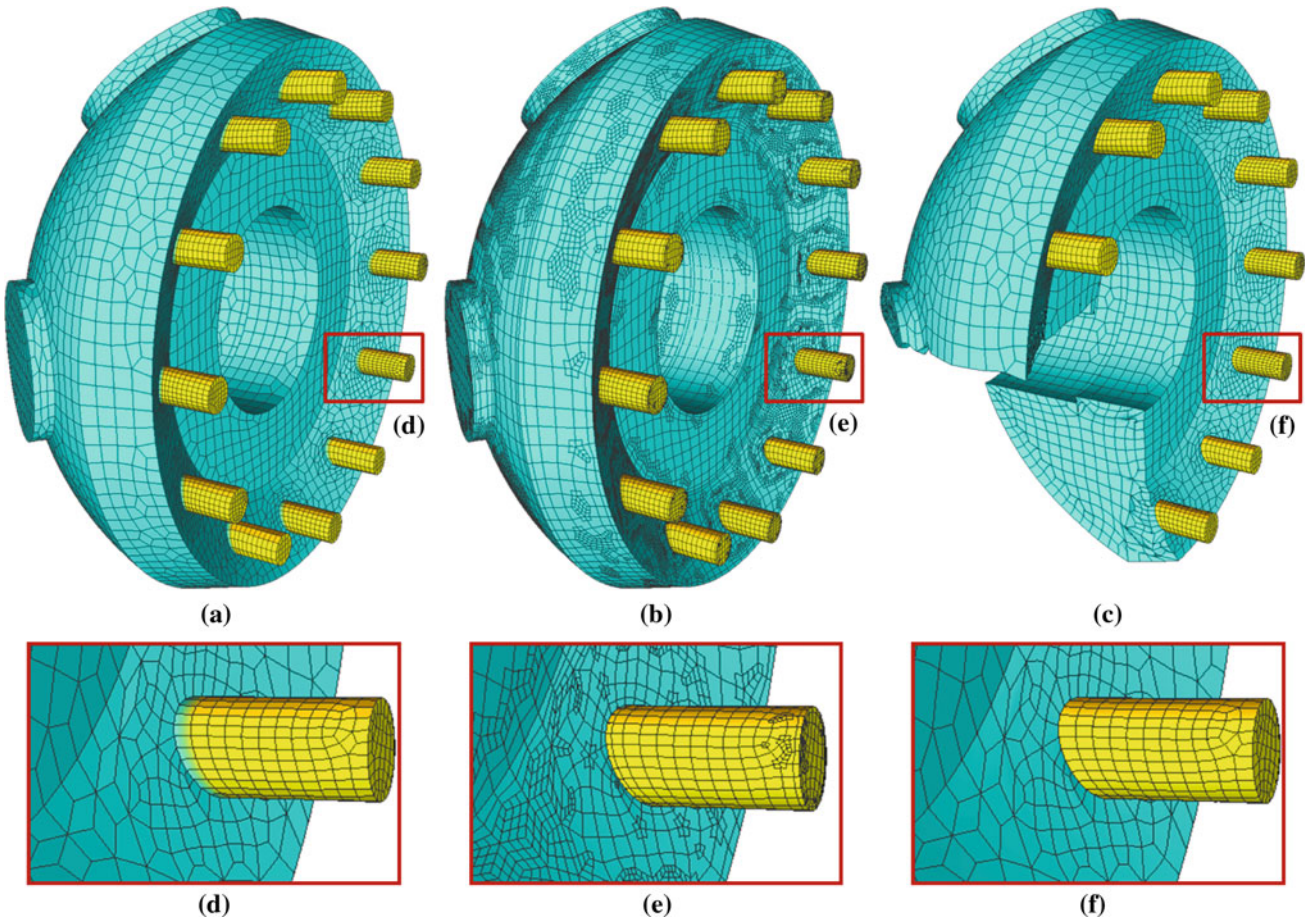


Fig. 26 The eddy plate assembly. **a** The input unstructured hexahedral mesh; **b** the constructed solid T-spline and T-mesh; and **c** the extracted solid Bézier elements with some elements removed to show the interior mesh. **d–f** show details

Table 3 Statistics of all the tested models

	Model	Input mesh		Number of T-mesh nodes	Number of Bézier elements	Time (s)
		(vertices, elements)	(PENs, ENs)			
Quad	Genus-3 model	(3,068, 3,072)	(0, 4)	3,132	3,072	0.24
	Head	(2,909, 2,908)	(0, 1,222)	12,677	2,908	3.5
	Hand	(6,070, 6,068)	(0, 2,527)	26,270	6,068	8.1
	Ribosome 30S	(13,217, 13,215)	(0, 7,217)	70,937	13,215	25.2
	Ribosome 50S	(16,537, 16,537)	(0, 9,039)	88,849	16,537	27.7
Hex	Statue	(14,095, 12,313)	(2,738, 1,663)	114,672	12,313	57.9
	Bearing	(12,184, 10,215)	(2,535, 756)	71,339	10,215	39.8
	Gear	(23,642, 19,438)	(6,149, 2,021)	171,827	19,438	83.1
	Eddy plate	(28,747, 24,269)	(8,309, 2,906)	239,660	24,269	111.2

PEN partial extraordinary node, *EN* extraordinary node

main stages: the topology stage and the geometry stage. In the topology stage, we design templates for each type of node and element in order to get a gap-free T-spline. Sharp features are preserved automatically in this stage. In the geometry stage, an efficient surface fitting technique is developed to improve the surface accuracy. Finally, a Bézier extraction technique is introduced and linear independence of the constructed T-spline is studied to facilitate T-spline based

isogeometric analysis. As part of future work, we plan to study linear independence directly from the T-mesh configuration and construct solid T-splines directly from the NURBS boundary representation.

Acknowledgements We would like to thank J. Qian for providing the input hexahedral meshes of the CAD assemblies. The work of W. Wang and Y. Zhang was supported in part by ONR Grant N00014-08-1-0653. G. Xu was supported in part by NSFC under the grant 60773165, NSFC key project under the grant 10990013 and Funds for Creative Research Groups of China (grant no. 11021101). The work of T. J. R. Hughes was supported by ONR Grant N00014-08-1-0992, NSF GOALI CMI-0700807/0700204, NSF CMMI-1101007 and a grant from SINTEF.

References

1. Bazilevs Y, Calo VM, Cottrell JA, Evans JA, Hughes TJR, Lipton S, Scott MA, Sederberg TW (2010) Isogeometric analysis using T-splines. Comput Methods Appl Mech Eng 199(5–8):229–263
2. Borden MJ, Scott MA, Evans JA, Hughes TJR (2011) Isogeometric finite element data structures based on Bézier extraction of NURBS. Int J Numer Methods Eng 87:15–47
3. Goldman R, Lyche T (1993) Knot insertion and deletion algorithms for B-spline curves and surfaces. Society for Industrial and Applied Mathematics, Philadelphia
4. Hughes TJR, Cottrell JA, Bazilevs Y (2005) Isogeometric analysis: CAD, finite elements, NURBS, exact geometry, and mesh refinement. Comput Methods Appl Mech Eng 194:4135–4195
5. Li W (2006) Automatic mesh to spline surface conversion. PhD thesis, Institut National Polytechnique de Lorraine, France
6. Li W, Ray N, Lévy B (2006) Automatic and interactive mesh to T-spline conversion. In Proceedings of the Fourth Eurographics Symposium on Geometry Processing, pp 191–200
7. Li X, Zheng J, Sederberg TW, Hughes TJR, Scott MA (2012) On linear independence of T-spline blending functions. Comput Aided Geom Des 29(1):63–76
8. Piegl LA, Tiller W (1997) The NURBS book (monographs in visual communication), 2nd ed. Springer-Verlag, New York
9. Qian J, Zhang Y (2010) Sharp feature preservation in octree-based all-hexahedral mesh generation for CAD assembly models. In 19th International Meshing Roundtable, pp 243–62
10. Scott MA, Borden MJ, Verhoosel CV, Sederberg TW, Hughes TJR (2011) Isogeometric finite element data structures based on Bézier extraction of T-splines. Int J Numer Methods Eng 87:15–47
11. Scott MA, Li X, Sederberg TW, Hughes TJR (2011) Local refinement of analysis-suitable T-splines. Comput Methods Appl Mech Eng. doi:10.1016/j.cma.2011.11.022
12. Sederberg TW, Cardon DL, Finnigan GT, North NS, Zheng J, Lyche T (2004) T-spline simplification and local refinement. In ACM SIGGRAPH 2004, pp 276–283
13. Sederberg TW, Zheng J, Bakenov A, Nasri A (2003) T-splines and T-NURCCs. ACM Transac Graph 22(3):477–484
14. Wang H, He Y, Li X, Gu X, Qin H (2007) Polycube splines. In Symposium on Solid and Physical Modeling, pp 241–251
15. Wang W, Zhang Y, Scott MA, Hughes TJR (2010) Converting an unstructured quadrilateral mesh to a standard T-spline surface. Comput Mech 48(4):477–498
16. Zhang Y, Hughes TJR, Bajaj CL (2010) An automatic 3D mesh generation method for domains with multiple materials. Comput Methods Appl Mech Eng 199(5–8):405–415