Isogeometric analysis using T-splines

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

Y. Bazilevs [b], V.M. Calo [d], J.A. Cottrell [a], J.A. Evans [a], T.J.R. Hughes [a], S. Lipton [a], M.A. Scott [a], T.W. Sederberg [c]

[a] Institute for Computational Engineering and Sciences, The University of Texas at Austin, 201 East 24th Street, 1 University Station C0200, Austin, TX 78712, USA
[b] Department of Structural Engineering, University of California, San Diego, 9500 Gilman Drive, Mail Code 0085 La Jolla, CA 92093, USA
[c] Department of Computer Science, Brigham Young University, 3361 TMCB P.O. Box 26576, Provo, UT 84602, USA
[d] Earth and Environmental Science and Engineering, King Abdullah University of Science and Technology, P.O. Box 55455, Jeddah 21534, Saudi Arabia

We explore T-splines, a generalization of NURBS enabling local refinement, as a basis for isogeometric analysis. We review T-splines as a surface design methodology and then develop it for engineering analysis applications. We test T-splines on some elementary two-dimensional and three-dimensional fluid and structural analysis problems and attain good results in all cases. We summarize the current status of T-splines, their limitations, and future possibilities.

Keywords:
T-splines
NURBS
Isogeometric analysis
CAD
FEA
Fluid dynamics
Structural analysis
PB-splines

1. Introduction

Engineering design and analysis have reached a critical juncture. Each has its own geometric representation, and the design description, embodied in computer aided design (CAD) systems, needs to be translated to an analysis-suitable geometry for mesh generation and use in a Finite Element Analysis (FEA) code. This task is far from trivial. For complex engineering designs it is now estimated to take over 80% of overall analysis time, and engineering designs are becoming increasingly complex. For example, presently, a typical automobile consists of about 3000 parts, a fighter jet over 30,000, the Boeing 777 over 100,000, and a modern nuclear submarine over 1,000,000 (see Fig. 1).

Engineering design and analysis are not separate endeavors. Design of sophisticated engineering systems is based on a wide range of computational analysis and simulation methods, such as structural mechanics, fluid dynamics, acoustics, electromagnetics, heat transfer, etc. Design speaks to analysis, and analysis speaks to design. However, analysis-suitable models are not automatically created or readily meshed from CAD geometry. Although not always appreciated in the academic analysis community, model generation is much more involved than simply generating a mesh. There are many time consuming, preparatory steps involved. And one mesh is no longer enough. According to Steve Gordon, Principal Engineer, General Dynamics Electric Boat Corporation, "We find that today's bottleneck in CAD-CAE integration is not only automated mesh generation; it lies with efficient creation of appropriate 'simulation-specific' geometry". (In the commercial sector, analysis is usually referred to as CAE, which stands for Computer Aided Engineering.)

The anatomy of the process has been studied by Ted Blacker, Manager of Simulation Sciences at Sandia National Laboratories, and is summarized in Fig. 2 along with the breakdown in the percentage of time devoted to each task. Note that at Sandia mesh generation accounts for only 20% of overall analysis time, whereas creation of the analysis-suitable geometry requires about 57%, and only 23% of overall time is actually devoted to analysis per se. The approximate 80/20 modeling/analysis ratio seems to be a very common industrial experience and there is a strong desire to reverse it, but so far little progress has been made despite enormous effort to do so. The integration of CAD and FEA has proved a formidable problem. It is our opinion that fundamental changes must take place to fully integrate engineering design and analysis.

Recent trends taking place in engineering analysis and high-performance computing are also demanding greater precision and tighter integration of the overall modeling–analysis process. We note that a finite element mesh is only an approximation of the CAD geometry, which we will view as "exact". This approximation can in many situations create errors in analytical results. The

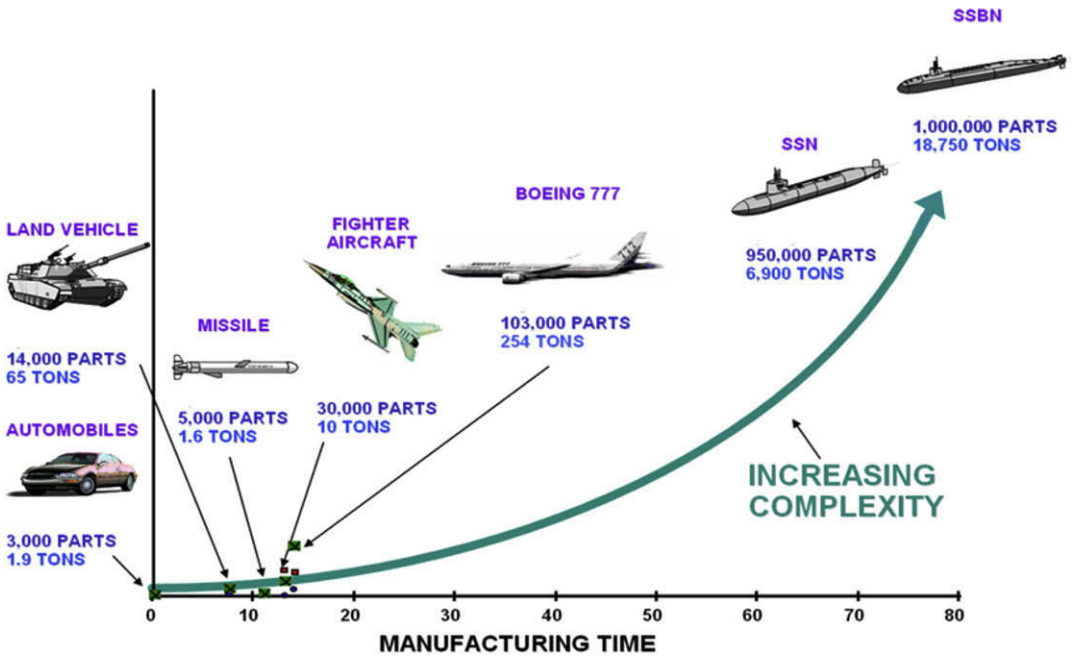


Fig. 1. Engineering designs are becoming increasingly complex. As the number of parts comprising an object increases, so too does the amount of time required for it to be manufactured. Such growth in complexity makes analysis a time consuming and expensive endeavor. (Courtesy of General Dynamics/Electric Boat Corporation.)

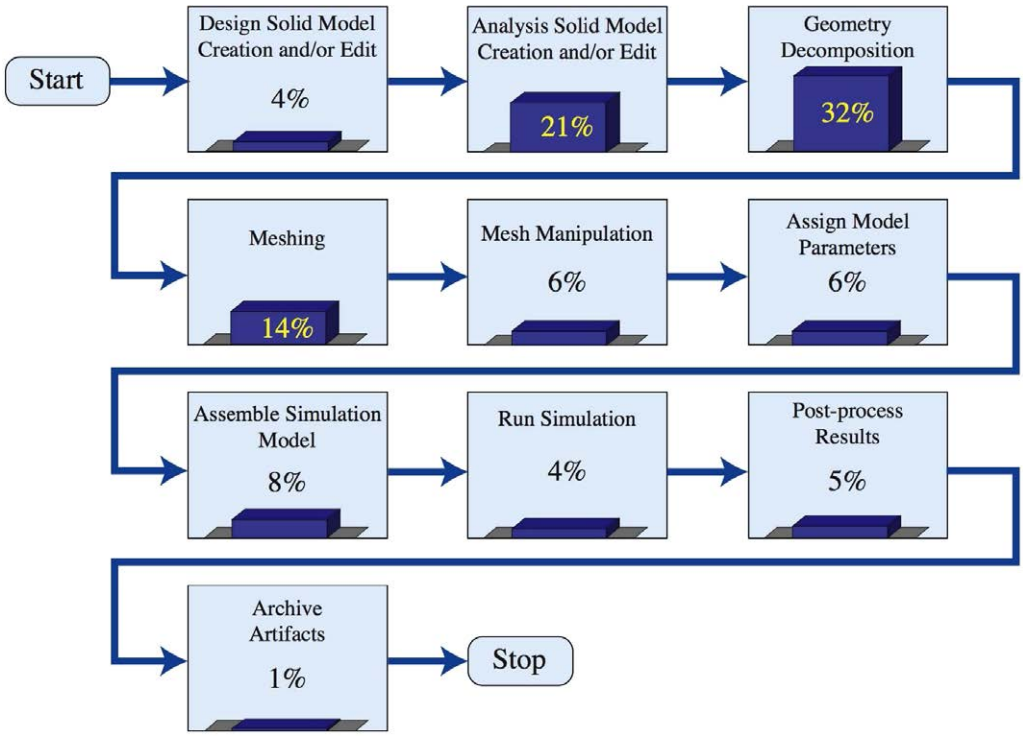


Fig. 2. Estimation of the relative time costs of each component of the model generation and analysis process at Sandia National Laboratories. Note that the process of building the model completely dominates the time spent performing analysis. (Courtesy of Ted Blacker, Sandia National Laboratories.)

following examples may be mentioned: shell buckling analysis is very sensitive to geometric imperfections, boundary layer phenomena and lift and drag are sensitive to precise geometry of aerodynamic and hydrodynamic configurations, and sliding contact between bodies cannot be accurately represented without precise geometric descriptions. Automatic adaptive mesh refinement has not been as widely adopted in industry as one might assume from the extensive academic literature because mesh refinement requires access to the exact geometry, and thus it also requires seamless and automatic communication with CAD, which simply does not exist. Without accurate geometry and mesh adaptivity, convergence and -precision results are in many cases impossible.

Deficiencies in current engineering analysis procedures also preclude successful application of important pace setting technol-

ogies, such as design optimization, verification and validation (V&V), uncertainty quantification (UQ), and petascale computing. The benefits of design optimization have been largely unavailable to industry. The bottleneck is that to do efficient shape optimization the CAD geometry-to-mesh mapping needs to be automatic, differentiable and tightly integrated with the solver and optimizer. This is simply not the case as meshes are disconnected from the CAD geometries from which they were generated. V&V requires error estimation and adaptivity, which in turn requires tight integration of CAD, geometry, meshing, and analysis. UQ requires simulations with numerous samples of models needed to characterize probability distributions. Sampling puts a premium on the ability to rapidly generate geometry models, meshes, and analyses, which again leads to the need for tightly integrated geometry, meshing, and analysis. The era of petaflop computing is on the horizon. Parallelism keeps increasing, but the largest unstructured

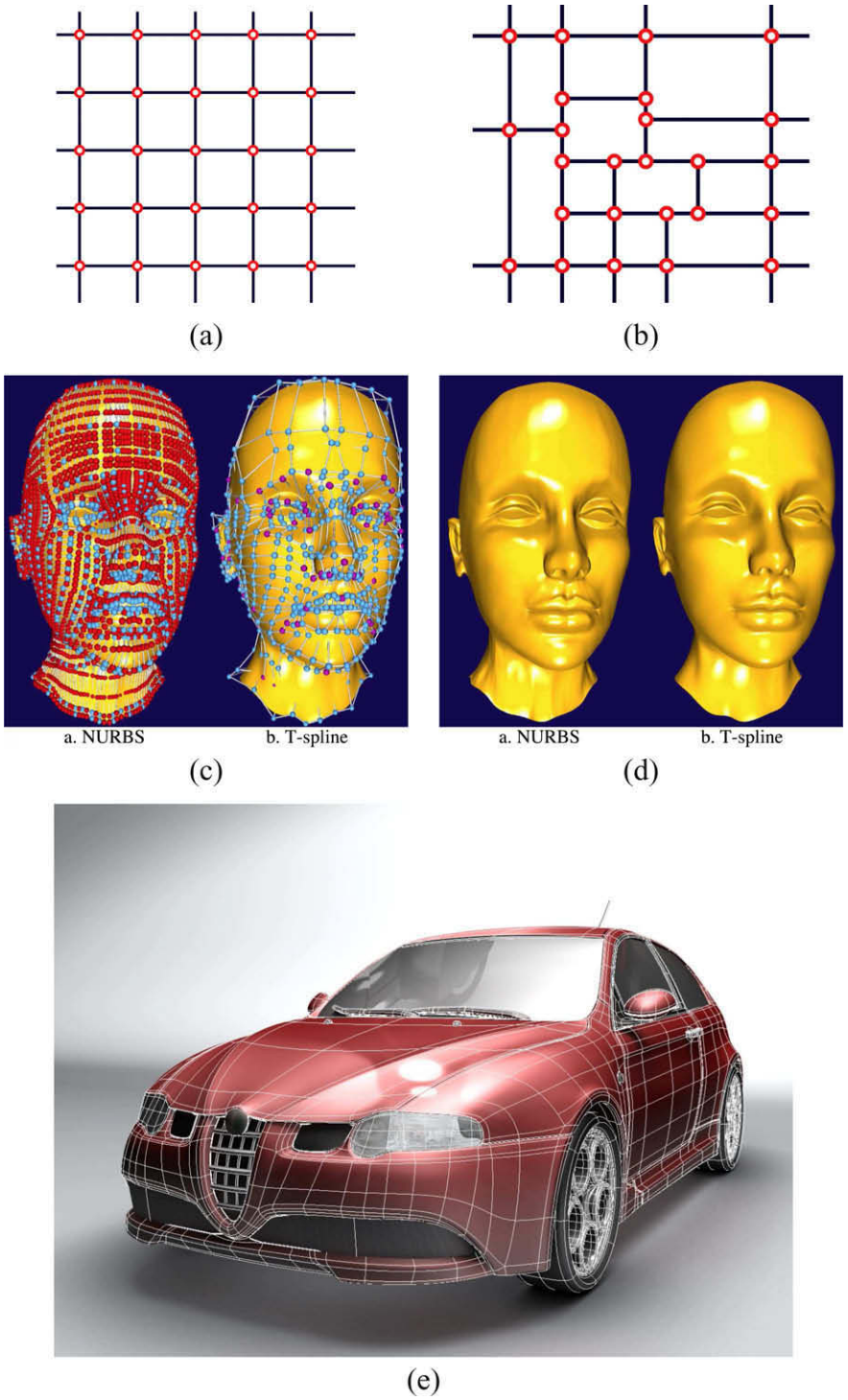


Fig. 3. NURBS control points lie in a rectangular grid. Rows of T-spline control point can be incomplete. (a) Topology of NURBS control grid. (b) Topology of sample T-spline control grid. (c) NURBS and T-spline control grids for head model. 4800 NURBS control points; 1100 T-spline control points. (d) Head modeled as NURBS and T-spline. (e) Car modeled using T-splines; one-fourth the modeling time of NURBS.

mesh simulations have stalled because no one truly knows how to generate and adapt massive meshes that keep up with increasing concurrency. To be able to capitalize on the era of $O(10^5)$ core parallel systems, CAD, geometry, meshing, analysis, adaptivity, and visualization all have to run in a tightly integrated way, in parallel and scalably.

Our vision is that the only way to breakdown the barriers between engineering design and analysis is to reconstitute the entire process. We believe that the fundamental step is to focus on one, and only one, geometric model, which can be utilized directly as an analysis model, or from which geometrically exact analysis models can be automatically built. This will require a change from classical finite element analysis to an analysis procedure based on exact CAD representations. This concept is referred to as isogeometric analysis, and was first proposed in Hughes et al. [30] and further developed in [3,4,6,7,18,19,21,22,27,49,52]. Isogeometric analysis is based on the same computational geometry representation as the CAD model.

There are a number of candidate computational geometry technologies that may be used in isogeometric analysis. The most widely used in engineering design is NURBS (non-uniform rational B-splines), the industry standard (see, e.g. [23,24,28,37,38,40]). The major strengths of NURBS are that they are convenient for free-form surface modeling, can exactly represent all quadric surfaces, such as cylinders, spheres, ellipsoids, etc., and that there exist many efficient and numerically stable algorithms to generate

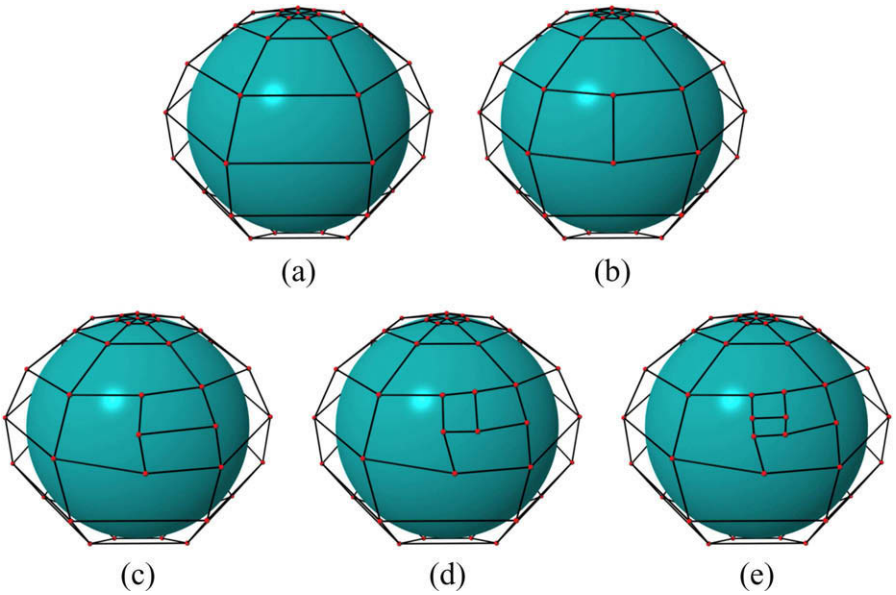


Fig. 4. Local refinement using T-splines. The added control points do not change the surface.

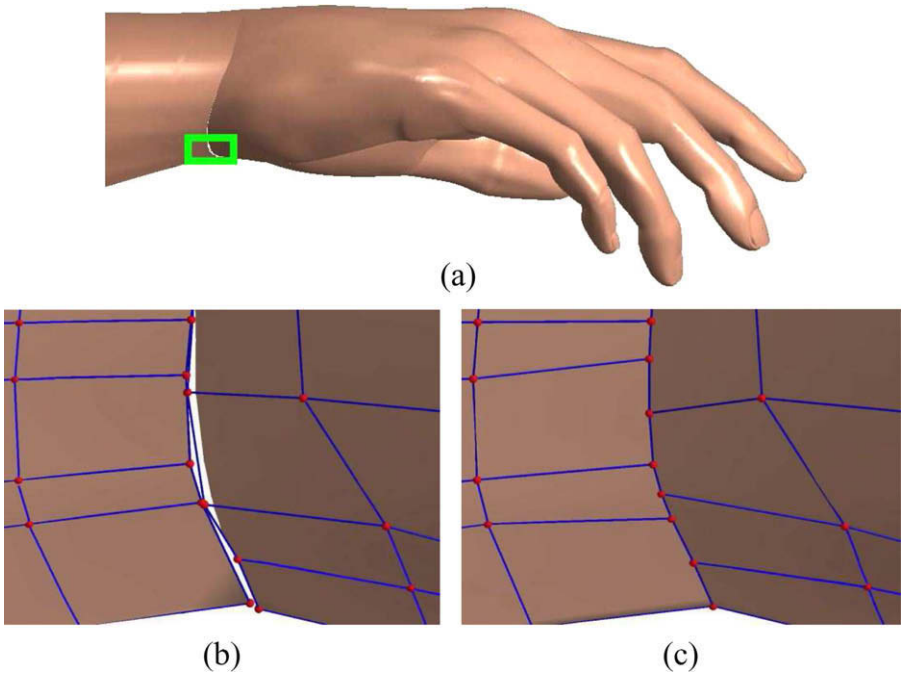


Fig. 5. (a) Hand modeled using seven NURBS surfaces. (b) Blowup of highlighted region showing NURBS control grids. (c) T-spline control grid; the gap is closed using T-spline merging.

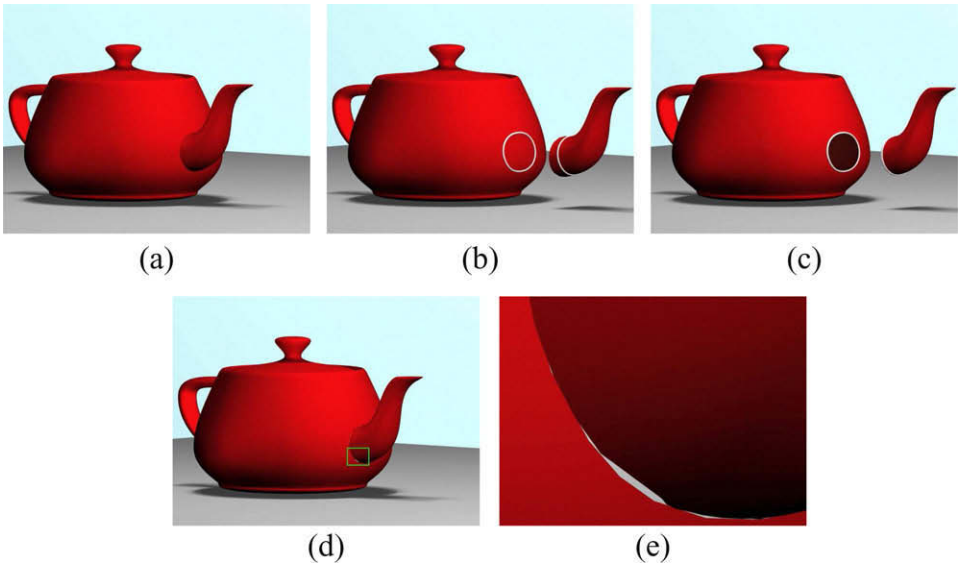


Fig. 6. Utah teapot model. (a) The original NURBS surfaces. (b) Intersection curves where the body and spout intersect (the spout has been moved away from the body). (c) Trimmed-NURBS representation in which the hole is cut in the body and the excess part of the spout is trimmed away. (d) Trimmed-NURBS body and spout moved back to their original positions. (e) Blowup showing the gap between the body and spout inside the green rectangle in (d).

NURBS objects. They also possess useful mathematical properties, such as the ability to be refined through knot insertion, C^{p-1}-continuity for degree p NURBS, and the variation diminishing and convex hull properties.[1] NURBS are ubiquitous in CAD systems, representing billions of dollars in development investment. The major deficiencies of NURBS are that gaps and overlaps at intersections of surfaces cannot be avoided, complicating mesh generation, and that they utilize a tensor product structure making the representation of detailed local features inefficient. Furthermore, it is impossible to represent most shapes using a single, watertight NURBS surface. T-splines are a recently developed generalization of NURBS technology (see [44]). T-splines correct the deficiencies of NURBS in that they permit local refinement and coarsening, and a solution to the gap/overlap problem. Commercial bicubic T-spline surface modeling capabilities have been recently introduced in Maya [45] and Rhino [46], two NURBS-based design systems. Extensions of T-spline surfaces to arbitrary polynomial degree have been described in Finnigan [26].

A NURBS surface is defined using a set of control points, which lie, topologically, in a rectangular grid, as shown in Fig. 3a. This means that a large percentage of NURBS control points are superfluous in that they contain no significant geometric information, but merely are needed to satisfy the topological constraints. In Fig. 3c, 80% of the NURBS control points are superfluous (colored red[2]). By contrast, a T-spline control grid is allowed to have partial rows of control points, as shown in Fig. 3b. A partial row of control points terminates in a T-junction, hence the name T-splines. In Fig. 3c, the purple T-splines control points are T-junctions. For the head modeled by NURBS and T-splines in Fig. 3c and d, the T-spline model requires only 24% of the control points compared to the NURBS model. For a designer, fewer control points means faster modeling time. The artist that created the T-spline model of the car in Fig. 3e estimated that it took one-fourth the time it would have taken using NURBS. Refinement, the process of adding new control points to a control mesh without changing the surface, is an important basic operation used by designers. A limitation of NURBS is that refinement requires the insertion of an entire row of control points. T-junctions enable T-splines to be locally refined. As shown in Fig. 4, a single control point can be added to a T-spline control grid. Another limitation of NURBS is that because a single NURBS surface must have a rectangular topology, most objects must be modeled using several NURBS surfaces. The hand in Fig. 5 is modeled using seven NURBS patches, one for the forearm, one for the hand, and one for each finger. It is difficult to join multiple NURBS surfaces in a single watertight model, as illustrated in Fig. 5a and b, especially if corners of valence other than four are introduced. However, it is possible to merge together several NURBS surfaces into a single gap-free T-spline, as shown in Fig. 5c.

Other methods have been devised for creating watertight smooth surfaces of arbitrary topology. A notable example is subdivision surfaces [51], which are defined in terms of various refinement rules that map a control polyhedron of arbitrary topology to a smooth surface. They have been used for shell analysis by Cirak et al. [15–17]. The appeal of subdivision surfaces is that, like T-splines, they create gap-free models and there is no restriction on the topology of the control grid. Subdivision surfaces are gaining widespread adoption in the animation industry. Most of the characters in Pixar animations are modeled using subdivision surfaces [51]. The president of Walt Disney Animation Studios and Pixar Animation Studios, Catmull, was one of the inventors of subdivision surfaces in 1978 [14]. The CAD industry has not adopted subdivision surfaces very widely because they are not compatible with NURBS. With billions of dollars of infrastructure invested in NURBS, the financial cost would be prohibitive.

Another serious problem with NURBS is that it is mathematically impossible for a trimmed NURBS to accurately represent the intersection of two NURBS surfaces without introducing gaps in the model. A reason for this is that a generic curve of intersection between two bicubic patches is degree 324 [41], whereas the degree of the image of a conventional trimming curve is only 18. Fig. 6 illustrates how these gaps occur. Fig. 6a shows the Utah teapot model in which the body and spout are modeled as two distinct NURBS surfaces. Using surface intersection operations that are standard in CAD systems, the spout cuts a hole in the body, and the body cuts away the excess portion of the spout. The white rings

[1] The standard variation diminishing property holds for NURBS curves but does not extend to surfaces.

[2] For interpretation of color in Figures, the reader is referred to the web version of this article.

in Fig. 6b show where the cuts will occur, and Fig. 6c shows the resulting cuts. The trimmed teapot body and spout in Fig. 6c are represented using trimmed NURBS. In Fig. 6d, the trimmed spout is moved back into position relative to the trimmed body. However, they do not form a watertight model, as the blowup in Fig. 6e shows.

An NSF-sponsored workshop in 1999 [1] identified the unavoidable gaps in trimmed NURBS as the most pressing unresolved problem in the field of CAD. This problem is a major cause of the interoperability between CAD and analysis software [34], which was once estimated to cost the US automotive industry alone over $1 billion annually [12]. One existing approach is to employ "healing" software, which does not fix the problem, but only reduces the size of gaps. The problem is a significant ingredient in the design-to-analysis bottleneck because a CAD model must be closed in order to generate an analysis-suitable mesh. T-splines provide a way to close the gap [43]. Fig. 7 shows how T-splines can represent the NURBS model in Fig. 6 as a single, gap-free T-splines surface using the merge capability of T-splines shown in Fig. 5c. In addition to the fact that T-splines solve many of the problems with NURBS that have vexed the CAD community for three decades, T-splines are forward and backward compatible with NURBS. Every NURBS is a special case of a T-spline (i.e., a T-spline with no T-junctions or extraordinary points) and every T-spline can be converted into one or more NURBS surfaces by performing repeated local refinement to eliminate all T-junctions. Compatibility is crucial for commercialization, especially in a mature industry like CAD, and it may allow T-splines and NURBS to co-exist during a gradual period of T-spline adoption.

The isogeometric concept would endeavor to use the surface design model directly in analysis. This would only suffice if analysis only requires the surface geometry, such as in the stress or buckling analysis of a shell. In many cases, if not most, the surface will enclose a solid and an analysis model will need to be created for the solid. The basic problem is to develop a three-dimensional (trivariate) representation of the solid in such a way that the surface representation is exactly preserved. This is far from a trivial problem. Surface differential and computational geometry and topology are now fairly well understood [50], but the three-dimensional problem is still open [47,48]. The hope is that through the use of new technologies, such as Ricci flows and polycube splines [35,50], progress will be forthcoming. The polycube spline concept has features in common with the template-based system developed by Zhang et al. [52] for modeling patient-specific arterial fluid–structure geometries.

The current state-of-the-art in isogeometric analysis is as follows: A number of single and multiple patch NURBS-based parametric models have been developed and analyzed [4,7,18,19,27,30,52]. Various mesh refinement schemes have been investigated, namely, *h*-, *p*- and *k*-refinement (i.e., mesh, C^0 degree

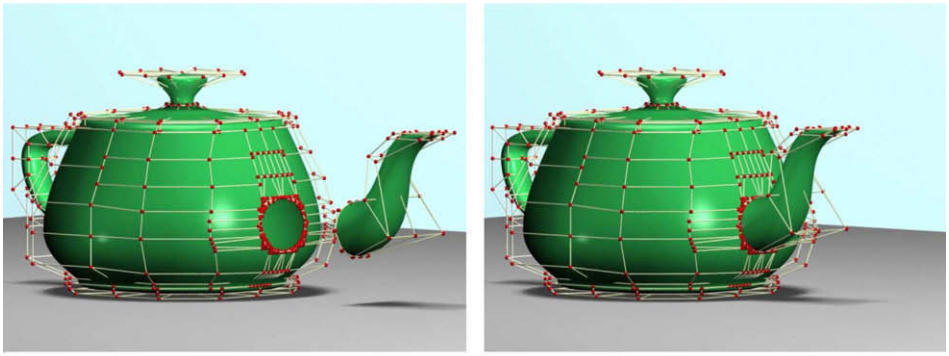


Fig. 7. Utah teapot model. (a) The trimmed body and spout from Fig. 6c each modeled as a T-spline surface. (b) The two T-splines in (a) merged into a single gap-free T-spline.

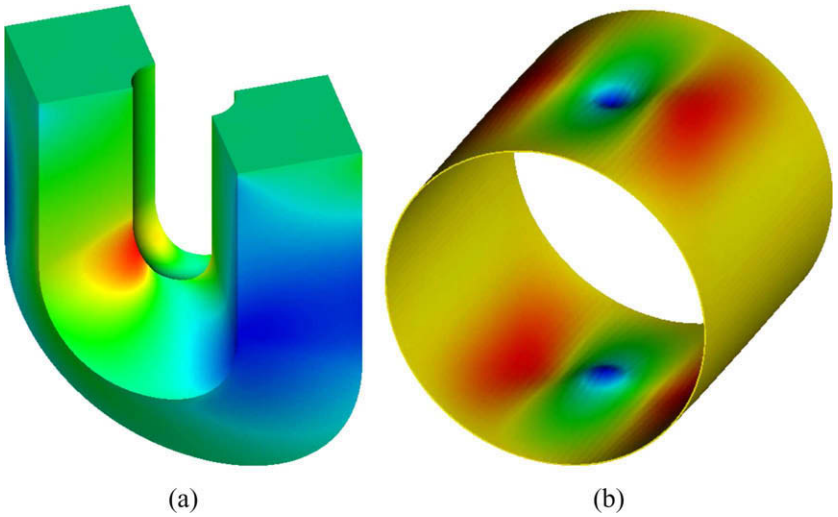


Fig. 8. Examples of some of the simple solid and thin-walled structures to which isogeometric analysis has been applied.

elevation, and C^{p-1} degree elevation, resp.). It has been shown that isogeometric analysis preserves geometry at all levels of refinement and that detailed features can be retained without excessive mesh refinement, in contrast with traditional finite element analysis. Superior accuracy to finite element analysis on a degree-of-freedom basis has been demonstrated in all cases, and indications of significantly increased robustness in vibration and wave propagation analysis has been noted (see [18,32]). So far, isogeometric analysis has been applied to simple solid parts and thin-walled structures (see Fig. 8), and its extension to more complex parts is apparent (see Fig. 9). The current range of applicability for isogeometric analysis includes more complex thin-walled structures (see Fig. 10). The challenge is to apply it to very complex modules and assemblages. Examples emanating from modern ship design are representative (see Fig. 11).

Given the incredible inertia existing in the design and analysis industries, one may reasonably ask why one should believe that the time is right to transform the technology of these industries. Here are our reasons. Recent initiatory investigations of the isogeometric concept have proven very successful. Compatibility with existing design and analysis technologies is attainable. There is interest in both the computational geometry and analysis commu-

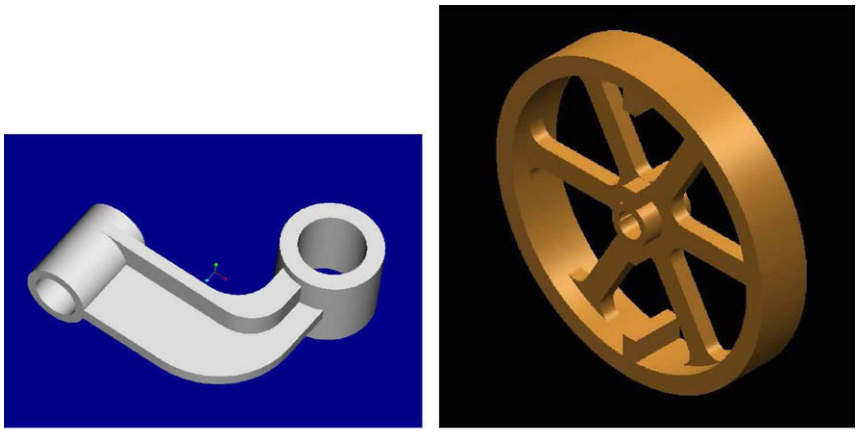


Fig. 9. Extension of the isogeometric analysis methodology to various parts and components is apparent. (Courtesy of General Dynamics/Electric Boat Corporation.)

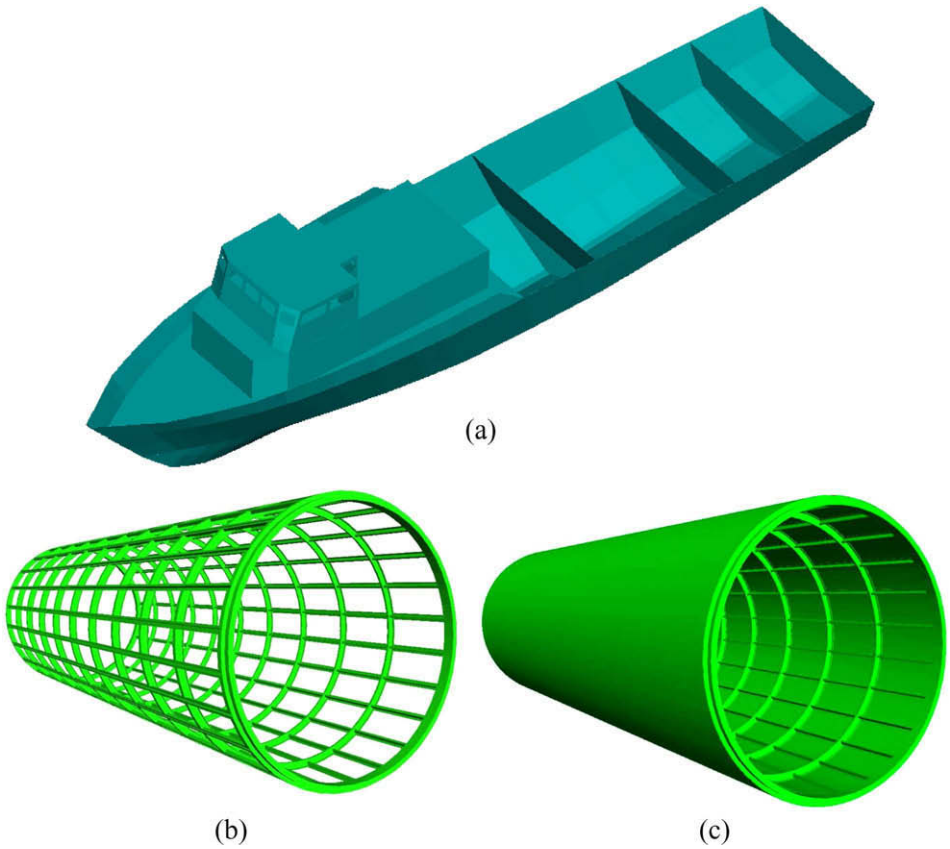


Fig. 10. Current range of applicability of isogeometric analysis includes more complex thin-walled structures.

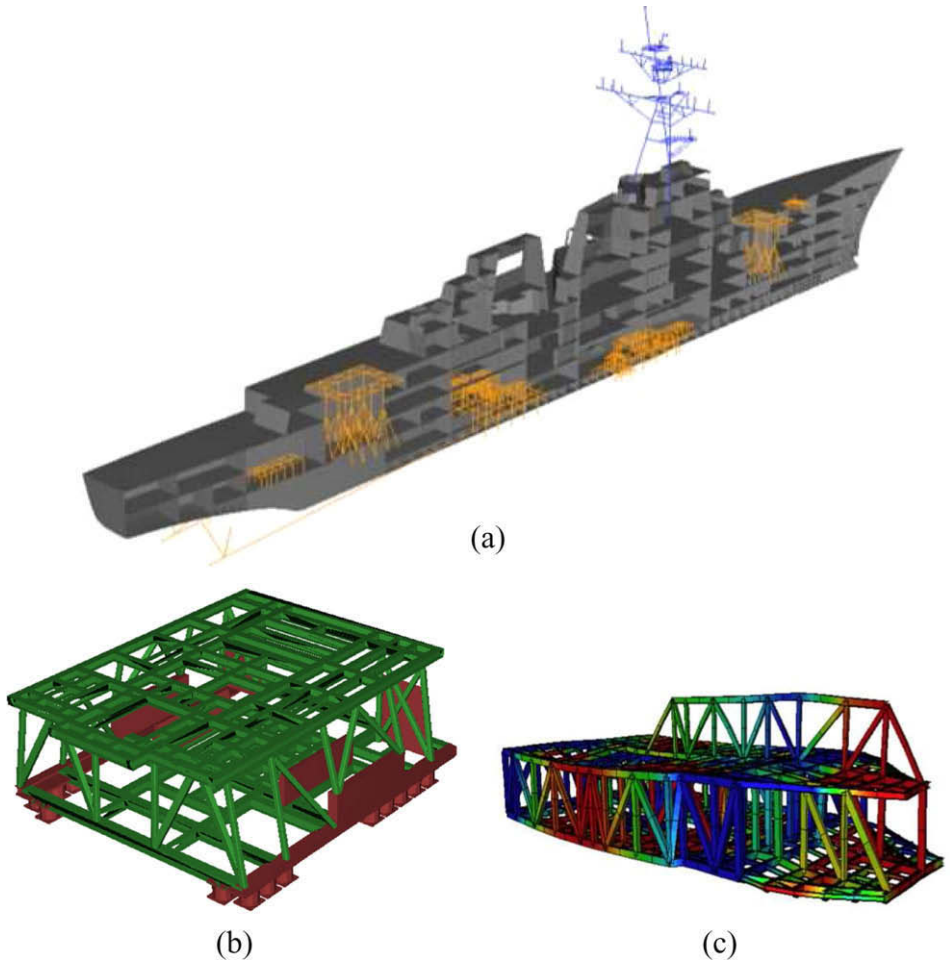


Fig. 11. Some examples from modern ship design of the types of complex modules and assemblages that present challenges to analysts. (Courtesy of S. Gordon, General Dynamics/Electric Boat Division.)

nities to embark on isogeometric research. Several minisymposia and workshops at international meetings have been held.[3] There is an inexorable march toward higher precision and greater reality. New technologies are being introduced and adopted rapidly in design software to gain competitive advantage. New and better analysis technologies can be built upon these new CAD technologies. These tools will be adopted by engineering designers because they are better, faster and cheaper. Engineering analysis can leverage these developments as a basis for the isogeometric concept.

In Section 2 we review B-splines and NURBS. In Section 3, as a prelude to T-splines, we describe PB-splines, an unstructured "meshless" spline technology. In Section 4, we present T-splines, and in Section 5 we apply T-splines to some basic problems of computational fluid and structural mechanics. In Section 6 we draw conclusions. Appendix A tabulates T-spline data for a simple mesh of a plate with a hole and Appendix B describes the process of compatibly merging two NURBS patches into a single T-spline mesh.

2. B-splines and NURBS

We begin with a review of B-splines and NURBS with particular emphasis on the features which are important for understanding the generalization to T-splines.

[3] For example, the 7th World Congress on Computational Mechanics; the 9th US National Congress on Computational Mechanics, the Seventh International Conference on Mathematical Methods for Curves and Surfaces, and the 6th International Conference on Computation of Shell and Spatial Structures.

2.1. B-splines

B-splines are piecewise polynomials that offer great flexibility and precision for a myriad of modeling applications. They are built from a linear combination of ***basis functions*** that span a corresponding ***B-spline space***. These basis functions are locally supported and have continuity that is easily controlled. Members of the resulting space have continuity properties that follow directly from those of the basis. As we introduce the basic machinery needed to build B-splines, we will attempt to foreshadow our needs in a T-spline setting. Notation will necessarily be overloaded as the complexity of the method builds, but every attempt will be made to call attention to any shifts in the notational convention as they arise.

2.1.1. Knot vectors and basis functions

Univariate B-spline basis functions are constructed from a ***knot vector***. A knot vector is a non-decreasing sequence of coordinates in the parameter space, written $\Xi = \{\xi_1, \xi_2, \ldots, \xi_{n+p+1}\}$, where $\xi_i \in \mathbb{R}$ is the ith knot, i is the knot index, $i = 1, 2, \ldots, n+p+1$, p is the polynomial degree, and n is the number of basis functions which comprise the B-spline. More than one knot can be located at the same location in the parameter space. A knot vector is said to be open if its first and last knots have multiplicity $p+1$. Open knot vectors are standard in the CAD literature, and we will only consider the use of open knot vectors in this paper.

B-spline basis functions for a given degree, p, are defined recursively in the parameter space by way of the knot vector, Ξ. Beginning with piecewise constants $(p = 0)$ we have

$$N_{i,0}(\xi) = \begin{cases} 1 & \text{if } \xi_i \leqslant \xi < \xi_{i+1}, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

For $p = 1, 2, 3, \ldots$, we define the basis functions by the Cox–de Boor recursion formula:

$$N_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi). \tag{2}$$

Fast, stable, and efficient algorithms exist for evaluation of B-spline basis functions and their derivatives. The de Boor algorithm is perhaps the most famous of these, see [24].

From (1) and (2), one can verify that B-spline basis functions possess the following properties:

(1) ***Partition of unity***: $\sum_{i=1}^{n} N_{i,p}(\xi) = 1, \ \xi \in [\xi_1, \xi_{n+p+1}]$.
(2) ***Pointwise nonnegativity***: $N_{i,p}(\xi) \geqslant 0, \quad i = 1, 2, \ldots, n$.
(3) ***Linear independence***: $\sum_{i=1}^{n} \alpha_i N_{i,p}(\xi) = 0 \iff \alpha_k = 0$, $k = 1, 2, \ldots, n$.
(4) ***Compact support***: $\{\xi | N_{i,p}(\xi) > 0\} \subset [\xi_i, \xi_{i+p+1}]$.
(5) ***Control of continuity***: If a knot value has multiplicity k (i.e., $\xi_i = \xi_{i+1} = \cdots = \xi_{i+k-1}$), then the basis functions are C^{p-k}-continuous at that location. When $k = p$, the basis is C^0 and interpolatory at that location.

These features are very useful in a finite element context. The first four properties ensure a well conditioned and sparse matrix. The fifth property allows for great flexibility. Not only do smooth bases lead to superior accuracy per degree-of-freedom compared with C^0-continuous bases [2,19,32], but the continuity can be reduced to better resolve steep gradients [18]. Additionally, one can use B-splines to build a basis that spans the same space as classical p-version finite elements (that is, a basis of degree p that is C^0 across element boundaries). This is the well-known Bernstein basis [36].

An example of a quadratic B-spline basis for $\Xi = \{0, 0, 0, 1, 2, 3, 4, 4, 5, 5, 5\}$ is shown in Fig. 12. The basis is interpolatory at the first and last knot values due to the use of an open knot vector, and also at $\xi = 4$, where the multiplicity of the knot value is equal to the polynomial degree. The basis is $C^{p-1} = C^1$ across element boundaries.

2.1.2. Anchors

Note that knots, $\{\xi_i\}_{i=1}^{n+p+1}$, and basis functions, $\{N_{j,p}\}_{j=1}^{n}$, are *not* in a one-to-one correspondence. As the use of open knot vectors is assumed throughout, we have $n + p + 1$ knots and n basis functions. However, it will prove convenient to identify locations in the parameter space to which basis functions are associated. We refer to these locations as *anchors*. They are defined as follows. For each $N_{i,p}$, its anchor, denoted t_i is given by

$$t_i = \begin{cases} \xi_{i+(p+1)/2} & \text{if } p \text{ is odd}, \\ \frac{1}{2}\left(\xi_{i+(p/2)} + \xi_{i+(p/2)+1}\right) & \text{if } p \text{ is even}. \end{cases} \tag{3}$$

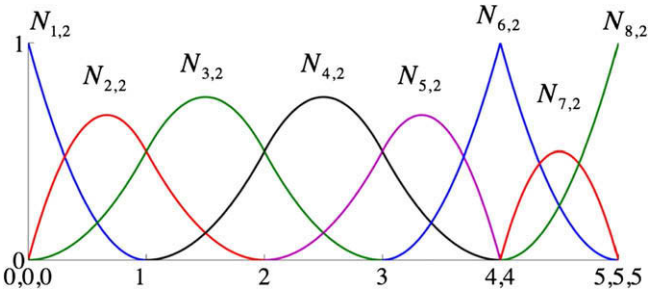


Fig. 12. Quadratic basis functions for open, non-uniform knot vector $\Xi = \{0, 0, 0, 1, 2, 3, 4, 4, 5, 5, 5\}$.

The idea is depicted in Fig. 13 for a uniform knot vector. Note that anchors associated with distinct basis functions may lie at the same parametric location.

2.1.3. B-spline curves

Let d_s denote the number of space dimensions. A ***B-spline curve*** defined in d_s-dimensional space $\mathbb{R}^{d_s}$ is defined as follows:

$$\mathbf{C}(\xi) = \sum_{i=1}^{n} \mathbf{P}_i N_{i,p}(\xi), \tag{4}$$

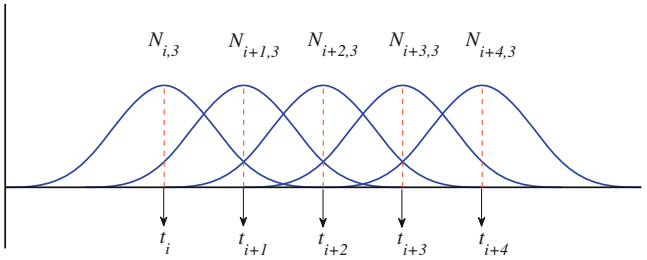
where $\mathbf{P}_i \in \mathbb{R}^{d_s}$ is a ***control point***. Piecewise linear interpolation of the control points defines the ***control polygon***.

Important properties of B-spline curves are:

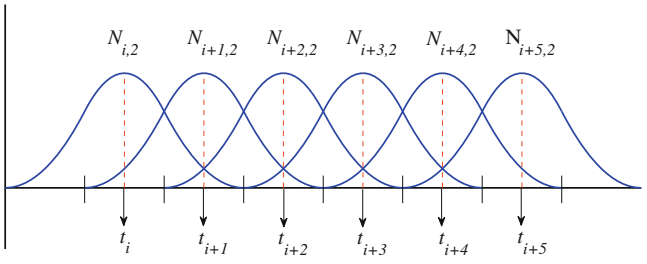
(1) ***Affine covariance***: An affine transformation of a B-spline curve is obtained by applying the transformation to its control points.
(2) ***Convex hull***: A B-spline curve lies within the convex hull of its control points (see [40] for the relationship between the convex hull and the polynomial degree of the curve).
(3) ***Variation diminishing***: A B-spline curve in $\mathbb{R}^{d_s}$ cannot cross an affine hyperplane of codimension 1 (e.g. a line in $\mathbb{R}^2$, plane in $\mathbb{R}^3$) more times than does its control polygon [37].

In addition, B-spline curves inherit all of the continuity properties of their underlying bases. This is illustrated in Fig. 14a, where we built a B-spline curve from the basis shown in Fig. 12. At the spatial location corresponding to parameter value $\xi = 4$, the B-spline curve is only continuous. The B-spline curve interpolates the control point $\mathbf{P}_6$ at this location. The use of open knot vectors ensures that the first and last control points, $\mathbf{P}_1$ and $\mathbf{P}_8$, are interpolated as well.

The control points are in one-to-one correspondence with the basis functions. This also means that the control points and the anchors are in one-to-one correspondence. Just as the anchor informs us approximately where in the parametric domain each function is



(a) Odd degree, the anchors are the knots



(b) Even degree, the anchors are the center of the knot spans

Fig. 13. Anchors. (a) In the odd case, each function is centered at a knot value. (b) In the even case, each function is centered at the center of a knot span. (In the case of a uniform knot vector, the anchor location corresponds to the maximum of the function in question, but this will not occur in the case of a general knot vector.)

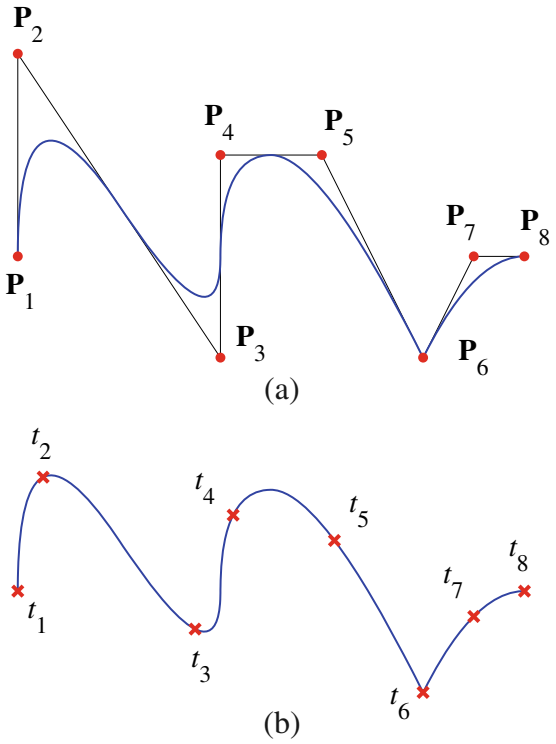


Fig. 14. Quadratic B-spline curve in $\mathbb{R}^2$. Knot vector and basis functions as in Fig. 12. (a) Curve with control points and control polygon. Control point locations are denoted by the red •. (b) Curve with anchors. Anchor locations are denoted by the red ×.

centered and takes its maximum, it also tells us approximately where the associated control point is located (see Fig. 14).

2.1.4. Multivariate B-spline functions

Multivariate B-spline basis functions are defined by tensor products of univariate B-spline basis functions. Inevitably, our notation becomes a bit cluttered as we must now indicate the parameter of interest when referring to each of the univariate entities comprising the spline. With d_p parameters, we shall let $\ell = 1, \ldots, d_p$ denote the direction of interest. While $d_p = 2$ (surfaces) and $d_p = 3$ (volumes) suffice for nearly all geometry and analysis problems, this section presents multivariate B-splines of any dimension. Note that the appearance of ℓ as a superscript is always used for disambiguation and should never be interpreted as an exponent. The B-spline basis functions are built from d_p knot vectors, one for each dimension. We write

$$\Xi^\ell = \left\{\xi^\ell_1, \xi^\ell_2, \ldots, \xi^\ell_{n_\ell+p_\ell+1}\right\}, \tag{5}$$

where p_ℓ indicates the polynomial degree along parametric direction ℓ, and n_ℓ is the associated number of functions. These univariate B-spline basis functions are denoted $\left\{N^\ell_{i_\ell,p_\ell}\right\}_{i_\ell=1}^{n_\ell}$ and are defined using knot vector Ξ^ℓ exactly as in Section 2.1.1.

We now define a multi-index $\mathbf{i} \in \mathbb{Z}^{d_p}$. In particular, we are interested in the set

$$I = \left\{\mathbf{i} = \{i_1, \ldots, i_{d_p}\} | i_\ell \in \{1, \ldots, n_\ell\}\ \forall i_\ell = 1, \ldots, d_p\right\}. \tag{6}$$

Similarly, we can denote the various polynomial degrees as $\mathbf{p} = \left\{p_1, p_2, \ldots, p_{d_p}\right\}$. Now, for each multi-index $\mathbf{i} \in I$, we can define a corresponding d_p-dimensional B-spline basis function as

$$B_{\mathbf{i},\mathbf{p}}(\boldsymbol{\xi}) \equiv \prod_{\ell=1}^{d_p} N^\ell_{i_\ell,p_\ell}(\xi^\ell), \tag{7}$$

where $\boldsymbol{\xi} = \left(\xi^1, \xi^2, \ldots, \xi^{d_p}\right)$.

It is important to note that, with this definition, there are a total of $\prod_{\ell=1}^{d_p} n_\ell$ B-spline basis functions associated with the d_p knot vectors. Multivariate B-spline basis functions inherit most of the aforementioned properties of their univariate counterparts, namely partition of unity, nonnegativity, compact support, higher continuity, and linear independence.

In Fig. 15, we have plotted $B_{\{3,3\},\{2,2\}}(\boldsymbol{\xi})$, a bivariate B-spline basis function corresponding to knot vectors $\Xi^1 = \{0,0,0,1,2,2,3,4,4,4\}$ and $\Xi^2 = \{0,0,0,1,1,2,3,4\}$, and the two univariate functions whose tensor product defines it. Note the decreased continuity that the basis function inherits from its univariate constituents; in particular, there is a break in continuity along the global knot lines $\xi^1 = 2$ and $\xi^2 = 1$.

The notion of an anchor for each function (or, equivalently, for each control point) generalizes to the multidimensional case as well. With each $B_{\mathbf{i},\mathbf{p}}$, we associate anchor $\mathbf{t}_\mathbf{i}$. The anchors of the multivariate functions are defined as the set containing d_p anchors associated with the univariate functions from which $B_{\mathbf{i},\mathbf{p}}$ was built.

2.1.5. B-spline surfaces and solids

With multivariate B-spline basis functions defined, we can now define B-spline surfaces and solids. Before proceeding, we define the ***control mesh***[4] to be our d_p-dimensional analogue of the control polygon. The control mesh is the collection of control points $\{\mathbf{P_i}\}_{\mathbf{i}\in I}$ with I as in (6). An example is given in Fig. 16 for a quadratic mesh comprised of two elements.

To define a B-spline surface, we require a multi-index $\mathbf{p} = \{p_1, p_2\}$, two knot vectors $\Xi^1 = \left\{\xi^1_1, \xi^1_2, \ldots, \xi^1_{n_1+p_1+1}\right\}$ and $\Xi^2 = \left\{\xi^2_1, \xi^2_2, \ldots, \xi^2_{n_2+p_2+1}\right\}$, and a corresponding control mesh $\{\mathbf{P_i}\}_{\mathbf{i}\in I}$. Then, the B-spline surface is defined as

$$\mathbf{S}(\boldsymbol{\xi}) = \sum_{\mathbf{i}\in I} \mathbf{P_i} B_{\mathbf{i},\mathbf{p}}(\boldsymbol{\xi}), \tag{8}$$

where the bivariate B-spline basis functions $B_{\mathbf{i},\mathbf{p}}(\boldsymbol{\xi})$ are defined by (7).

Analogously, to define a B-spline solid, we need a multi-index $\mathbf{p} = \{p_1, p_2, p_3\}$, three knot vectors $\Xi^1 = \left\{\xi^1_1, \xi^1_2, \ldots, \xi^1_{n_1+p_1+1}\right\}$, $\Xi^2 = \left\{\xi^2_1, \xi^2_2, \ldots, \xi^2_{n_2+p_2+1}\right\}$, and $\Xi^3 = \left\{\xi^3_1, \xi^3_2, \ldots, \xi^3_{n_3+p_3+1}\right\}$, and a corresponding control mesh $\{\mathbf{P_i}\}_{\mathbf{i}\in I}$. Then, the B-spline solid is defined as

$$\mathbf{V}(\boldsymbol{\xi}) = \sum_{\mathbf{i}\in I} \mathbf{P_i} B_{\mathbf{i},\mathbf{p}}(\boldsymbol{\xi}), \tag{9}$$

where $B_{\mathbf{i},\mathbf{p}}(\boldsymbol{\xi})$ are now trivariate B-spline basis functions.

From henceforth, let us simply use $\mathbf{S}$ to refer to a multivariate spline object. While there is a jump from the univariate case of curves to the multivariate case, there is little fundamental difference between different values of $d_p \geqslant 2$.

Collectively, any B-spline associated with a particular set of knot vectors, polynomial degrees, and control points is referred to as a ***patch***.[5] Each patch has its own parameter space. Large geometries are frequently built from many patches. When two patches meet, the control points coming from each side must be identical along the interface where they meet, and the corresponding knot vectors must be identical as well. Under these conditions, only C^0-continuity of the basis is achieved across the patch boundaries.

[4] The terms control mesh, control net, and control lattice are used interchangeably. We prefer the term control mesh because it corresponds to a mesh of multilinear finite elements. Note, however, that the control mesh and physical mesh are distinct objects in isogeometric analysis.

[5] In computational geometry the word "patch" refers to the case of a surface. In our discussion we expand the concept to any dimension.

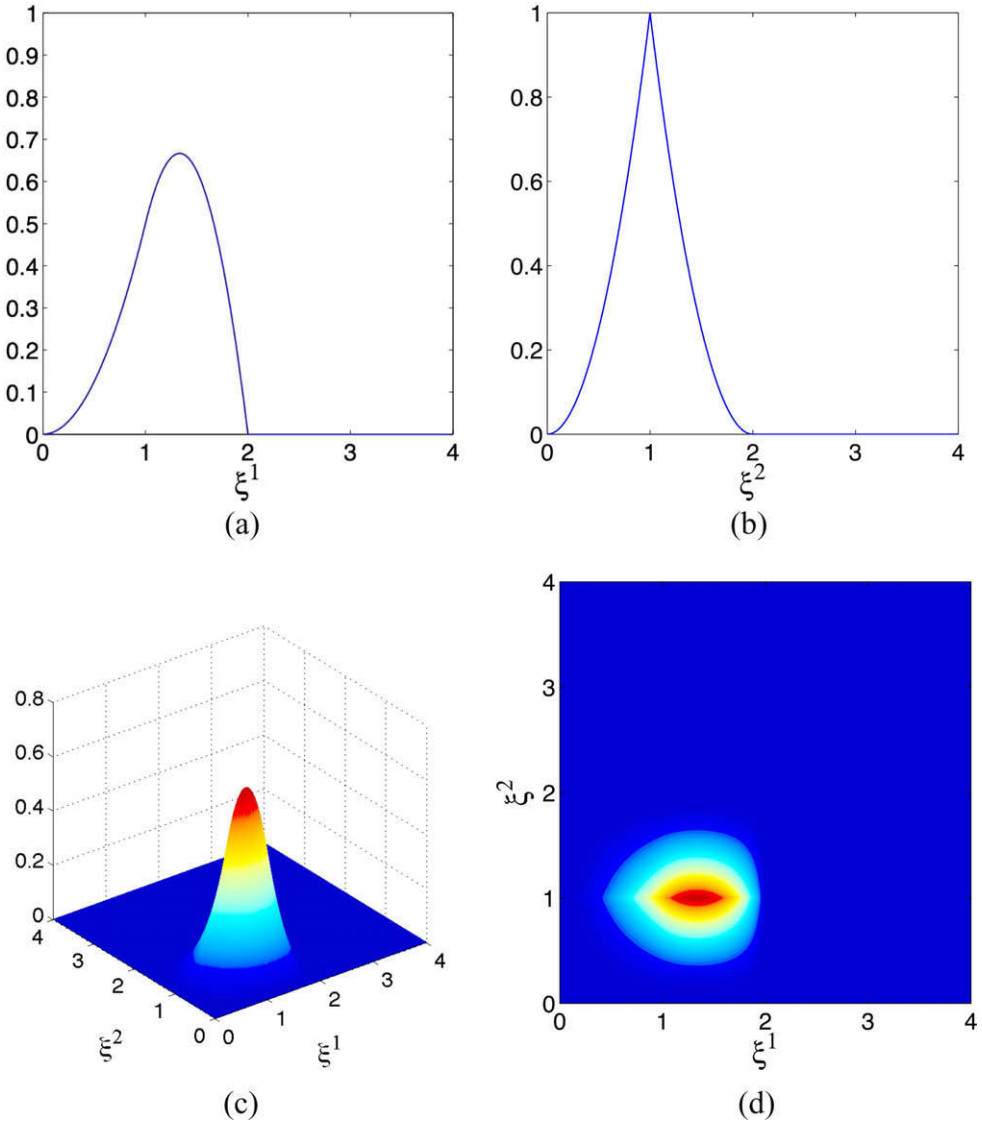


Fig. 15. Bivariate quadratic B-spline basis function $B_{\{3,3\},\{2,2\}}(\xi^1,\xi^2)$ for knot vectors $\Xi^1=\{0,0,0,1,2,2,3,4,4,4\}, \Xi^2=\{0,0,0,1,1,2,3,4\}$. (a) Univariate B-spline basis function $N^1_{3,2}(\xi^1)$. (b) Univariate B-spline basis function $N^2_{3,2}(\xi^2)$. (c) Tilted view of $B_{\{3,3\},\{2,2\}}(\xi)$. (d) Overhead view of $B_{\{3,3\},\{2,2\}}(\xi)$.

2.1.6. The index, parameter, and physical spaces

Fig. 16 shows both the physical mesh and control mesh in the ***physical space***. It is also informative to consider the domain of the mesh, a subset of the ***parameter space***, shown in Fig. 17, which is simply the pre-image of the physical mesh. The B-spline mapping takes each point in the parameter space to a point in the physical space, and the images of the knot lines under the NURBS mapping bound the physical elements.

The concept of the ***index space*** was introduced in Hughes et al. [30]. It is created by plotting the knots at equally spaced intervals, regardless of their actual spacing, and labeling each knot line with its index value. This point of view is extremely useful for developing algorithms, as well as for building intuition. For example, in the index space, it is easy to identify the knot lines at which the support of any given function will begin or end, as well as which functions have support within any given element. This is trickier in the parameter space as some of the knots may have the same value.

As mentioned previously, the parameter space definition of anchors, given by (3), may lead to anchors associated with distinct basis functions lying at identical parametric locations. Index space enables us to define unique locations of anchors associated to distinct basis functions. Corresponding to (3), we assign to each one-dimensional B-spline basis function $N_{i,p}$ a unique anchor in index space, s_i, given by

$$s_i = i + (p+1)/2. \tag{10}$$

Note that when p is odd, s_i has integer value, whereas when p is even, s_i is real and the average of consecutive integers. Multi-dimensional B-spline basis functions have coordinate anchors whose components in each direction are given by (10).

Fig. 18 shows the index space for the surface in Fig. 16. We have plotted the anchors of the functions. As $p=2$, these fall in the center of the cells. Were this an odd degree, the anchors would fall at the intersections. In both cases, *the anchor for each function lies at the exact center of its support in the index space*. There is a continuous, piecewise linear mapping from index space to parameter space that is surjective but generally not bijective. This mapping renders definition (10) consistent with (3).

2.2. Non-uniform rational B-splines

There are geometric entities in $\mathbb{R}^{d_s}$ that cannot be modeled exactly by piecewise polynomials. Many important ones, however, can be obtained through a projective transformation of a

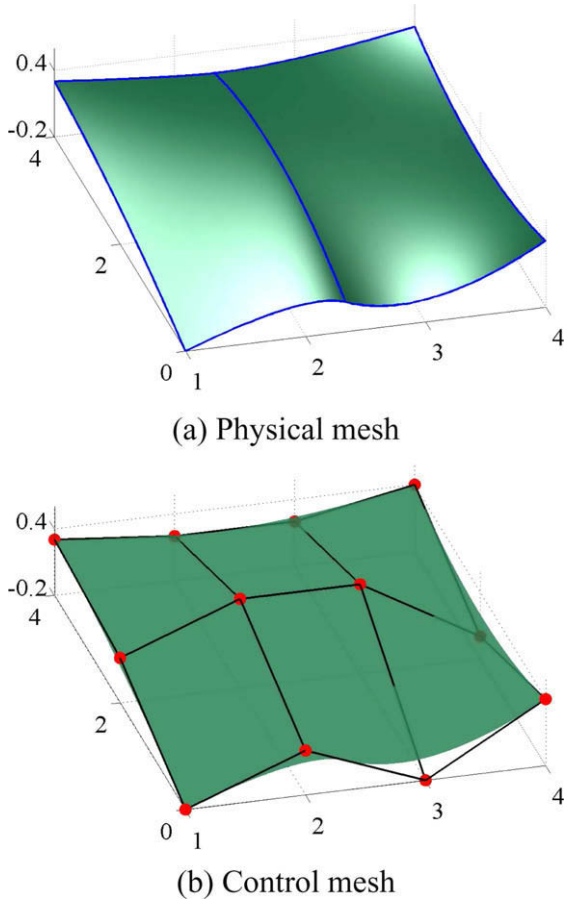


Fig. 16. A biquadratic B-spline surface generated from $\Xi^1 = \{0,0,0,1,2,2,2\}$ and $\Xi^2 = \{0,0,0,1,1,1\}$. (a) The physical mesh is comprised of the image of the parent domain under the geometrical mapping. The curves in the physical mesh are the images of the knot lines. In this case, we have two elements. (b) The control mesh is comprised of the red control points, the black lines connecting the control points, and the images of the knot spans under a bilinear mapping. This is completely analogous to a finite element mesh of four-node bilinear elements. Note that the control points are not generally interpolated by the surface itself. It is extremely important to distinguish between the physical mesh and the control mesh.

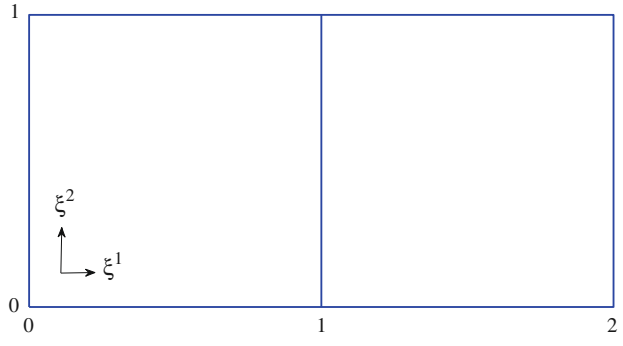


Fig. 17. The domain of the mesh, a subset of the parameter space, corresponding to the biquadratic B-spline surface seen in Fig. 16. The domain of the mesh is the pre-image of the physical mesh.

corresponding B-spline entity in $\mathbb{R}^{d_s+1}$ yielding a *rational B-spline*. In particular, conic sections, such as circles and ellipses, can be exactly constructed by projective transformations of piecewise quadratic curves.

The construction of a rational B-spline curve in $\mathbb{R}^{d_s}$ begins by choosing a set of control points $\{\mathbf{P}_i^w\}$ for a B-spline curve in $\mathbb{R}^{d_s+1}$ with knot vector $\Xi = \{\xi_1, \ldots, \xi_{n+p+1}\}$. These are referred to as the

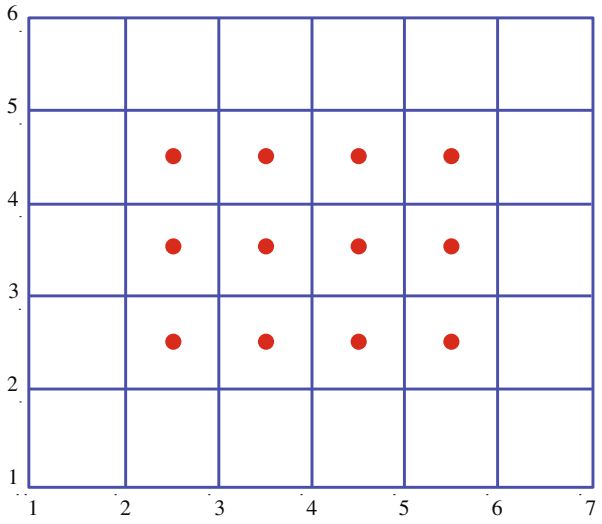


Fig. 18. The index space corresponding to the biquadratic B-spline surface seen in Fig. 16. The anchor for each function is shown as a red circle. For this even-degree case ($p = 2$), the anchors fall in the centers of the cells. For an odd polynomial degree, the anchors would fall at the intersections of knot lines. Note that, regardless of degree, an anchor falls at the center of the support of a function in the index space.

"projective" control points. The control points in $\mathbb{R}^{d_s}$ are then derived from the following relations:

$$(\mathbf{P}_i)_j = \frac{(\mathbf{P}_i^w)_j}{w_i}, \quad j = 1, \ldots, d_s, \tag{11}$$

$$w_i = (\mathbf{P}_i^w)_{d_s+1}, \tag{12}$$

where $(\mathbf{P}_i)_j$ is the jth component of the vector $\mathbf{P}_i$ and w_i is referred to as the ith weight. It is common practice to require all weights to be nonnegative. This assures that the convex hull property holds. Then, the NURBS (non-uniform rational B-spline) curve is defined by

$$\mathbf{C}(\xi) = \sum_{i=1}^{n} \mathbf{P}_i R_{i,p}(\xi), \tag{13}$$

where the projected NURBS basis functions $R_{i,p}(\xi)$ are defined by

$$R_{i,p}(\xi) = \frac{w_i N_{i,p}(\xi)}{\sum_{j=1}^{n} w_j N_{j,p}(\xi)}. \tag{14}$$

Multivariate NURBS basis functions are defined analogously using d_p knot vectors, Ξ^ℓ, and a set of weights $\{w_\mathbf{i}\}_{\mathbf{i}\in I}$ where I is the appropriate index set. Given $\mathbf{p}$ and $\mathbf{i}$, the corresponding multivariate NURBS basis function is defined as

$$R_{\mathbf{i},\mathbf{p}}(\boldsymbol{\xi}) = \frac{w_\mathbf{i} B_{\mathbf{i},\mathbf{p}}(\boldsymbol{\xi})}{\sum_{\mathbf{j}\in I} w_\mathbf{j} B_{\mathbf{j},\mathbf{p}}(\boldsymbol{\xi})}. \tag{15}$$

NURBS surfaces and solids are then defined in the same manner as B-splines surfaces and solids, namely

$$\mathbf{S}(\boldsymbol{\xi}) = \sum_{\mathbf{i}\in I} \mathbf{P}_\mathbf{i} R_{\mathbf{i},\mathbf{p}}(\boldsymbol{\xi}). \tag{16}$$

NURBS inherit all of the important properties from their piecewise polynomial counterparts. These include

(1) Partition of unity.
(2) Pointwise nonnegative.
(3) Affine covariance.
(4) The convex hull property.

and the continuity of a NURBS object follows from that of the basis in exactly the same manner as for B-splines. It is common practice

to require all weights to be nonnegative; otherwise, the convex hull property may be violated.

A B-spline object that does not use weights (i.e., the form presented in Section 2.1) is called a *polynomial* B-spline to distinguish it from a rational B-spline. Note that setting all the weights to be equal reduces a rational B-spline to a polynomial B-spline. In this paper, we refer to polynomial B-splines as B-splines and rational B-splines as NURBS. As in the case of B-splines, we refer collectively to any NURBS object associated with a single set of knot vectors, polynomial degrees, and control points as a "patch".

2.3. Refinement

Some of the ways in which NURBS can be refined were discussed in [18]. For the purposes of the present work, we recall only ***knot insertion***, as that will be the focus of our investigation of T-splines. We will begin with the case of a B-spline curve, and then extend it to the multivariate case. Exactly the same process applies to NURBS, but we refine the projective control points rather than the control points themselves.

2.3.1. Knot insertion

The mechanism for implementing *h*-refinement is knot insertion. Knots may be inserted without changing a curve geometrically or parametrically. Given a knot vector $\Xi = \{\xi_1, \xi_2, \ldots, \xi_{n+p+1}\}$, let $\overline{\Xi} = \{\overline{\xi}_1 = \xi_1, \overline{\xi}_2, \ldots, \overline{\xi}_{n+m+p+1} = \xi_{n+p+1}\}$ be an *extended* knot vector such that $\Xi \subset \overline{\Xi}$. The new $n+m$ basis functions are formed as before by applying recursion formulas (1) and (2)to the new knot vector $\overline{\Xi}$. The new $n+m$ control points, $\overline{P} = \{\overline{\mathbf{P}}_1, \overline{\mathbf{P}}_2, \ldots, \overline{\mathbf{P}}_{n+m}\}^T$, are formed from the original control points, $P = \{\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_n\}^T$, by a linear transformation

$$\overline{P} = \mathbf{T}^p P, \tag{17}$$

where

$$T_{ij}^0 = \begin{cases} 1 & \text{if } \overline{\xi}_i \in [\xi_j, \xi_{j+1}), \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

and

$$T_{ij}^{q+1} = \frac{\overline{\xi}_{i+q} - \xi_j}{\xi_{j+q} - \xi_j} T_{ij}^q + \frac{\xi_{j+q+1} - \overline{\xi}_{i+q}}{\xi_{j+q+1} - \xi_{j+1}} T_{ij+1}^q \quad \text{for } q = 0, 1, 2, \ldots, p-1. \tag{19}$$

Knot values already present in the knot vector may be repeated as above but the continuity of the basis will be reduced. Continuity of the curve is preserved by choosing the control points as in (17)–(19).

If instead of a curve we wish to insert knots into one of the knot vectors, Ξ^ℓ, of a surface or solid, we utilize the same procedure. The matrix $\mathbf{T}^p$ is generated exactly as above by considering Ξ^ℓ and $\overline{\Xi}^\ell$. The new control points would be generated by applying (17) to each row or column in the control mesh. For example, let us consider a B-spline surface into which we insert m knots in the parametric direction $\ell = 1$. That is, Ξ^1 will be refined, but Ξ^2 remains the same. The new control points for the surface are given by

$$\overline{\mathbf{P}}_{ik} = \sum_{j=1}^{n_1} T_{ij}^p \mathbf{P}_{jk}, \tag{20}$$

for $i = 1, 2, \ldots, n_1 + m$ and $k = 1, 2, \ldots, n_2$. The new tensor product basis functions are generated in the standard way from $\overline{\Xi}_1$ and $\overline{\Xi}_2$.

2.3.2. Function subdivision

There is another way to view this process that provides some insight into the logic behind T-splines. Let us denote by $\mathcal{S}$ the space of all curves that can be built from our original knot vector, Ξ, and let $\overline{\mathcal{S}}$ be the space of all curves that can be built from the extended knot vector, $\overline{\Xi}$. Clearly, if both our geometry and its parameterization are to be preserved, then we must have that

$$\mathcal{S} \subset \overline{\mathcal{S}}. \tag{21}$$

A natural way to ensure that this is true is to insist that each of our original basis functions can be expressed as a linear combination of

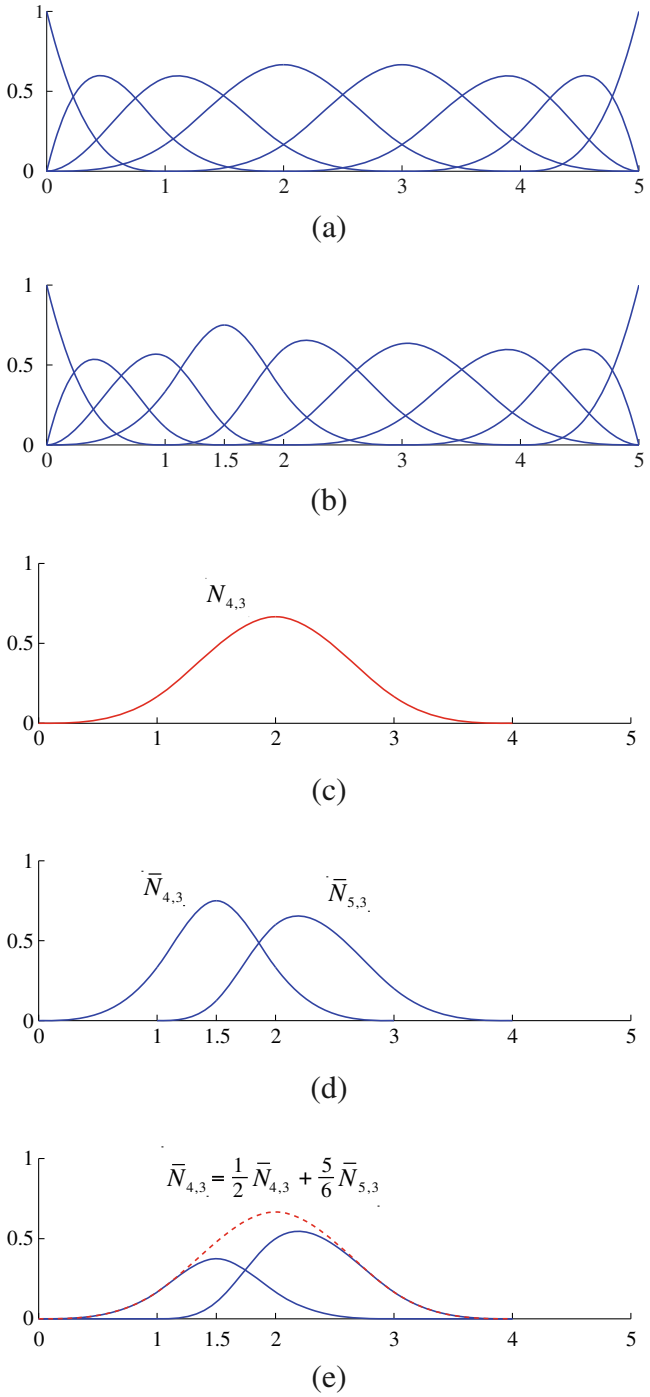


Fig. 19. Function subdivision. (a) B-spline basis functions generated from $\Xi = \{0,0,0,0,1,2,3,4,5,5,5,5\}$ and $p = 3$. (b) New B-spline basis functions generated from $\Xi = \{0,0,0,0,1,1.5,2,3,4,5,5,5,5\}$ and $p = 3$. (c) As an example, consider $N_{4,3}$ from our original basis, which has support on $[0,4]$ and approaches the origin with a second derivative of zero. (d) The new basis has two functions, $\overline{N}_{4,3}$ and $\overline{N}_{5,3}$, which also have support on $[0,4]$ and approach the origin with zero second derivative. (e) We can represent $N_{4,3}$ as a linear combination of $\overline{N}_{4,3}$ and $\overline{N}_{5,3}$ with coefficients of 1/2 and 5/6, respectively.

the functions from the refined basis. This notion of ***function subdivision*** is already present in (17). To see this, let $N = \{N_1(\xi), N_2(\xi), \ldots, N_n(\xi)\}^T$ be a vector containing the basis functions generated from Ξ, and let $\overline{N} = \{\overline{N}_1(\xi), \overline{N}_2(\xi), \ldots, \overline{N}_{n+m}(\xi)\}^T$ be a vector containing the basis generated from $\overline{\Xi}$, where we have suppressed the polynomial degree from our notations for clarity. We have the following expressions for our B-spline curve:

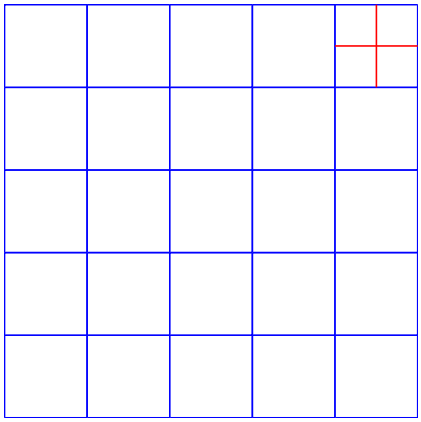
$$\mathbf{C}(\xi) = P^T N = \overline{P}^T \overline{N} = (\mathbf{T}^p P)^T \overline{N} = P^T \left((\mathbf{T}^p)^T \overline{N} \right) \tag{22}$$

and thus,

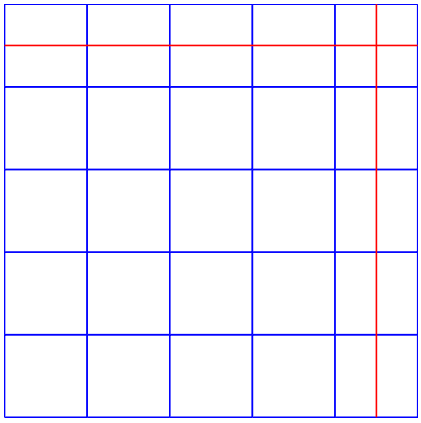
$$N = (\mathbf{T}^p)^T \overline{N}. \tag{23}$$

An example of this concept is shown in Fig. 19. We begin with the knot vector $\Xi = \{0,0,0,0,1,2,3,4,5,5,5,5\}$, from which we generate a cubic basis, shown in Fig. 19a. After inserting a new knot value of $\bar{\xi} = 1.5$ into the knot vector, we could use (1) and (2) and generate a new basis, shown in Fig. 19b. Each function in the original basis, like the one in Fig. 19c, can be represented as a linear combination of functions from the refined basis. In this case, the two functions in Fig. 19d can be combined to reproduce our original function, as in Fig. 19e.

The relationship expressed in (23) is merely one way to look at knot insertion. Nothing is gained or lost in the univariate setting by considering refinement of the functions, as opposed to refining the knot vector and generating a new set of functions directly from them. However, refining functions is a more general concept, and it will lead to more flexibility when we revisit refinement below in the context of T-splines.



(a) Local refinement



(b) Global refinement

Fig. 20. (a) There are many instances in which we would like to locally refine an initial NURBS mesh by subdividing an individual element. (b) Unfortunately, knot insertion is a global process that necessitates the propagation of the refinement throughout the domain.

2.4. Limitations of a NURBS-based framework

NURBS have been, and continue to be, widely used by designers. As mentioned, they are a standard in the CAD community and have recently been used with a great deal of success as a basis for isogeometric analysis. However, they have several drawbacks that we would like to avoid. One is that they generally achieve only C^0-continuity across patch boundaries. (In special circumstances, they can achieve higher continuity.) However, if two NURBS surfaces do not share a common boundary curve, they cannot even achieve C^0 continuity without perturbing at least one of the surfaces. Another drawback is that the joining of two patches that were created separately can be problematic, frequently requiring the insertion of many knots from one patch into the other, and vice versa. This is a significant disadvantage of NURBS: knot insertion is a global operation. When we refine by inserting knots into the knot vectors of a surface, the knot lines extend throughout the entire domain (see Fig. 20b). In [18], one approach to local refinement was considered that involved multiple patches. This technique, however, required the use of constraint equations which are inconvenient to implement, and refinement still propagates throughout a given patch. This is a problem in both geometrical modeling and in analysis. What we would like is a technology that allows us to use the smooth functions and geometrical flexibility of NURBS, while permitting local refinement.

3. PB-splines: an unstructured, meshless spline technology

As a prelude to the description of T-splines, we introduce the concept of ***point-based splines***, or simply ***PB-splines*** [44]. Though we will not actually compute with PB-splines, we feel that they have the potential to have an impact in the area of meshless methods. Here, we examine them as a generalization of the concept of NURBS, and discuss both what is to be gained and lost by their use.

3.1. Local knot vectors

Thus far, we have built spline basis functions beginning with a knot vector and the function definitions of (1) and (2). From the building blocks of these univariate, non-rational B-spline functions, multivariate, rational NURBS functions can be constructed. Note that this process has already taken on a global perspective – knot vectors alone describe the entire parametric domain. Recall, however, that the support of a B-spline function, $N_{i,p}$, is contained in $[\xi_i, \xi_{i+p+1}]$. As such, the only knots that contribute to the definition of $N_{i,p}$ are $\{\xi_i, \xi_{i+1}, \ldots, \xi_{i+p+1}\}$. Thus, if we are only interested in $N_{i,p}$, we have no need for the global knot vector. We can instead define a ***local knot vector***,

$$\Xi_i^{loc} = \{\xi_{i+j}\}_{j=0}^{p+1} \tag{24}$$

and use it in conjunction with (1) and (2) to define $N_{i,p}$, without altering the result in any way.

To illustrate the notion of a local knot vector, consider first the global knot vector $\Xi = \{0,0,0,0,1,2,3,4,5,5,5,5\}$ and the associated basis for $p = 3$, shown in Fig. 21a. We take $N_{4,3}$ to be the function of interest, shown in Fig. 21b. This function may be characterized by the local knot vector $\Xi_4^{loc} = \{0,1,2,3,4\}$. This local knot vector alone is enough to define $N_{4,3}$ through (1) and (2). That is, we do not even have to know the polynomial degree. A local knot vector will always contain exactly $p+2$ knots, and so we

can infer the degree of the function from the length of the local knot vector.

Thus, although somewhat cumbersome and redundant in the case of B-splines, one could choose to abandon the original notion of a single global knot vector and utilize a set of local knot vectors to define the basis functions. Of course, in a univariate B-spline setting, all we would have gained is additional overhead and notational complexity. Still, we have effectively deconstructed the spline into its basic parts. We now must decide how to put them back together.

3.2. PB-splines

Instead of beginning with a B-spline and extracting the local knot vector and associated basis function, consider the case where we are provided some *local* knot vector of arbitrary length

$$\Xi_\alpha = \{\xi_i\}_{i=1}^{m_\alpha}, \tag{25}$$

where we have dropped the superscript "*loc*" as we no longer have any notion of a global knot vector and, hence, every knot vector will be interpreted as a local knot vector. We associate with it a single function $N_\alpha(\xi)$ of degree $p_\alpha = m_\alpha - 2$. Eqs. (1) and (2) guarantee that $N_\alpha(\xi)$ is a unique and well-defined function. We will refer to it as a ***blending function*** rather than a basis function as no space has yet been defined. Given many such functions, without assuming that the various knot vectors are related in any way, we could assign coefficients and create a curve from them.

In the multivariate case, we will again let $\ell = 1, \ldots, d_p$ denote a given parametric direction. In order to define a function, $B_\alpha(\boldsymbol{\xi})$, we now require d_p knot vectors, $\Xi_\alpha^\ell = \left\{\xi_1^\ell, \xi_2^\ell, \ldots, \xi_{m_\ell}^\ell\right\}$. From these d_p knot vectors, we compute d_p univariate functions $N_\alpha^\ell(\xi)$ such that

$$B_\alpha(\boldsymbol{\xi}) \equiv \prod_{\ell=1}^{d_p} N_\alpha^\ell(\xi^\ell). \tag{26}$$

We will consolidate notation by collecting the knot vectors in each parametric direction into a set, denoted $\boldsymbol{\Xi}_\alpha$, given by

$$\boldsymbol{\Xi}_\alpha = \left\{\Xi_\alpha^\ell\right\}_{\ell=1}^{d_p}, \tag{27}$$

such that each $\boldsymbol{\Xi}_\alpha$ uniquely defines a function $B_\alpha(\boldsymbol{\xi})$. To each $\boldsymbol{\Xi}_\alpha$ we will also associate the support of (26), namely

$$\mathbf{D}_\alpha = \bigotimes_{\ell=1}^{d_p} [\xi_{1_\ell}, \xi_{m_\ell}]. \tag{28}$$

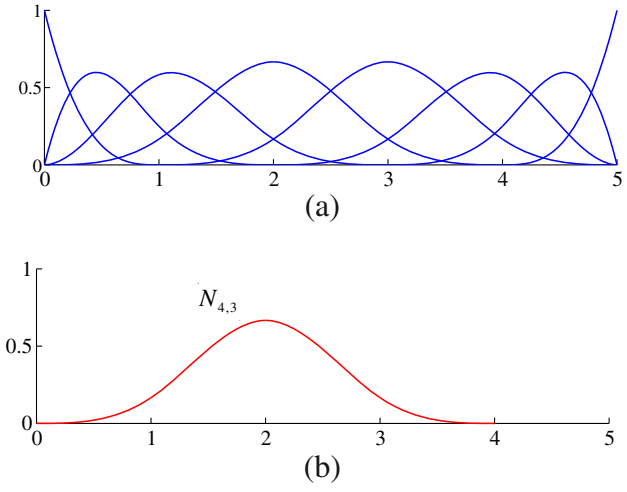


Fig. 21. (a) Eight B-spline basis functions for $\Xi = \{0,0,0,0,1,2,3,4,5,5,5,5\}$ and $p = 3$. (b) Basis function $N_{4,3}$ extracted from global knot vector associated with local knot vector $\Xi_4^{loc} = \{0,1,2,3,4\}$.

Let us define a new index set, A, containing all α for which we have a set of local knot vectors and corresponding blending functions. In order to build a PB-spline, we must first define a domain, $\mathbf{D}$. Though $\mathbf{D}$ may be irregular, we must have that

$$\mathbf{D} \subset \bigcup_{\alpha \in A} \mathbf{D}_\alpha. \tag{29}$$

The only requirement for $\mathbf{D}$ is that $\sum_{\alpha \in A} B_\alpha(\boldsymbol{\xi}) > 0$ for all $\boldsymbol{\xi} \in \mathbf{D}$. We can now define blending functions on $\mathbf{D}$ by creating a partition of unity:

$$R_\alpha(\boldsymbol{\xi}) = \frac{B_\alpha(\boldsymbol{\xi})}{\sum_{\beta \in A} B_\beta(\boldsymbol{\xi})}. \tag{30}$$

The blending functions will constitute a basis if their linear independence can be established. To each $\alpha \in A$ we assign a control point, $\mathbf{P}_\alpha$, and define a PB-spline by

$$S(\boldsymbol{\xi}) = \sum_{\alpha \in A} \mathbf{P}_\alpha R_\alpha(\boldsymbol{\xi}). \tag{31}$$

Note that it no longer makes sense to refer to a "control mesh". The term ***control cloud*** seems more appropriate as there need be no organization or topology to the set of control points. Each point multiplies one function. Of course, choosing points completely at

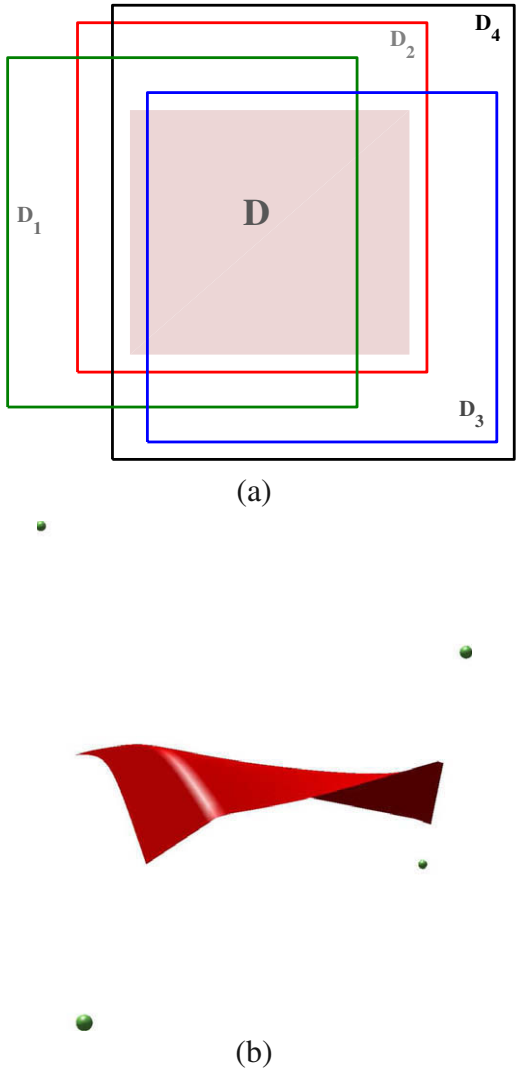


Fig. 22. PB-splines. (a) The supports, $\mathbf{D}_\alpha$, and the parameter space for the spline, $\mathbf{D}$. (b) The corresponding biquadratic PB-spline with its four control points.

random will result in degenerate geometries, but the concept remains: there is no clear ordering of the control points as in the case of NURBS. The notion of an anchor for each function also still persists. As with the control points, these anchors are completely unrelated to each other. As such, they do not appear to be useful here.

An example of a PB-spline surface in $\mathbb{R}^3$ is shown in Fig. 22. The supports of the functions and the global domain are shown in the parameter space in Fig. 22a. In Fig. 22b, we see the PB-spline, along with the control cloud. In this case, the functions used are all biquadratic, but nothing precludes using different combinations of polynomial degrees for each function.

Several of the nice properties of NURBS persist in PB-splines, now in a completely unstructured environment. This construction results in objects that possess the convex hull property. They have at least as many continuous derivatives as implied by the local knot vectors from which they were built, and the spaces can be locally enriched by adding more blending functions wherever they are desired. The space of blending functions is even complete in that it is capable of reproducing arbitrary linear polynomials (any isoparametric basis that is also a partition of unity has this property; see [29]).

Unfortunately, several undesirable features emerge. First, there is no notion of an element in the classical sense. While the support of a function, R_α, is easily discerned[6] from its local knot vector Ξ_α, there is no clearly identifiable region that we might call an element. Also, as each function has been constructed without regard for any other function, it is very difficult to speak of refinement in the traditional sense. Though new blending functions may be added at will, it is not at all clear that new control points could be selected in such a way as to preserve the original geometry. Finally, there is no clear way of assessing the approximability of discretization spaces comprised of PB-splines. Thus, PB-splines have some nice properties, but deficiencies as well. For example, we have lost any sense of structure. With T-splines, we maintain much of the freedom of PB-splines, recover the orderly structure of NURBS, and preserve many the desirable properties of both.

4. T-splines: smooth functions, geometrical flexibility, local refinement

T-splines combine much of the flexibility of PB-splines with the topology and structure of NURBS [44]. They allow us to build spaces that are complete up to a desired polynomial degree, as smooth as an equivalent NURBS basis, and capable of being locally refined in a manner similar to PB-splines but while keeping the original geometry and parameterization unchanged. The properties that make T-splines useful for geometrical modeling also make them useful for finite element analysis.

In [44], T-splines were defined for bicubic surfaces. We have generalized this concept to three dimensions and arbitrary degree p. As such, we have adopted slightly different definitions. We begin with the two-dimensional case.

4.1. Control points, knot vectors, anchors, and the T-mesh

With NURBS, we used global knot vectors from which all of the functions were defined. With PB-splines, each function had its own local knot vectors that remained completely independent of the other functions and their knot vectors. For T-splines, we strike a balance between the two cases. Each function has its own local knot vector, but these local knot vectors are inferred from a global structure, the T-mesh, that encodes a topology and parameterization for the entire T-spline object.

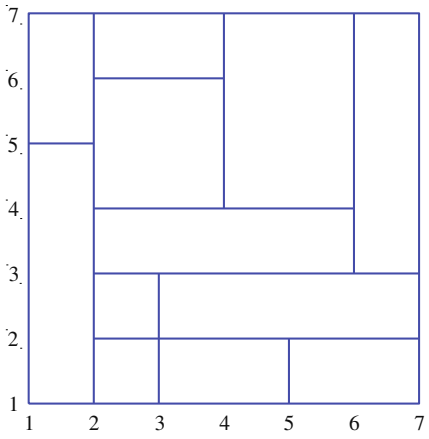


Fig. 23. Example of a T-mesh. Each line in the mesh corresponds to a knot value.

In Section 2.1.6, NURBS basis functions were defined from the parameter space and the index space was utilized as an auxiliary tool. With T-splines, for reasons that will be made clear later on, this order will be reversed. We begin by defining an ***index space*** version of a ***T-mesh*** as a rectangular tiling of a region in $\mathbb{R}^2$ such that each edge of every rectangle has positive integer value. An example of a T-mesh is given in Fig. 23. Note its similarity to the index space representation of a NURBS, except that now vertices connecting three edges, referred to as ***T-junctions***, are allowed. We now choose a degree p for the T-spline.[7] Then, for each index space direction i and each integer j for which some edge in the T-mesh has value j in direction i, we choose a ***knot*** $\xi_j^i \in \mathbb{R}$. We require that if $k > j$, then $\xi_k^i \geqslant \xi_j^i$. Subsequent knots in the same direction may have the same value. Thus, lines in the index space version of a T-mesh will correspond to knot indices. Again, note the similarity to the index space representation of NURBS, where the knots were plotted equally spaced.

At this point, though very similar, we must treat the cases of even and odd degrees separately. Let us begin with the odd-degree case. For each vertex in the T-mesh, we now define an ***anchor*** $\mathbf{s}_\alpha$ at that point. Each anchor will be used to infer local knot vectors which in turn will define a T-spline blending function in the same way as PB-splines. For each index space direction i, we create a ***local knot vector*** Ξ_α^i corresponding to anchor $\mathbf{s}_\alpha$. At first, this vector is empty. Next, we take the anchor's location $\mathbf{s}_\alpha = \{i,j\}$ and place ξ_i^1 and ξ_j^2 in Ξ_α^1 and Ξ_α^2, respectively. These knots will remain at the middle of the local univariate knot vectors. Next, we travel horizontally to the right of the anchor, record the value k of the first orthogonal edge encountered, and place ξ_k^1 at the end of Ξ_α^1. We continue this process until we have encountered a total of $(p+1)/2$ orthogonal edges to the right. Then, we travel horizontally to the left of the anchor, record the value k of the first orthogonal edge encountered, and place ξ_k^1 at the beginning of Ξ_α^1. We continue this process until we have encountered a total of $(p+1)/2$ orthogonal edges to the left. The remaining knots for the second index space direction are found in a similar manner by traveling vertically from the anchor, recording the first $(p+1)/2$ orthogonal edges encountered downwards and upwards of the anchor, and adding the corresponding knots into Ξ_α^2. If at any point no more orthogonal edges are encountered, but there still remain spaces for knots to be added, we repeat the last recorded knot

[6] The fact that the support of the function is so easily identified may make this a very useful meshless technology.

[7] It is possible to choose different polynomial degrees for each index space direction, but for the purposes of this paper we assume the same degree in all directions.

in that direction for each remaining space in the knot vector. This has the effect of producing a behavior similar to that of open knot vectors without requiring them explicitly.

As an example, consider the T-mesh shown in Fig. 24 with $p = 3$, which corresponds to a cubic T-spline. Every line in the T-mesh corresponds to a knot value, and every vertex is the anchor for a control point, two of which have been drawn and labeled in Fig. 24a. In order to build the corresponding blending functions, we need to find the local multivariate knot vectors.

Let us consider the anchor $\mathbf{s}_\alpha = \{4, 4\}$. We begin by placing the knots ξ_4^1 and ξ_4^2 corresponding to the anchor's coordinates into the anchor's knot vectors Ξ_α^1 and Ξ_α^2, respectively. To find the remaining local knots for the first parametric direction, we travel horizontally from the anchor, recording the first $(p+1)/2$ orthogonal edges that we encounter to the left of the anchor, and the first $(p+1)/2$ orthogonal edges encountered to its right. For $\mathbf{s}_\alpha$, this yields $\Xi_\alpha^1 = \{\xi_1^1, \xi_2^1, \xi_4^1, \xi_6^1, \xi_7^1\}$. In the second parametric direction, we do likewise in the vertical direction. It does not matter that the anchor lies at a T-junction in the grid, we still search outward from the anchor in the vertical direction, recording the first $(p+1)/2$ orthogonal edges encountered above and below the anchor. This yields $\Xi_\alpha^2 = \{\xi_2^2, \xi_3^2, \xi_4^2, \xi_6^2, \xi_7^2\}$. Note that we do include ξ_6^2, though this line terminates in a T-junction along the line of our search.

Let us consider a second anchor, $\mathbf{s}_\beta = \{2, 3\}$. Our process is the same as before: we search for $(p+1)/2$ orthogonal edges to the left and right of the $\mathbf{s}_\beta$. In the second parametric direction, we proceed exactly as before to obtain $\Xi_\beta^2 = \{\xi_1^2, \xi_2^2, \xi_3^2, \xi_4^2, \xi_5^2\}$. In the first parametric direction, we have the issue that the T-mesh terminates before we have found $(p+1)/2$ orthogonal edges to the left of the anchor. As stated previously, any knot values that cannot be determined in a given direction are taken to be equal to the last value added. This mimics the behavior of NURBS open knot vectors, which have their first and last knots repeated $p+1$ times. In this case, we have $\Xi_\beta^1 = \{\xi_1^1, \xi_1^1, \xi_2^1, \xi_3^1, \xi_6^1\}$.

Things proceed in a similar fashion for the case of even-degree polynomials. One difference is that anchors now fall at the center of every rectangle in the T-mesh instead of at every vertex. In addition, we no longer include the knots corresponding to the anchor's coordinates during the construction of the knot vectors. Fig. 25 shows the T-mesh corresponding to a quadratic T-spline with two anchors plotted. Note the anchors no longer have integer coordinates. We find the local knot vectors by scanning horizontally and vertically from the anchor and recording the knots associated with the first $p/2 + 1$ orthogonal edges that we encounter in each direction. For example, for $\mathbf{s}_\alpha = \{3.5, 3.5\}$ in Fig. 25, we record the knots associated with the first $p/2 + 1 = 2$ orthogonal edges to the left and right of the anchor to obtain $\Xi_\alpha^1 = \{\xi_1^1, \xi_3^1, \xi_4^1, \xi_6^1\}$. Similarly,

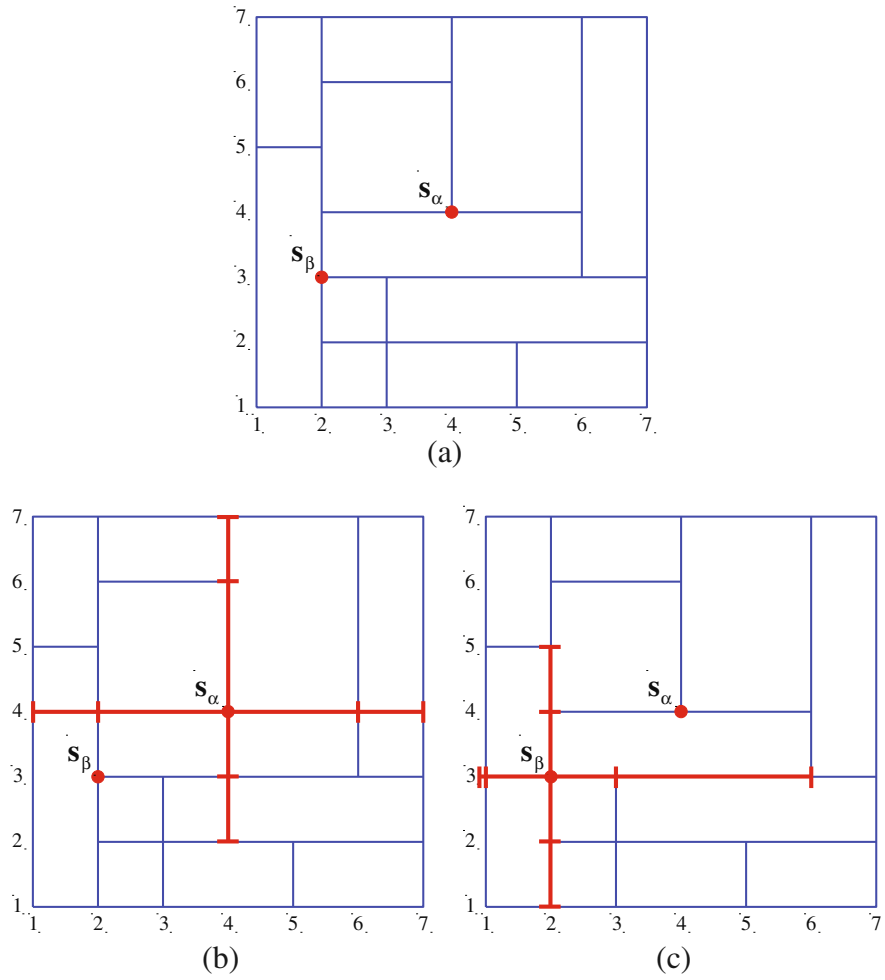


Fig. 24. For odd polynomial degrees, such as the cubic example here, an anchor is located at each vertex of the T-mesh. (a) Two anchors for a T-mesh with degree $p = 3$. The anchors $\mathbf{s}_\alpha = \{4, 4\}$ and $\mathbf{s}_\beta = \{2, 3\}$ are identified by the red circles. (b) The knot vectors for $\mathbf{s}_\alpha$ are determined by marching horizontally and vertically from the anchor until orthogonal edges are encountered in the T-mesh. These paths are indicated by the red lines and the orthogonal edges are indicated by red ticks. We have $\Xi_\alpha^1 = \{\xi_1^1, \xi_2^1, \xi_4^1, \xi_6^1, \xi_7^1\}$ and $\Xi_\alpha^2 = \{\xi_2^2, \xi_3^2, \xi_4^2, \xi_6^2, \xi_7^2\}$. (c) The knot vectors for $\mathbf{s}_\beta$ are $\Xi_\beta^1 = \{\xi_1^1, \xi_1^1, \xi_2^1, \xi_3^1, \xi_6^1\}$ and $\Xi_\beta^2 = \{\xi_1^2, \xi_2^2, \xi_3^2, \xi_4^2, \xi_5^2\}$.

in the second parametric direction we have $\Xi^2_\alpha = \{\xi^2_1, \xi^2_3, \xi^2_4, \xi^2_5\}$. Proceeding in analogous fashion for $\mathbf{s}_\beta = \{6, 5.5\}$, its knot vectors are $\Xi^1_\beta = \{\xi^1_3, \xi^1_5, \xi^1_7, \xi^1_7\}$ and $\Xi^2_\beta = \{\xi^2_1, \xi^2_4, \xi^2_7, \xi^2_7\}$.

Note that in the special case where the T-mesh occupies a rectangular subdomain in $\mathbb{R}^2$ and is a full tensor product mesh, the T-spline is equivalent to a NURBS patch.

4.2. Building a T-spline

For a given T-mesh and degree p, let $A \subset \mathbb{Z}^2$ be the index set containing every α such that $\mathbf{s}_\alpha$ is an anchor. With local knot vectors Ξ^1_α and Ξ^2_α defined for every $\alpha \in A$, using the process above, we may define functions B_α in the ***parameter space*** exactly as we did for PB-splines using (26). For each $\alpha \in A$ we choose a corresponding ***control point*** $\mathbf{P}_\alpha \in \mathbb{R}^d$, where d is the chosen dimension for the ***physical space***. For full generality, we assume that each control point has an associated weight, w_α, and construct a set of T-spline ***blending functions*** as

$$R_\alpha(\boldsymbol{\xi}) = \frac{w_\alpha B_\alpha(\boldsymbol{\xi})}{\sum_{\beta \in A} w_\beta B_\beta(\boldsymbol{\xi})}. \tag{32}$$

With such a definition, T-spline blending functions form a partition of unity. Finally, our T-spline in physical space is given by

$$S(\boldsymbol{\xi}) = \sum_{\alpha \in A} \mathbf{P}_\alpha R_\alpha(\boldsymbol{\xi}). \tag{33}$$

As in the case of NURBS, it is useful to examine the three spaces in which a T-spline may be viewed: the index, parameter, and physical spaces. In fact, though the T-mesh itself lies in the index space, it may be mapped onto the parameter and physical spaces. Such interpretations of the T-mesh may be useful in terms of inferring the structure of the individual T-spline blending functions in the parameter and physical spaces. Throughout this paper, we shall refer to the mapping of a T-mesh to the parameter or physical space as a T-mesh as well for simplicity. In Fig. 26, we have plotted the T-mesh and its images in the parameter and physical spaces for a plate with a circular hole. In Appendix A, we have detailed the control points, weights, and global knot vectors for this example. Finally, note that we may also identify locations of anchors in the parameter and physical spaces using the aforementioned mapping just as in the case of NURBS.

4.3. Continuity and definition of elements

The continuity of a T-spline in physical space follows directly from that of its blending functions in the parameter space. In the NURBS setting, this was a fairly unambiguous statement, but the local T-spline construction warrants closer consideration, particularly as it bears on numerical quadrature.

The continuity of each blending function is determined from its local knot vector, exactly as in the B-spline and NURBS cases. Each

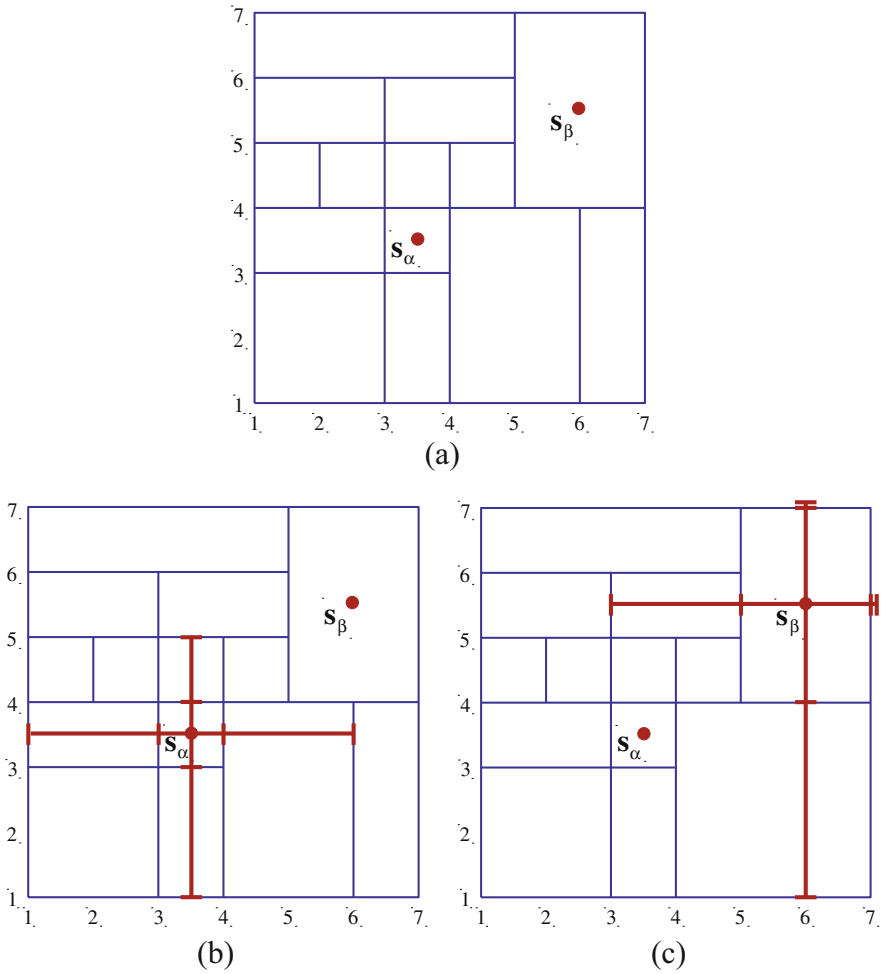


Fig. 25. For even polynomial degrees, such as the quadratic example here, an anchor is located the center of each rectangle of the T-mesh. (a) Two anchors for a T-mesh with degree $p = 2$. The anchors $\mathbf{s}_\alpha = \{3.5, 3.5\}$ and $\mathbf{s}_\beta = \{6, 5.5\}$ are identified by the red circles. (b) The knot vectors for $\mathbf{s}_\alpha$ are determined by marching horizontally and vertically from the anchor until orthogonal edges are encountered in the T-mesh. These paths are indicated by the red lines and the orthogonal edges are indicated by red ticks. We have $\Xi^1_\alpha = \{\xi^1_1, \xi^1_3, \xi^1_4, \xi^1_6\}$ and $\Xi^2_\alpha = \{\xi^2_1, \xi^2_3, \xi^2_4, \xi^2_5\}$. (c) The knot vectors for $\mathbf{s}_\beta$ are $\Xi^1_\beta = \{\xi^1_3, \xi^1_5, \xi^1_7, \xi^1_7\}$ and $\Xi^2_\beta = \{\xi^2_1, \xi^2_4, \xi^2_7, \xi^2_7\}$.

function of degree p will have C^{p-k} continuous derivatives across a knot value, where k is the multiplicity of the knot value in the local knot vector. The difficulty arises from the fact that these values only pertain to the support of the individual function. A line of decreased continuity does not necessarily propagate throughout the domain, and thus T-splines may have different degrees of smoothness within a T-mesh, as illustrated in Fig. 27.

Consider the example shown in Fig. 28 for a quadratic T-spline surface. In Fig. 28b, the continuity of the function R_α, corresponding to anchor $\mathbf{t}_\alpha$, is not immediately obvious. To determine it, we must first consider its support, $\mathbf{D}_\alpha$. From the T-mesh in Fig. 28a, we observe that the local knot vectors are $\Xi^1_\alpha = \{\xi^1_1, \xi^1_3, \xi^1_4, \xi^1_5\}$ and

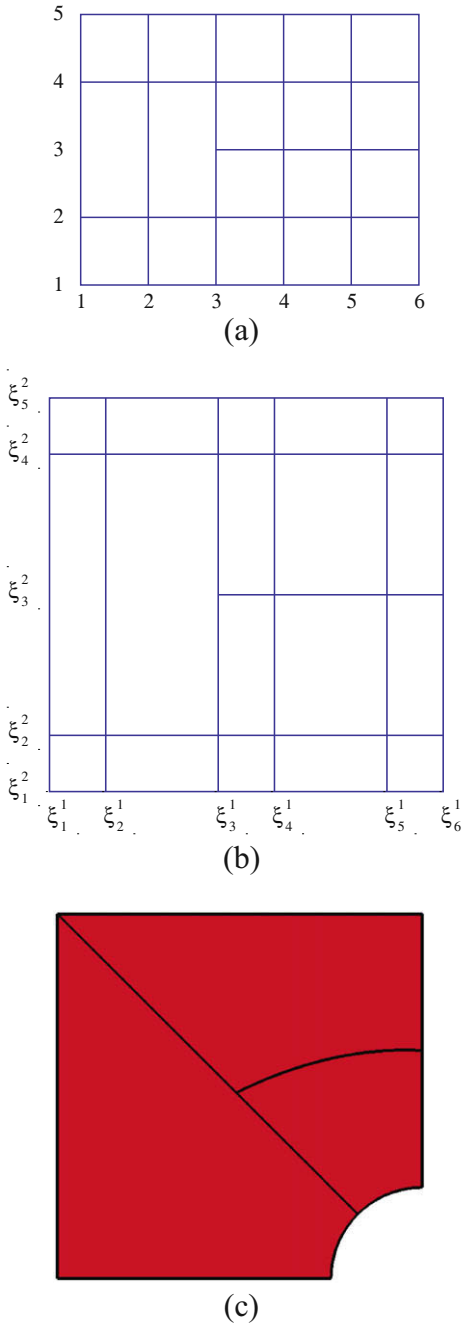


Fig. 26. T-mesh and its images in the parameter and physical spaces for a plate with a circular hole. (a) T-mesh in the index space. (b) T-mesh in the parameter space. (c) T-mesh in the physical space.

Fig. 27. T-splines allow continuity to change abruptly.

$\Xi^2_\alpha = \{\xi^2_1, \xi^2_2, \xi^2_4, \xi^2_5\}$, and thus $\mathbf{D}_\alpha = [\xi^1_1, \xi^1_5] \times [\xi^2_1, \xi^2_5]$. Function R_α is only aware of its own knots, but from its point of view they extend throughout $\mathbf{D}_\alpha$ as if this were a B-spline, as shown in Fig. 28c. The continuity of R_α is no greater[8] than C^1 across each of the knots in its knot vector. To denote the reduced continuity, we augment the original T-mesh by extending the relevant knot lines throughout $\mathbf{D}_\alpha$, using dotted lines so that the local knot vectors for other functions can still be determined unambiguously, as shown in Fig. 28d. Finally, we repeat this process for all of the functions in the T-mesh, adding ***continuity reduction lines*** where necessary, as shown in Fig. 28e. Recall that we have an anchor for a function at the center of each rectangle in index space. For the sake of nomenclature, let us distinguish between continuity reduction lines, the dotted lines we added to make continuity discernible by examining the T-mesh, and the solid lines defining the T-mesh itself. Continuity of the functions is reduced across both the continuity reduction lines and the edges of the T-mesh.

The union of all of the edges and continuity reduction lines in the T-mesh in parameter space represent the set of all of the lines across which continuity of the spline is less than C^∞. This union divides the T-mesh into rectangular regions over which the T-spline blending functions are smooth. Thus, when integrating, we may perform numerical quadrature over these regions using classical quadrature rules for C^∞ functions. This gives us a definition of ***elements*** for the purpose of numerical quadrature. In an analysis setting, it is convenient to represent T-meshes in parameter space with the continuity reduction lines included as in Fig. 28d such that elements may be defined.

Note that, in the case of odd polynomial degrees, nothing precludes us from generalizing the definition of a valid T-mesh to include ***L-junctions*** of the type seen in Fig. 29a.[9] They do not present any ambiguity or new considerations in the approach we take for inferring the local knot vectors from the T-mesh. For the experienced finite element practitioner, however, they can seem a bit strange when first encountered. It is crucial to remember that the T-mesh and the mesh are not the same thing. We define the mesh as the union of the knot lines of the T-mesh with the continuity reduction lines, as in Fig. 29b. We note that L-junctions are not permitted for even degrees.

[8] The continuity could be lower if knot values in the knot vector are repeated, just as for B-splines. In the present discussion we are only concerned with determining lines along which continuity is less than C^∞.

[9] In fact, "I-junctions" and isolated "point junctions" are legal in this framework although we will not consider these constructs in this paper.

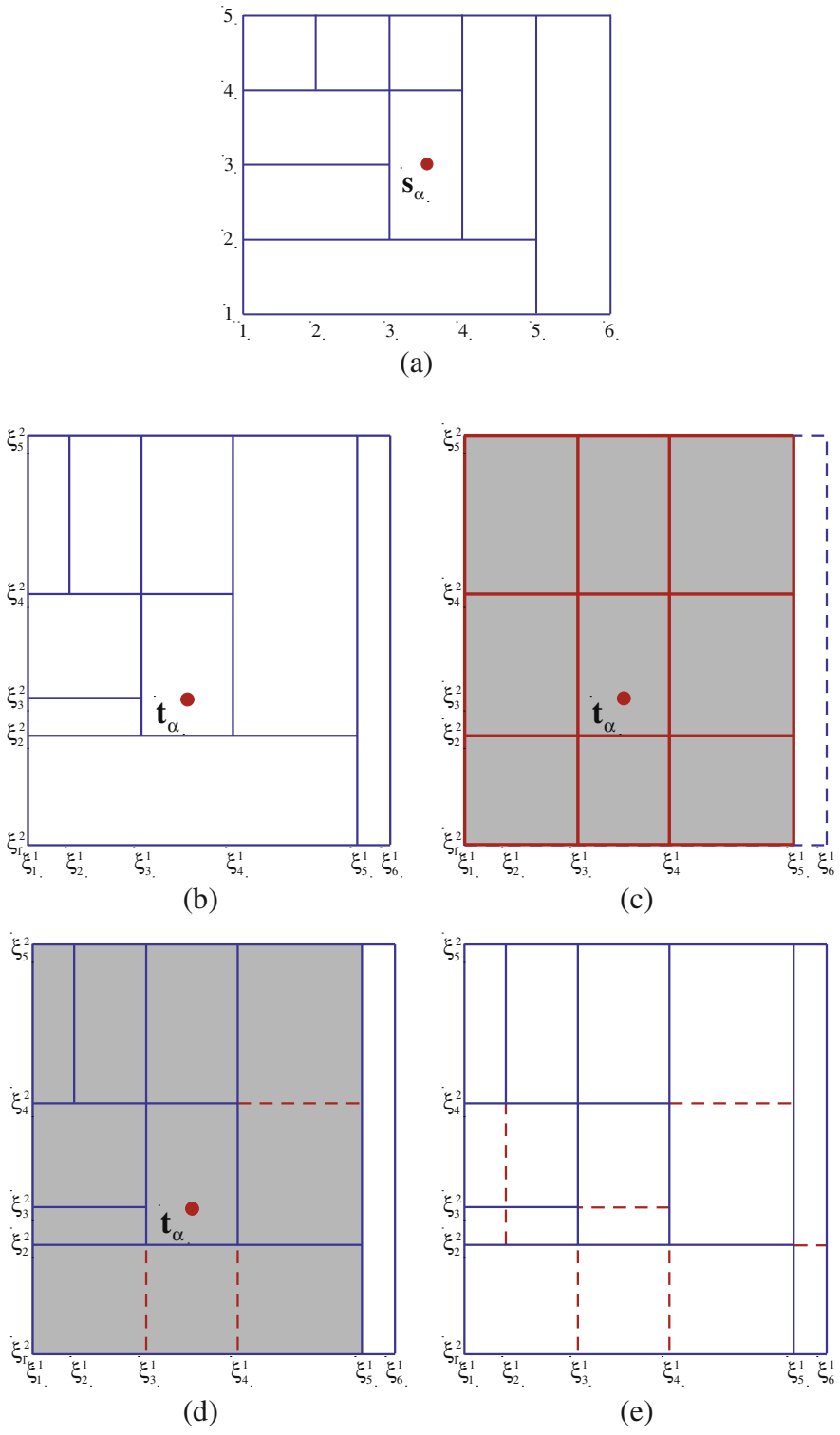


Fig. 28. Determining continuity from the T-mesh. (a) A T-mesh in index space with $p = 2$ and anchor $\mathbf{s}_\alpha$ indicated by a red circle. (b) The T-mesh in the parameter space with anchor $\mathbf{t}_\alpha$ indicated by a red circle. (c) The support $\mathbf{D}_\alpha$. The red lines indicate where the blending function has only C^1 continuity. (d) We extend the knot lines throughout $\mathbf{D}_\alpha$ to indicate that R_α is no more than C^1 continuous across them. (e) Repeating this process for all of the functions in the T-mesh enables us to determine lines of reduced continuity. Inside each rectangle, the blending functions are C^∞ and classical Gaussian quadrature may be used to evaluate element integrals. However, more efficient rules may be possible (see [33]).

4.4. T-spline volumes

The extension of T-splines to three dimensions is relatively straightforward. First we define an ***index space*** version of a ***T-mesh*** as a tiling of rectangular prisms in $\mathbb{R}^3$ where every face has a positive integer value. As an example, consider the three-dimensional T-mesh shown in Fig. 30. Next, we choose a degree p for the T-spline. Then, for each index space direction i and each integer j for which some face in the T-mesh has value j in direction i, we choose a ***knot*** $\xi_j^i \in \mathbb{R}$. We require that if $k > j$, then $\xi_k^i \geqslant \xi_j^i$. Subse-

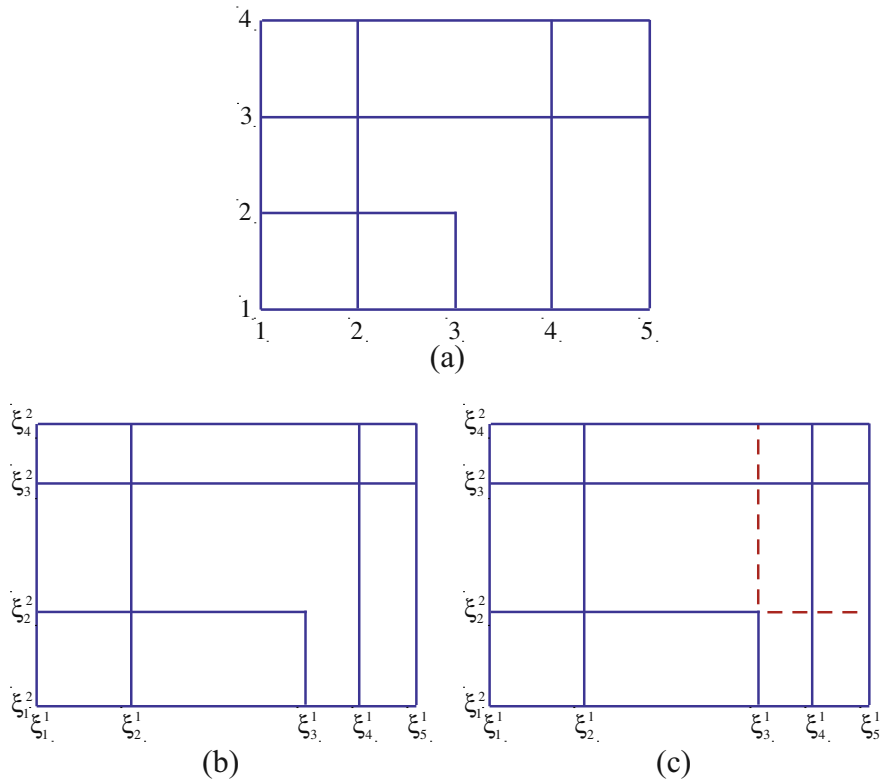


Fig. 29. Valid T-meshes for splines of odd polynomial degree can have L-junctions. The rules for inferring local knot vectors are in no way changed for such a structure. (a) Example of a T-mesh in index space with an L-junction and $p = 3$. (b) T-mesh in the parameter space. (c) Elements are defined by the union of the knot lines and the continuity reduction lines.

quent knots in the same direction may have the same value. Thus, faces in the T-mesh will correspond to knot indices.

For odd degrees, we now define an ***anchor*** $\mathbf{s}_\alpha$ at each vertex in the T-mesh. Then, for each α and each index space direction i, we create a ***local knot vector*** Ξ_α^i. As in the two-dimensional case, we place the knots corresponding to the anchor's location in the middle of their respective local knot vectors. To find the remaining knots for each index space direction i, we march in direction i from the anchor until orthogonal faces are encountered and record the knots ξ_j^i corresponding to their values in the appropriate spots in the local knot vector Ξ_α^i. This is illustrated in Fig. 31.

As before, if at any point no more faces are encountered and there still remain spots to be filled in the local knot vector, we repeat the last knot recorded in the remaining spots. This mimics the behavior of open knot vectors in the case of NURBS. The even-degree case follows in the same manner except that now we define an anchor at the center of each prism in the T-mesh. Once we determine the local knot vectors, the set of T-spline blending functions is constructed in exactly the same manner as before, utilizing (32). Thus, given a T-mesh and an appropriate set of control points, we may define a three-dimensional T-spline volume using (33). Notice that now we will have faces of reduced continuity.

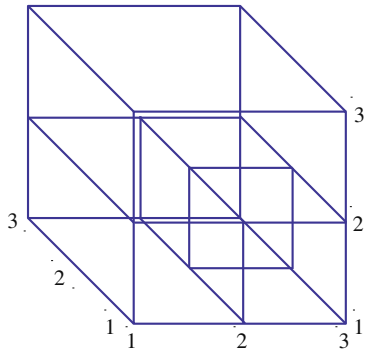


Fig. 30. Three-dimensional T-mesh.

5. Numerical results

5.1. Isogeometric fluids analysis

We begin our numerical results section by applying the T-splines based isogeometric paradigm to fluids analysis. Isogeometric analysis of fluid flows with NURBS discretizations is a well-studied topic, with applications in turbulence modeling [2,5,9], cardiovascular flows [7,13,52], and fluid–structure interaction [6–8]. It has been shown in these works that NURBS functions exhibiting higher continuity are an ideal candidate for approximating such flows.

The model problem which we consider here is the linear advection–reaction–diffusion equation:

$$cu + \mathbf{a} \cdot \nabla u - \nabla \cdot (\kappa \nabla u) = f, \tag{34}$$

where u denotes the concentration, c is the reaction coefficient, $\mathbf{a}$ is the velocity, κ is the diffusivity, and f is the source.

We consider two opposite regimes of the advection–reaction–diffusion system:

- reaction–diffusion ($\mathbf{a} = 0$), and
- advection–diffusion ($c = 0$).

The first example involves the simulation of the reaction–diffusion problem described in Fig. 32. We note that the boundary condition is zero except near the corners. Since this particular problem

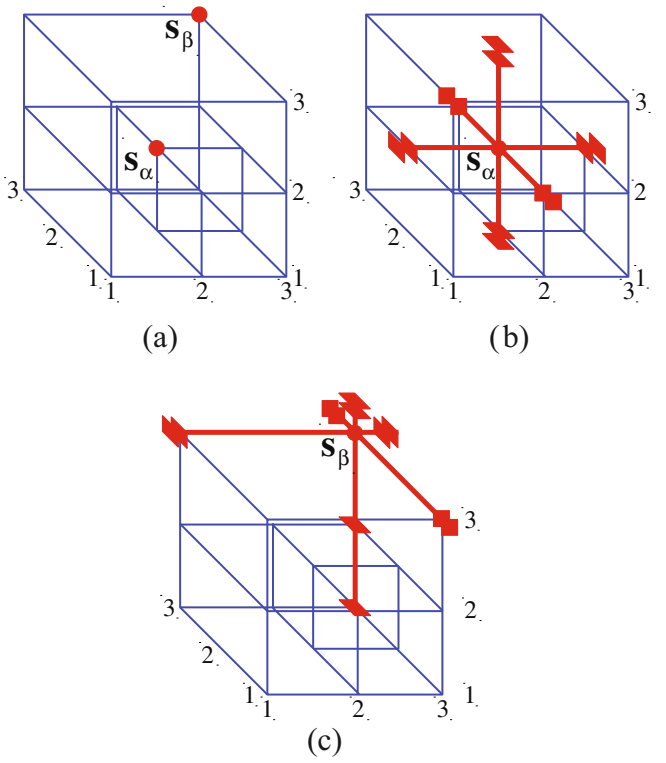


Fig. 31. Extracting local knot vectors from a three-dimensional T-mesh. (a) T-mesh in index space with $p = 3$ and anchors $\mathbf{s}_\alpha = \{2,2,2\}$ and $\mathbf{s}_\beta = \{3,3,3\}$ indicated by red circles. (b) The knot vectors for $\mathbf{s}_\alpha$ are $\Xi^1_\alpha = \{\xi^1_1, \xi^1_1, \xi^1_2, \xi^1_3, \xi^1_3\}$, $\Xi^2_\alpha = \{\xi^2_1, \xi^2_1, \xi^2_2, \xi^2_3, \xi^2_3\}$, and $\Xi^3_\alpha = \{\xi^3_1, \xi^3_1, \xi^3_2, \xi^3_3, \xi^3_3\}$. (c) The knot vectors for $\mathbf{s}_\beta$ are $\Xi^1_\beta = \{\xi^1_1, \xi^1_1, \xi^1_3, \xi^1_3, \xi^1_3\}$, $\Xi^2_\beta = \{\xi^2_1, \xi^2_1, \xi^2_3, \xi^2_3, \xi^2_3\}$, and $\Xi^3_\beta = \{\xi^3_1, \xi^3_2, \xi^3_3, \xi^3_3, \xi^3_3\}$.

is strongly dominated by reaction (the Damköhler number is 10^3), we expect that the solution is zero except near the corners, where it rapidly spikes to one. With local refinement, we hope to resolve these spikes near the corners.

The T-meshes we utilized for solving this problem are presented in Fig. 33, and numerical results were obtained using Galerkin's method. Solution profiles for polynomial degrees $p = 1, 2, 3$ are presented in Figs. 34–36. With increasing polynomial degree and local refinement, we see that not only are we able to more sharply capture the corner phenomena but we are also able to eliminate spurious oscillations.

The second example involves solution of the advection–diffusion problem shown in Fig. 37. Here, the Péclet number, which characterizes competition between advection and diffusion, is 10^{-6}, making the problem strongly advection-dominated, and thus we expect a sharp interior layer and sharp boundary layers at the outflow. Successful capturing of such layers requires robust, stable numerical techniques in addition to increased resolution. The problem was investigated using NURBS-based isogeometric analysis, SUPG stabilization [11], and k-refinement in [30]. There, it was noted that globally k-refined meshes produced results that were nearly monotone. Here, we investigate the effect of local h-refinement on the solution.

We begin with a uniform T-mesh of 10×10 elements, which corresponds to a NURBS discretization. We then employ an automatic refinement scheme that makes use of a simple gradient-based error indicator and the algorithm described in [42]. At each step of the refinement, elements are identified for refinement based on the magnitude of the indicator. A list of elements is then defined, and they are bisected in each parametric direction sequentially according to the order they appear in the list. In all cases, the standard SUPG formulation is used with stabilization parameter

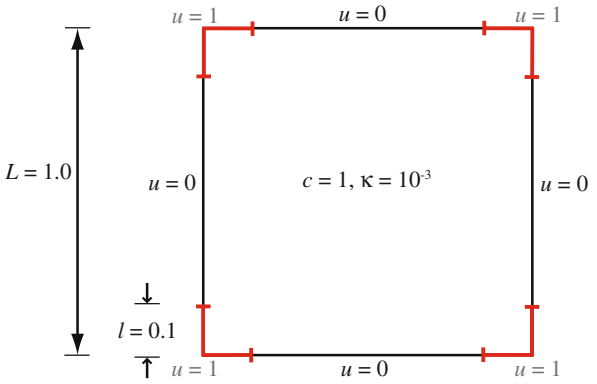


Fig. 32. Reaction–diffusion problem. Problem description and data.

$\tau = h_a/2a$, where h_a is the integration element length in the direction of the flow velocity. For the problem considered, $a = |\mathbf{a}| = 1$ and $h_a = \sqrt{2}h$, where h is the element edge length. Recall that integration elements are defined by adding continuity reduction lines as described in Section 4.3.

The T-meshes generated by automatic refinement for the linear ($p = 1$) case are shown in Fig. 38. We note that T-splines in this case correspond to standard bilinear transition elements (see [29], Figs. 3.11.3 and 3.11.4, and the accompanying exercise). We see that the resulting T-meshes are refined near the interior and boundary layers and the effects of refinement are quite localized. We further see that L-junctions naturally arise from the automatic refinement scheme. The results for these T-meshes are presented in Fig. 39. We find that with refinement, the layers become sharper, and the overshoots and undershoots about the layers are attenuated but not eliminated. This is in contrast with the situation described in Hughes et al. [30] in which k-refined meshes converged toward monotone solutions. It is a somewhat surprising fact that smooth, higher-degree functions produce better results than locally-refined low-degree functions. This runs counter to the conventional wisdom.

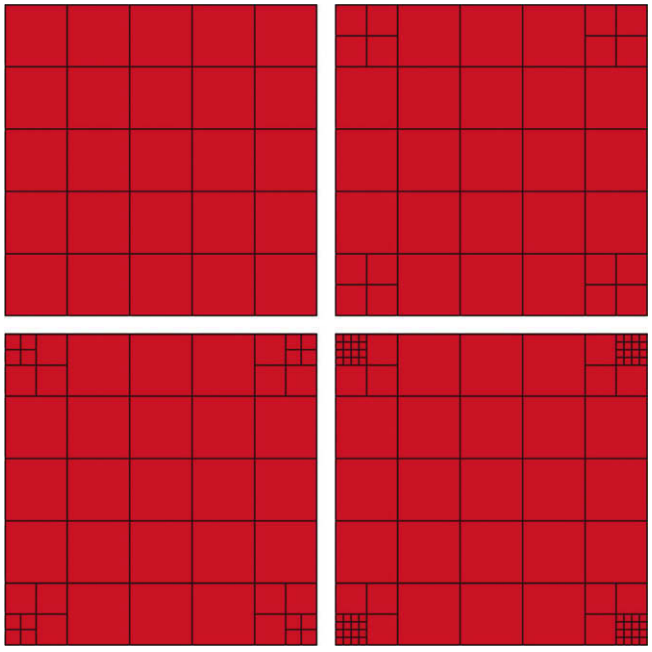


Fig. 33. Reaction–diffusion problem. Sequence of T-meshes in physical space.

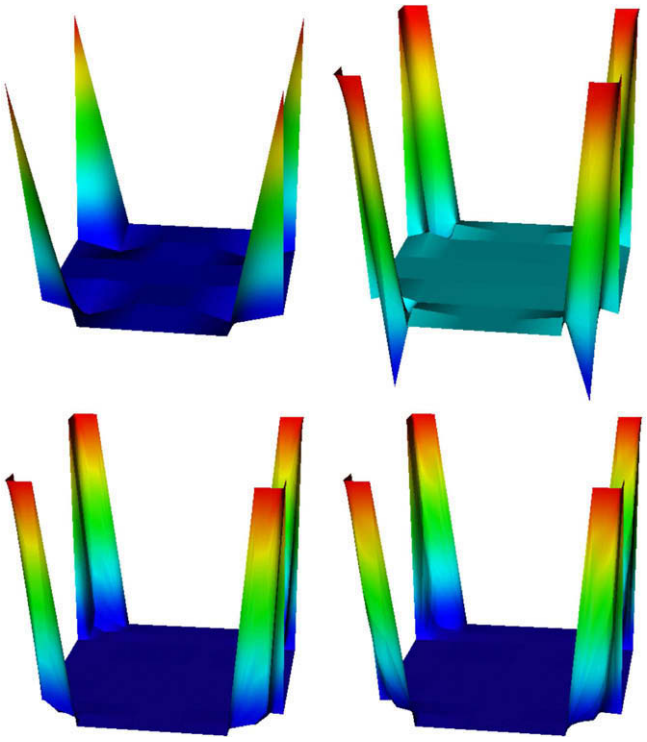


Fig. 34. Reaction–diffusion problem. Results for $p = 1$.

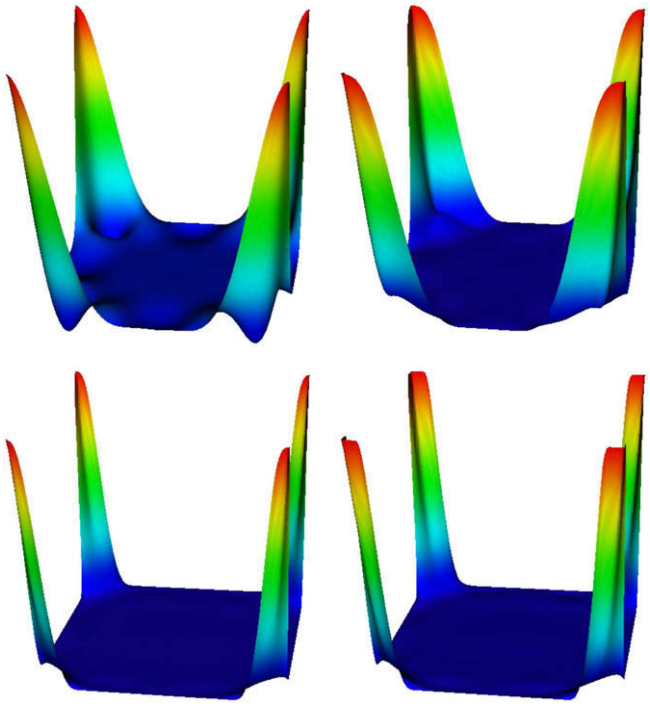


Fig. 36. Reaction–diffusion problem. Results for $p = 3$.

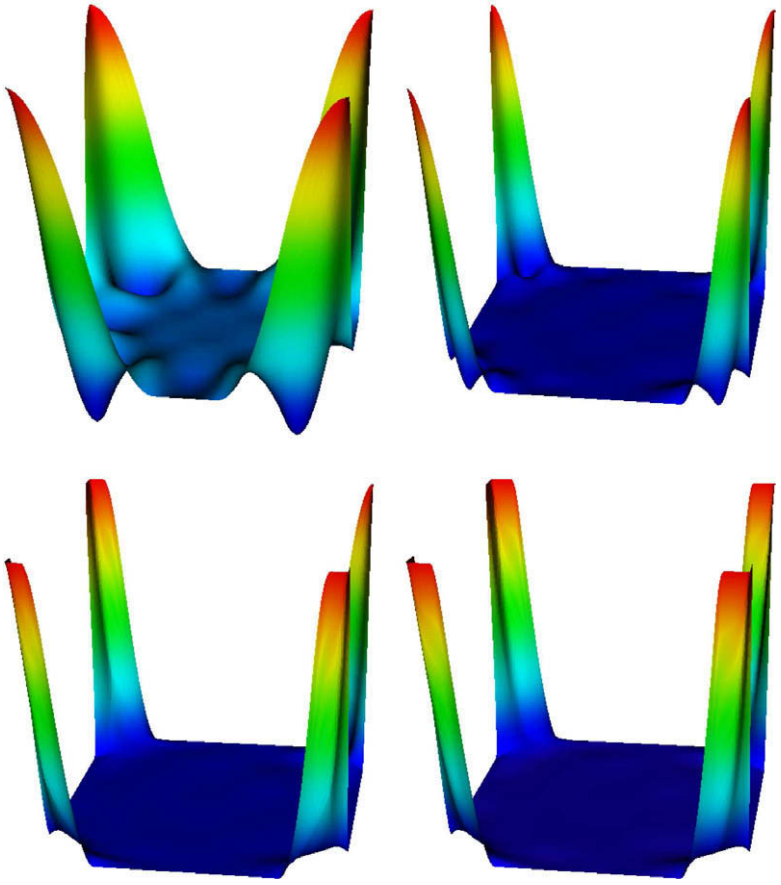


Fig. 35. Reaction–diffusion problem. Results for $p = 2$.

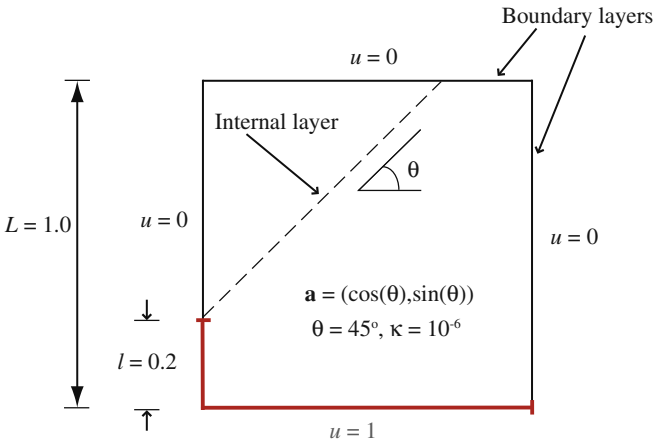


Fig. 37. Advection–diffusion problem, $\theta = 45^{\circ}$. Problem description and data.

In passing we note that the refinement algorithm described in [42] depends on the order of the elements identified for refinement and, in certain cases, the refinement can propagate throughout a T-mesh. For example, in a recent work by Dorfel et al. [20], automatic T-spline local refinement employing bicubic T-splines resulted in globally refined T-meshes when applied to the same advection–diffusion problem as above. We have experienced similar behavior, and this indicates to us the importance of developing new refinement algorithms with better invariance and localization properties (see also [20]).

5.2. Isogeometric structural analysis

An isogeometric structural analysis framework based on T-splines preserves all of the desirable properties of its NURBS counterpart. In particular, T-splines also satisfy standard patch tests. In what follows, we present numerical solutions for linear elastic solids and structures. The Galerkin formulation of linear elasticity is employed. All the calculations involve thin shells, but these are modeled as three-dimensional solids and no shell assumptions are employed. A direct algebraic equation solver was employed for each of the calculations.

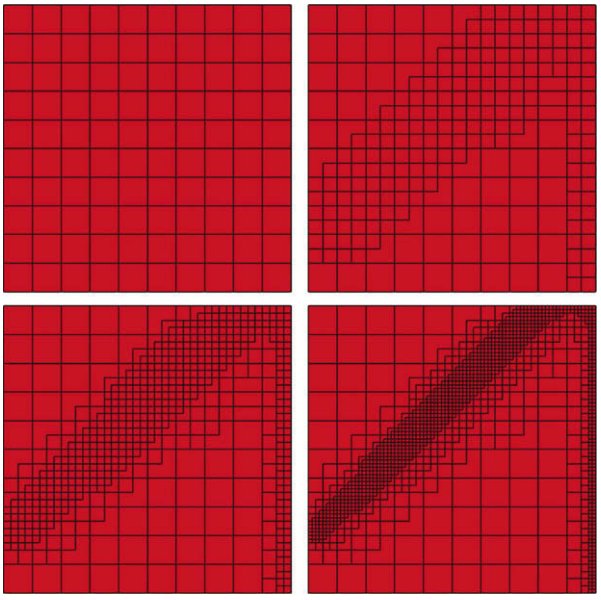


Fig. 38. Advection skew to the mesh, $\theta = 45^{\circ}$. Sequence of T-meshes in physical space for $p = 1$.

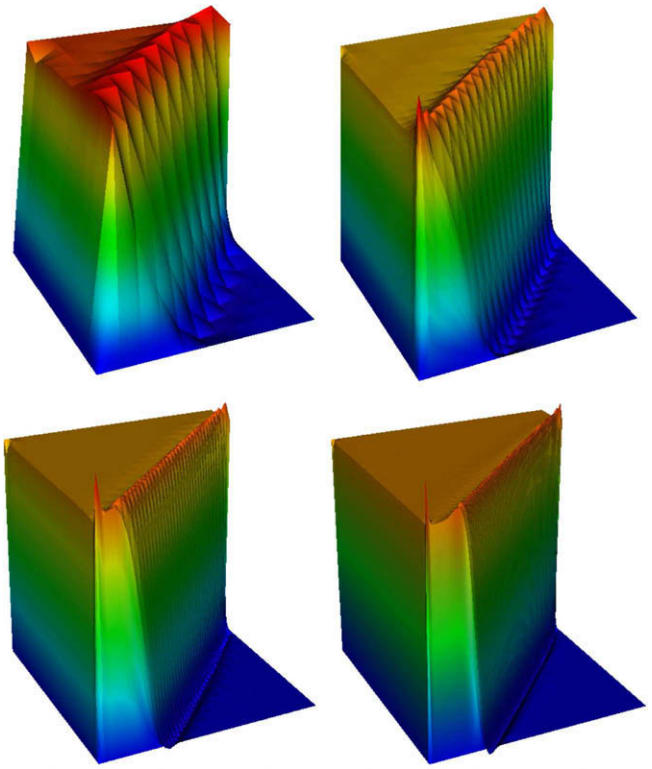


Fig. 39. Advection skew to the mesh, $\theta = 45^{\circ}$. Results for $p = 1$.

The first two examples come from the so-called shell obstacle course: the pinched hemisphere and the pinched cylinder. These problems, and their relevance to the assessment of shell analysis procedures, have been discussed extensively in the literature, and the particular form of the problems we have chosen is adapted from Felippa [25] and Belytschko et al. [10]. These problems were chosen due to the local nature of their external forcing (in fact, in both examples point loads are applied). The last example we consider is the hemispherical shell with stiffener presented in Rank et al. [39] who solved the problem using a finite element method and p-refinement strategy. In each of these examples, rather than using an automatic refinement strategy, we hand-crafted a sequence of T-meshes for various polynomial degrees.

5.2.1. Pinched hemisphere

In the pinched hemisphere, equal and opposite concentrated point load forces are applied at antipodal points of the equator.

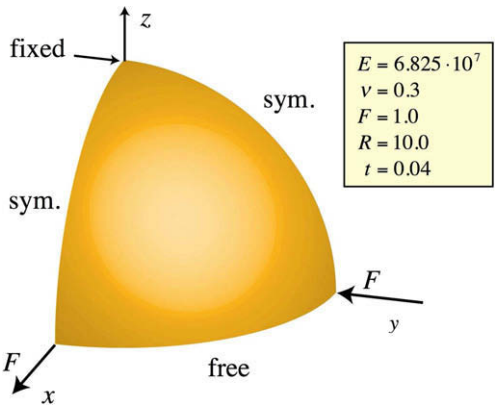


Fig. 40. Shell obstacle course. Pinched hemisphere problem description and data.

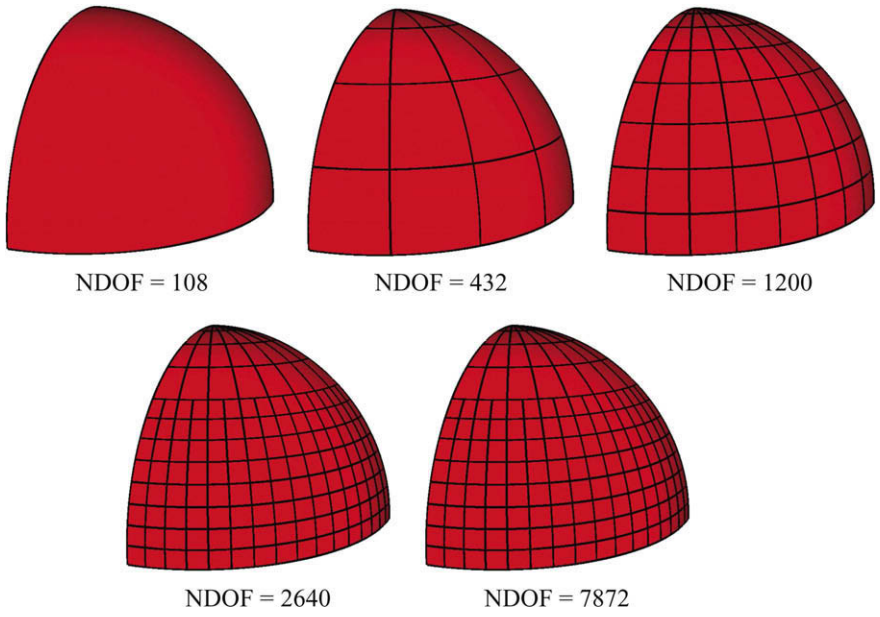


Fig. 41. Shell obstacle course. Sequence of T-meshes in physical space for pinched hemisphere for $p = 2$.

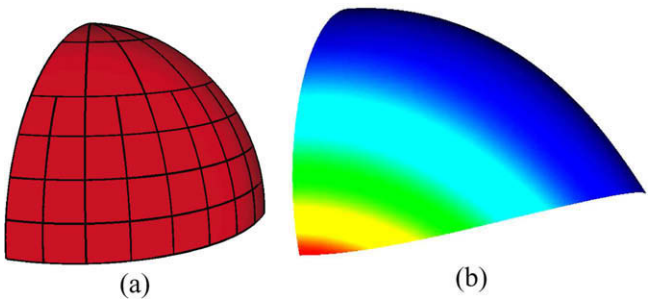


Fig. 42. Shell obstacle course. Pinched hemisphere with $p = 5$, NDOF = 1524: (a) T-mesh in physical space and (b) displacement contours in direction of inward directed point load on deformed configuration (scaling factor of 30 used).

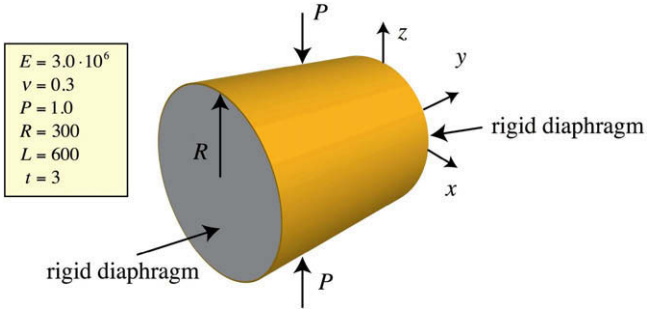


Fig. 44. Shell obstacle course. Pinched cylinder problem description and data.

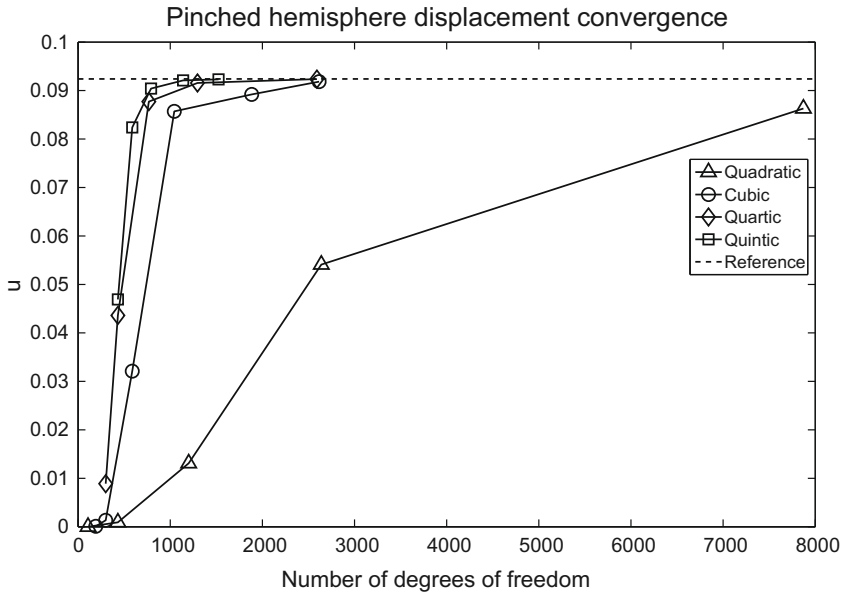


Fig. 43. Shell obstacle course. Pinched hemisphere displacement convergence.

The equator is otherwise considered to be free (see Fig. 40). An example sequence of T-meshes is shown in Fig. 41. Due to symmetry, only one quadrant is meshed. In this case, two quadratic NURBS elements were employed in the through-thickness direction (see [30]). Quadratic through quintic surface T-splines were employed. A contour plot of the displacement on the deformed configuration is shown in Fig. 42. Convergence of displacement under the inward directed load for the various degrees is presented in

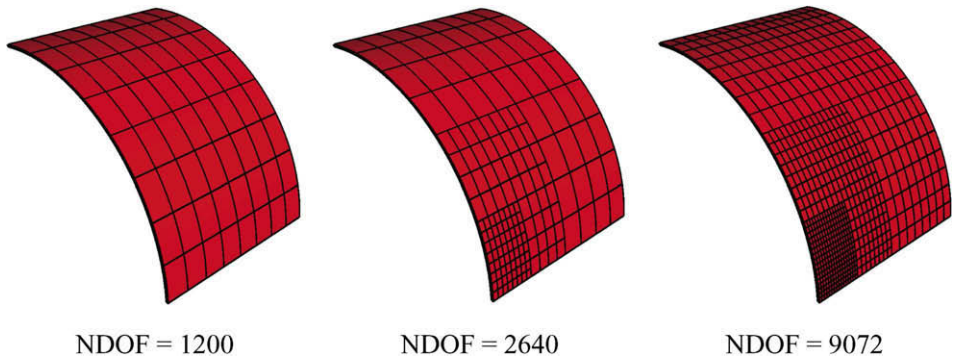


Fig. 45. Shell obstacle course. Sequence of T-meshes in physical space for pinched cylinder for $p = 2$.

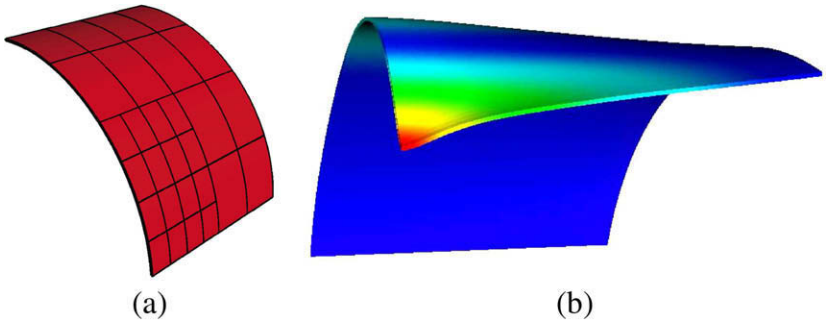


Fig. 46. Shell obstacle course. Pinched cylinder with $p = 5$, NDOF = 1260: (a) T-mesh in physical space and (b) displacement contours on deformed configuration (scaling factor of 3×10^6 used).

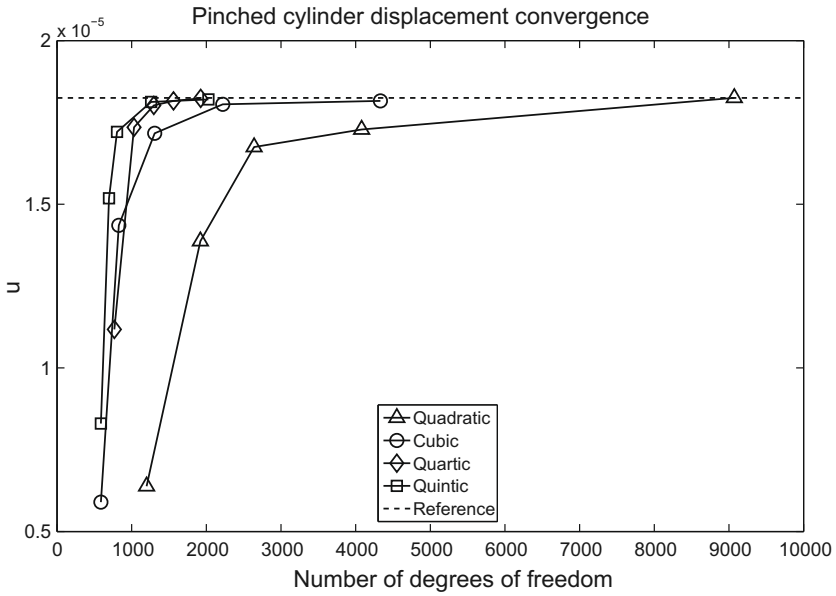
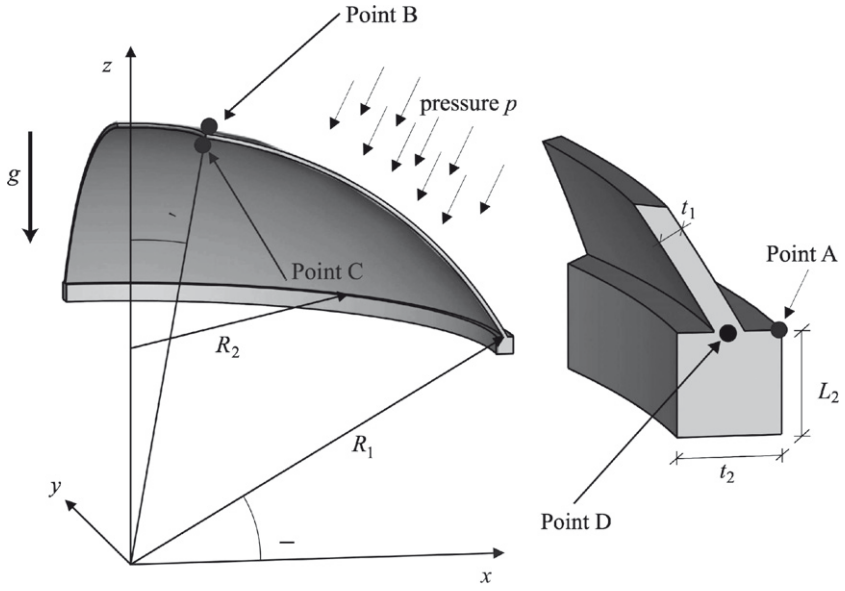


Fig. 47. Shell obstacle course. Pinched cylinder displacement convergence.

Fig. 43. As in the case of NURBS [30], the quadratic case converges very slowly and higher-degree functions lead to much faster convergence to the exact solution.

5.2.2. Pinched cylinder

The pinched cylinder is subjected to equal and opposite concentrated forces at its midspan (see Fig. 44). The ends are supported by



Boundary conditions:

At bottom surface of stiffener:
$u_z = 0$

Symmetry at x-z-plane:
$u_y = 0$

Symmetry at y-z-plane:
$u_x = 0$

$$
\begin{aligned}
\alpha &= 30^\circ \\
\beta &= 10^\circ \\
R_1 &= 10m \\
R_2 &= 5\sqrt{3}m \\
t_1 &= 0.1m \\
t_2 &= 0.4m \\
L_2 &= 0.4m \\
E &= 6.825 \cdot 10^7 \tfrac{kN}{m^2} \\
\nu &= 0.3 \\
\rho &= 500 \tfrac{kg}{m^3} \\
g &= 10.0 \tfrac{m}{s^2} \\
p &= 100.0 \tfrac{kN}{m^2}
\end{aligned}
$$

Fig. 48. Hemispherical shell with stiffener. Problem description from Rank et al. [39].

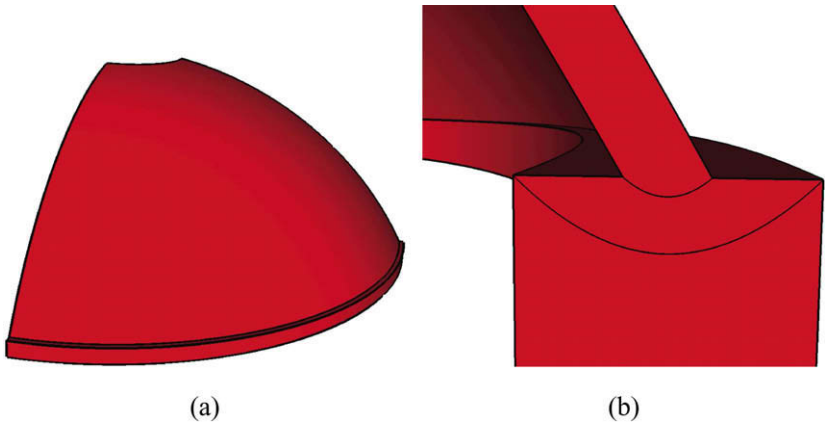


Fig. 49. Hemispherical shell with stiffener. Initial T-mesh in physical space, NDOF = 360: (a) Coarse T-mesh in physical space and (b) detail of stiffener.

rigid diaphragms. This constraint results in highly localized deformation under the loads. Only one octant of the cylinder is used in the calculations due to symmetry. As in the case of the pinched hemisphere, two quadratic NURBS elements were utilized in the through-thickness direction and quadratic through quintic degrees of T-splines were utilized on the surface. An example sequence of

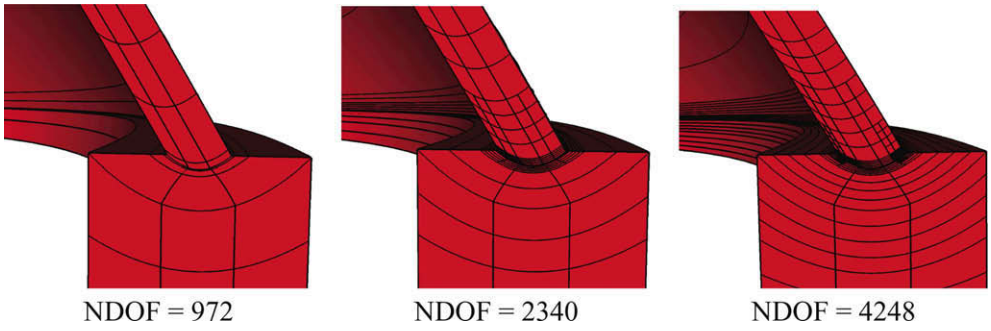


Fig. 50. Hemispherical shell with stiffener. Refined T-meshes in physical space.

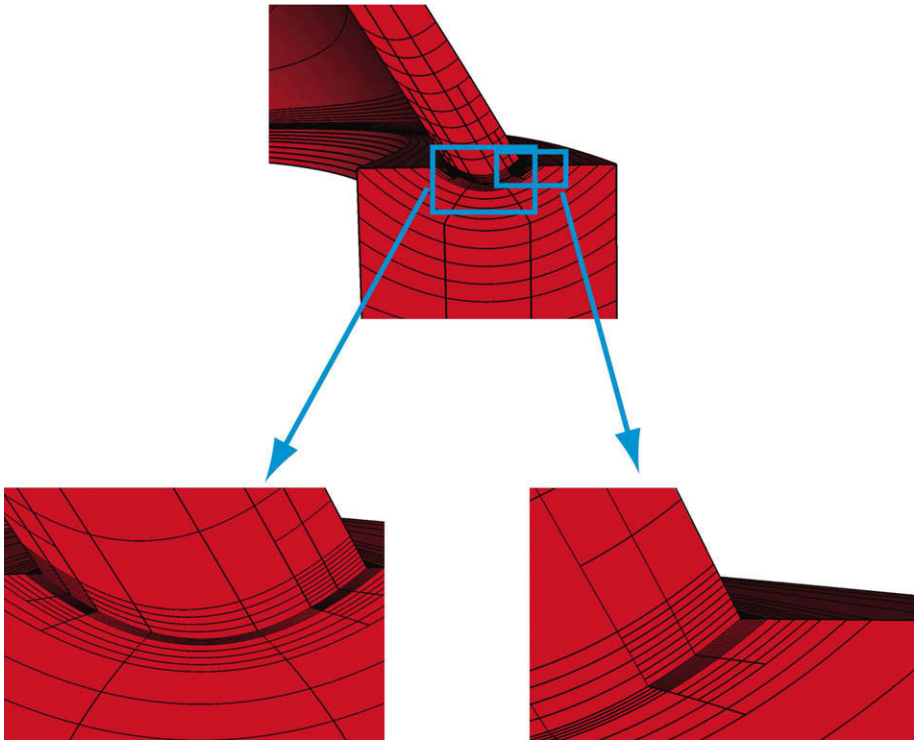


Fig. 51. Hemispherical shell with stiffener. Detail of refinement for finest T-mesh in physical space (NDOF = 4248).

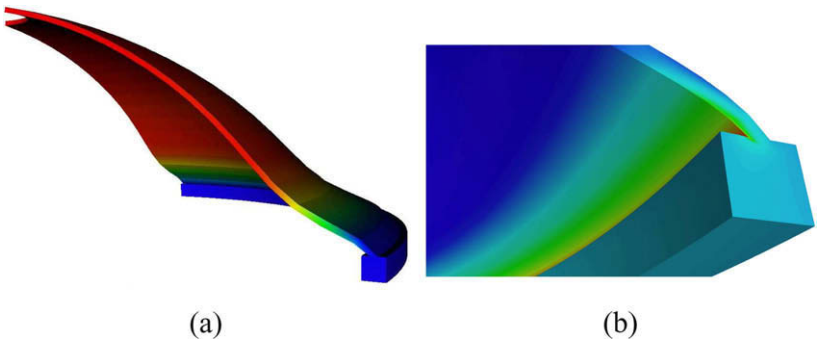


Fig. 52. Finest T-mesh in physical space (NDOF = 4248): (a) Vertical displacement contours (scaling factor of 500 used) and (b) von Mises stress contours, detail of stiffener.

T-meshes is shown in Fig. 45. A contour plot of the displacement on the deformed configuration is shown in Fig. 46. Convergence of the displacement under the load is plotted in Fig. 47. It is well known that, as long as the characteristic surface element dimension is large compared with the thickness, formulations which permit transverse shear deformations typically closely approximate formulations which satisfy the Kirchhoff constraint (i.e., zero transverse shear) [31].

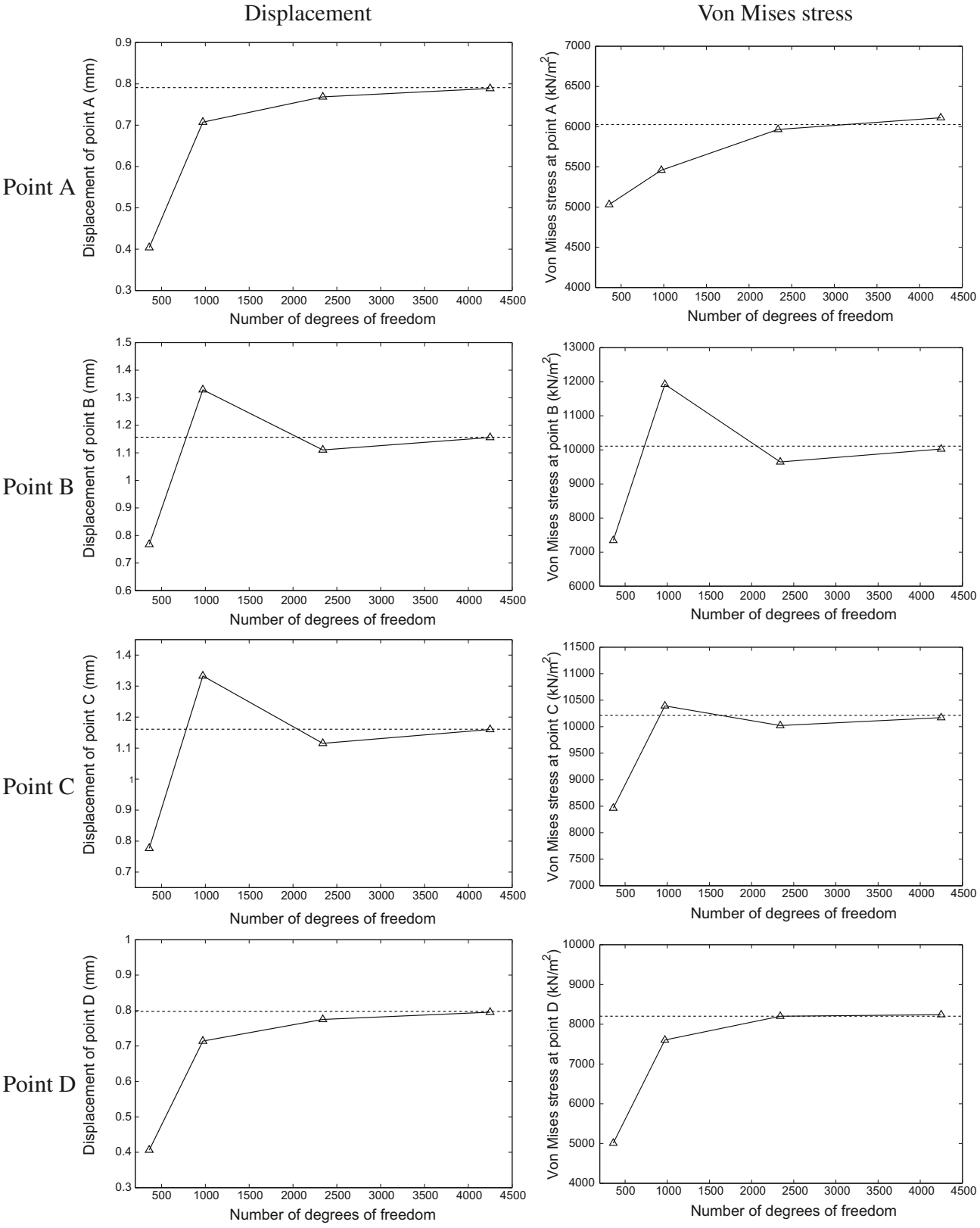


Fig. 53. Hemispherical shell with stiffener. Convergence of displacement and von Mises stress at various points to benchmark solution [39].

5.2.3. Hemispherical shell with a stiffener

The hemispherical shell with stiffener is subjected to gravity loading and external pressure, with the bottom surface fixed in the vertical direction (see Fig. 48). Only a quarter of the domain is modeled due to symmetry. The initial T-mesh is constructed using rational quadratic NURBS in the circumferential direction and cubic NURBS in the other two directions and is shown in Fig. 49. A series of refined T-meshes is shown in Fig. 50. We found that no refinement was needed in the circumferential direction due to the axisymmetry of the solution and the fact that an exact geometry is employed (see Fig. 51). Fig. 52 shows the vertical displacement and von Mises stress on the deformed configuration.

The Euclidean norm of the displacement and the von Mises stress were calculated at points A–D, identified in the problem description (Fig. 48). Results for sequences of T-meshes are plotted in Fig. 53 along with results from Rank et al. [39] for their finest simulation ($p = 8$) which we take as a reference. Good agreement in the converged displacements and von Mises stresses is observed for cases except for the von Mises stress at point A, in which case the present result of the finest T-mesh employed is somewhat higher than the result of Rank et al. [39].

6. Conclusions

T-splines represents an extension of NURBS technology that permits local refinement, watertight merging of patches, and a solution to the trimmed surface problem. These features are highly desirable in a design context and T-splines have recently become available in two NURBS-based design systems, namely, Maya [45] and Rhino [46]. In this paper we have explored T-splines as a basis for isogeometric analysis. The same features that make T-splines attractive for design, make it attractive for analysis. We have applied bivariate and trivariate T-splines of various degrees to elementary problems of fluid and structural mechanics and in all cases they performed well. As far as surface modeling goes, we believe T-splines technology provides a nearly complete basis for analysis.

Existing T-spline surface modeling tools need to output all paraphernalia necessary to perform analysis. Presumably, this is not a major task and may be available soon. Outstanding issues concern efficient refinement strategies and treatment of so-called extraordinary points. These issues need to be researched from an analysis perspective. The main issue facing three-dimensional analysis is model generation. There is probably no unique solution to this problem judging from the variety of mesh generation procedures currently used in finite element analysis. Different types of procedures may suit different classes of analysis problems. An ambitious challenge would be to generate a trivariate T-spline representation preserving a given T-spline surface representation. A less ambitious, but nevertheless still very challenging task would be to generate a trivariate T-spline that approximates a given T-spline surface to a specified accuracy. Solutions of these problems would represent important advances in eliminating the engineering design-to-analysis bottleneck. For a particularly promising technology based on the concept of polycube splines, see [35,50]. T-splines also need to be thoroughly researched and compared with traditional finite element methodology on a wider variety of analysis applications.

Acknowledgements

We wish to thank Omar Ghattas for helpful insights into the relationship between isogeometric analysis and high-performance computing. J.A. Evans was partially supported by the Department of Energy Computational Science Graduate Fellowship, provided under Grant Number DE-FG02-97ER25308. S. Lipton was partially supported by the Department of Defense National Defense Science and Engineering Fellowship. Support of the Office of Naval Research Contract N00014-03-0263, Dr. Luise Couchman, contract monitor, is gratefully acknowledged.

Appendix A. T-spline data for plate with a hole

In this appendix, we explicitly tabulate the T-spline paraphernalia for a simple example. Namely, we fully describe all of the attributes of the quadratic T-mesh of a rectangular plate with a circular hole illustrated in Fig. A.1. The mesh is built from the T-mesh described in Fig. A.2. The knot values for the two parametric directions as well as the locations of anchors in parametric space are described in Fig. A.3. Table A.1 lists the local knot vectors associated with each of the basis functions for the T-mesh while Table A.2 lists the control points and weights to build the plate with a hole.

Appendix B. Patch merging with T-splines

In this appendix we explain the concept of patch merging using two simple examples. In these two examples, we merge two quadratic NURBS patches such that the final geometry is C^0- and C^1-continuous, respectively. The T-spline local refinement algorithm

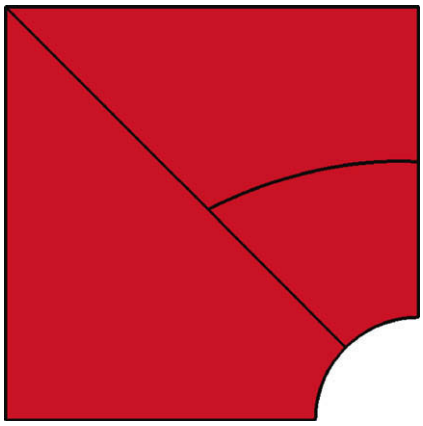


Fig. A.1. Plate with a hole: T-mesh in physical space.

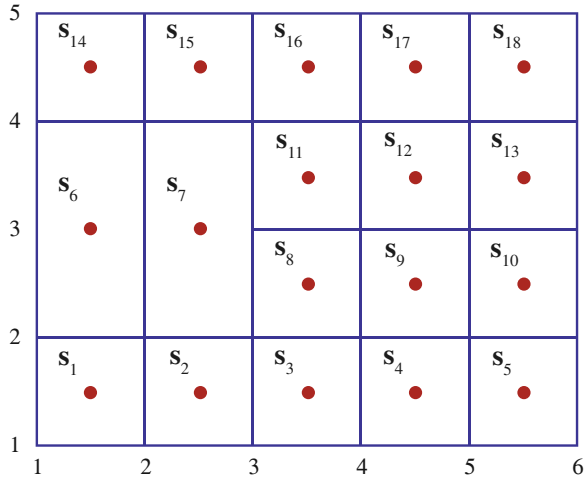


Fig. A.2. Plate with a hole: T-mesh in index space. Anchors are designated by red circles.

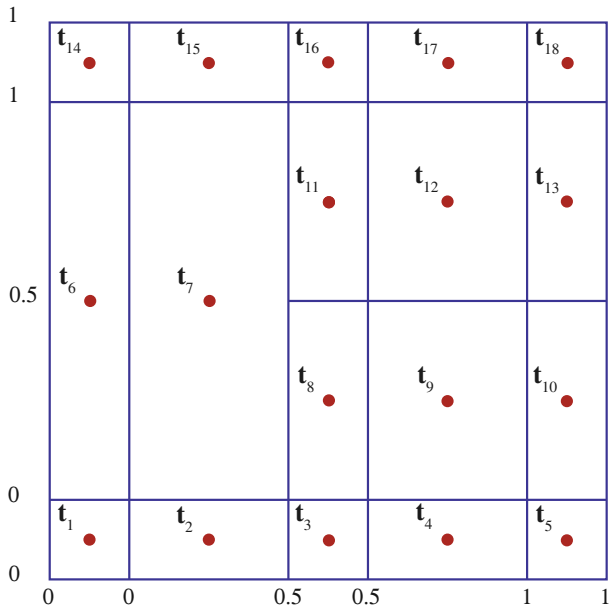


Fig. A.3. Plate with a hole: T-mesh in parameter space. Repeated knot lines are expanded to emphasize the connection between parameter and index space. Anchors are designated by red circles.

used in this merging as well as more details on patch merging algorithms in the context of T-splines may be found in [44].

Suppose we wish to merge two quadratic NURBS patches whose T-meshes are shown in Fig. B.1a and b. For a C^0 merge, we first locally refine the last column of the T-mesh in Fig. B.1a and the first column of the T-mesh in Fig. B.1b as demonstrated in Fig. B.2. Following this refinement step, we adjust, if necessary, the control points associated with the last column of the T-mesh in Fig. B.2a and the first column of the T-mesh in Fig. B.2b such that they are equal. This ensures that the boundaries of the two geometries match and completes the C^0 merging of the two quadratic patches.

The final geometry can be represented by the single T-mesh given in Fig. B.3 with appropriately chosen control points. These control points are inherited from the original two patches and this

Table A.1
Local knot vectors for T-spline basis functions.

Anchor $\mathbf{t}_\alpha$	Horizontal local knot vector Ξ^1_α	Vertical local knot vector Ξ^2_α
$\mathbf{t}_1$	$\{0,0,0,0.5\}$	$\{0,0,0,1\}$
$\mathbf{t}_2$	$\{0,0,0.5,0.5\}$	$\{0,0,0,1\}$
$\mathbf{t}_3$	$\{0,0.5,0.5,1\}$	$\{0,0,0,0.5\}$
$\mathbf{t}_4$	$\{0.5,0.5,1,1\}$	$\{0,0,0,0.5\}$
$\mathbf{t}_5$	$\{0.5,1,1,1\}$	$\{0,0,0,0.5\}$
$\mathbf{t}_6$	$\{0,0,0,0.5\}$	$\{0,0,1,1\}$
$\mathbf{t}_7$	$\{0,0,0.5,0.5\}$	$\{0,0,1,1\}$
$\mathbf{t}_8$	$\{0,0.5,0.5,1\}$	$\{0,0,0.5,1\}$
$\mathbf{t}_9$	$\{0.5,0.5,1,1\}$	$\{0,0,0.5,1\}$
$\mathbf{t}_{10}$	$\{0.5,1,1,1\}$	$\{0,0,0.5,1\}$
$\mathbf{t}_{11}$	$\{0,0.5,0.5,1\}$	$\{0,0.5,1,1\}$
$\mathbf{t}_{12}$	$\{0.5,0.5,1,1\}$	$\{0,0.5,1,1\}$
$\mathbf{t}_{13}$	$\{0.5,1,1,1\}$	$\{0,0.5,1,1\}$
$\mathbf{t}_{14}$	$\{0,0,0,0.5\}$	$\{0,1,1,1\}$
$\mathbf{t}_{15}$	$\{0,0,0.5,0.5\}$	$\{0,1,1,1\}$
$\mathbf{t}_{16}$	$\{0,0.5,0.5,1\}$	$\{0.5,1,1,1\}$
$\mathbf{t}_{17}$	$\{0.5,0.5,1,1\}$	$\{0.5,1,1,1\}$
$\mathbf{t}_{18}$	$\{0.5,1,1,1\}$	$\{0.5,1,1,1\}$

Table A.2
Control points and weights for plate with a hole.

Anchor $\mathbf{t}_\alpha$	Control point $\mathbf{P}_\alpha$	Weight w_α
$\mathbf{t}_1$	$\{-1,0\}$	1
$\mathbf{t}_2$	$\{-1,0.41421\}$	0.85355
$\mathbf{t}_3$	$\{-0.70711,0.70711\}$	0.85355
$\mathbf{t}_4$	$\{-0.41421,1\}$	0.85355
$\mathbf{t}_5$	$\{-1.375,3.25\}$	1
$\mathbf{t}_6$	$\{-2.5,0\}$	1
$\mathbf{t}_7$	$\{-2.5,0.75\}$	1
$\mathbf{t}_8$	$\{-1.20231,1.20231\}$	0.92678
$\mathbf{t}_9$	$\{-0.59537,1.80926\}$	0.92678
$\mathbf{t}_{10}$	$\{0,1.75\}$	1
$\mathbf{t}_{11}$	$\{-2.8125,2.8125\}$	1
$\mathbf{t}_{12}$	$\{-1.375,3.25\}$	1
$\mathbf{t}_{13}$	$\{0,3.25\}$	1
$\mathbf{t}_{14}$	$\{-4,0\}$	1
$\mathbf{t}_{15}$	$\{-4,2\}$	1
$\mathbf{t}_{16}$	$\{-4,4\}$	1
$\mathbf{t}_{17}$	$\{-2,4\}$	1
$\mathbf{t}_{18}$	$\{0,4\}$	1

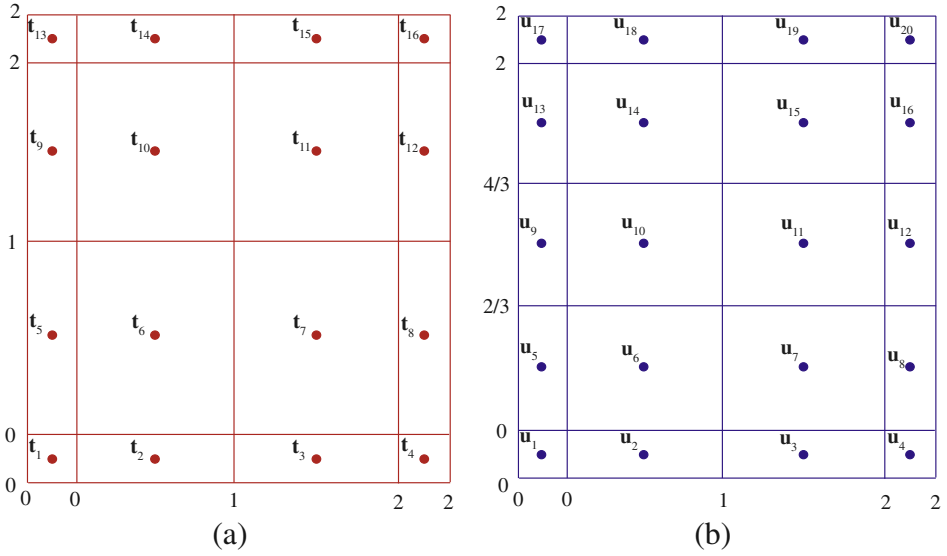


Fig. B.1. T-meshes in parameter space and knot values of two quadratic NURBS patches. In this figure, anchors are designated by red and blue circles.

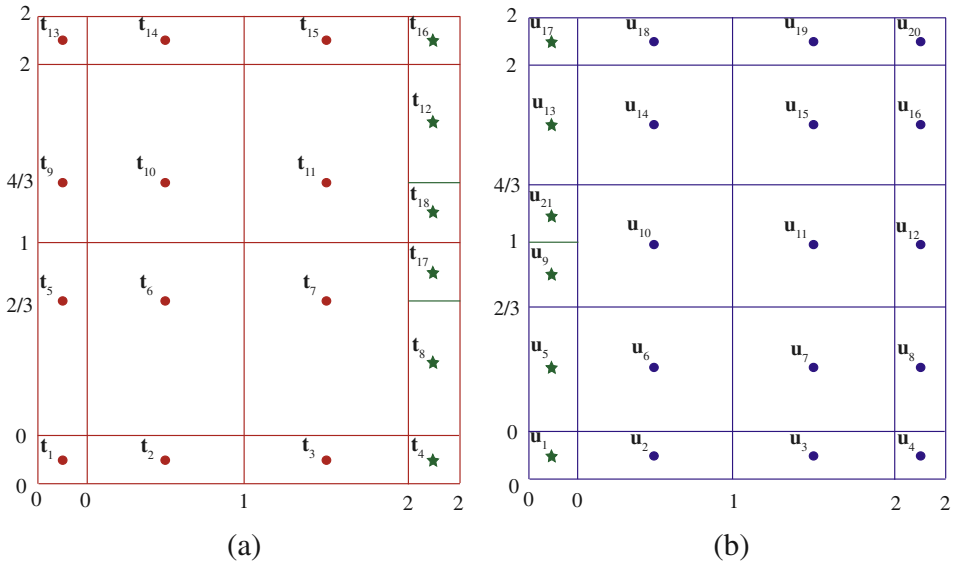


Fig. B.2. Local refinements of T-meshes from Fig. B.2 to arrive at matching patches. Control points of anchors designated by stars are adjusted such that the boundaries of the two geometries match.

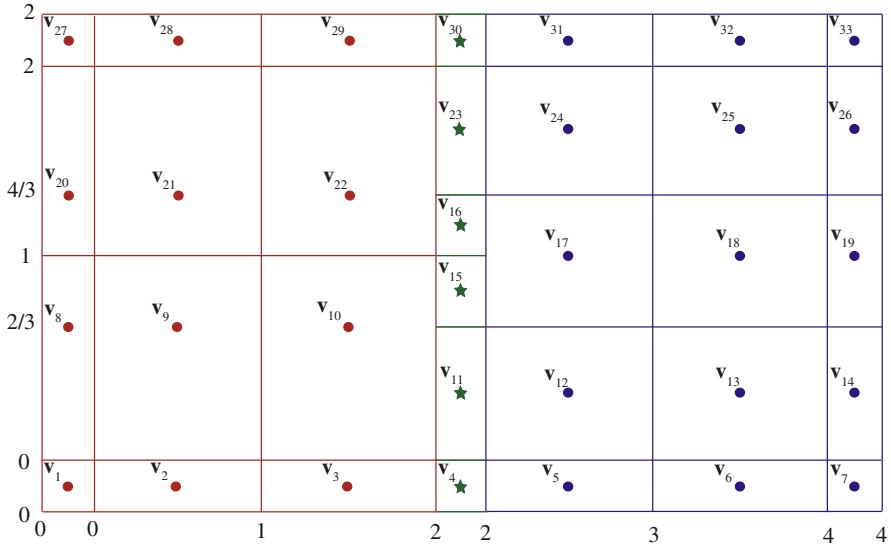


Fig. B.3. Merged C^0 geometry represented by a single T-mesh.

Table B.1
Inheritance diagram for the single T-mesh given in Fig. B.3. The anchors in the first column inherit the control points of the anchors in the second column. Analogously, the anchors in the second-to-last column inherit the control points of the anchors in the last column.

Anchor in Fig. B.3	Anchors in Fig. B.2	Anchor in Fig. B.3	Anchors in Fig. B.2
$\mathbf{v}_1$	$\mathbf{t}_1$	$\mathbf{v}_{18}$	$\mathbf{u}_{11}$
$\mathbf{v}_2$	$\mathbf{t}_2$	$\mathbf{v}_{19}$	$\mathbf{u}_{12}$
$\mathbf{v}_3$	$\mathbf{t}_3$	$\mathbf{v}_{20}$	$\mathbf{t}_9$
$\mathbf{v}_4$	$\mathbf{t}_4$ and $\mathbf{u}_1$	$\mathbf{v}_{21}$	$\mathbf{t}_{10}$
$\mathbf{v}_5$	$\mathbf{u}_2$	$\mathbf{v}_{22}$	$\mathbf{t}_{11}$
$\mathbf{v}_6$	$\mathbf{u}_3$	$\mathbf{v}_{23}$	$\mathbf{t}_{12}$ and $\mathbf{u}_{13}$
$\mathbf{v}_7$	$\mathbf{u}_4$	$\mathbf{v}_{24}$	$\mathbf{u}_{14}$
$\mathbf{v}_8$	$\mathbf{t}_5$	$\mathbf{v}_{25}$	$\mathbf{u}_{15}$
$\mathbf{v}_9$	$\mathbf{t}_6$	$\mathbf{v}_{26}$	$\mathbf{u}_{16}$
$\mathbf{v}_{10}$	$\mathbf{t}_7$	$\mathbf{v}_{27}$	$\mathbf{t}_{13}$
$\mathbf{v}_{11}$	$\mathbf{t}_8$ and $\mathbf{u}_5$	$\mathbf{v}_{28}$	$\mathbf{t}_{14}$
$\mathbf{v}_{12}$	$\mathbf{u}_6$	$\mathbf{v}_{29}$	$\mathbf{t}_{15}$
$\mathbf{v}_{13}$	$\mathbf{u}_7$	$\mathbf{v}_{30}$	$\mathbf{t}_{16}$ and $\mathbf{u}_{17}$
$\mathbf{v}_{14}$	$\mathbf{u}_8$	$\mathbf{v}_{31}$	$\mathbf{u}_{18}$
$\mathbf{v}_{15}$	$\mathbf{t}_{17}$ and $\mathbf{u}_9$	$\mathbf{v}_{32}$	$\mathbf{u}_{19}$
$\mathbf{v}_{16}$	$\mathbf{t}_{18}$ and $\mathbf{u}_{21}$	$\mathbf{v}_{33}$	$\mathbf{u}_{20}$
$\mathbf{v}_{17}$	$\mathbf{u}_{10}$		

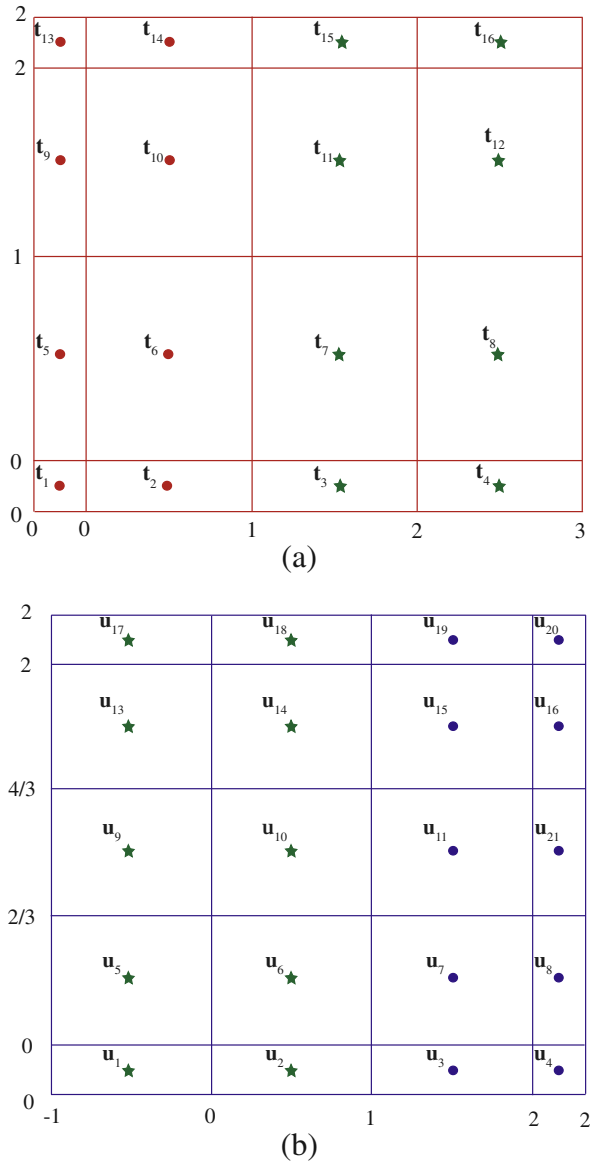


Fig. B.4. Elongation of T-meshes from Fig. B.1.

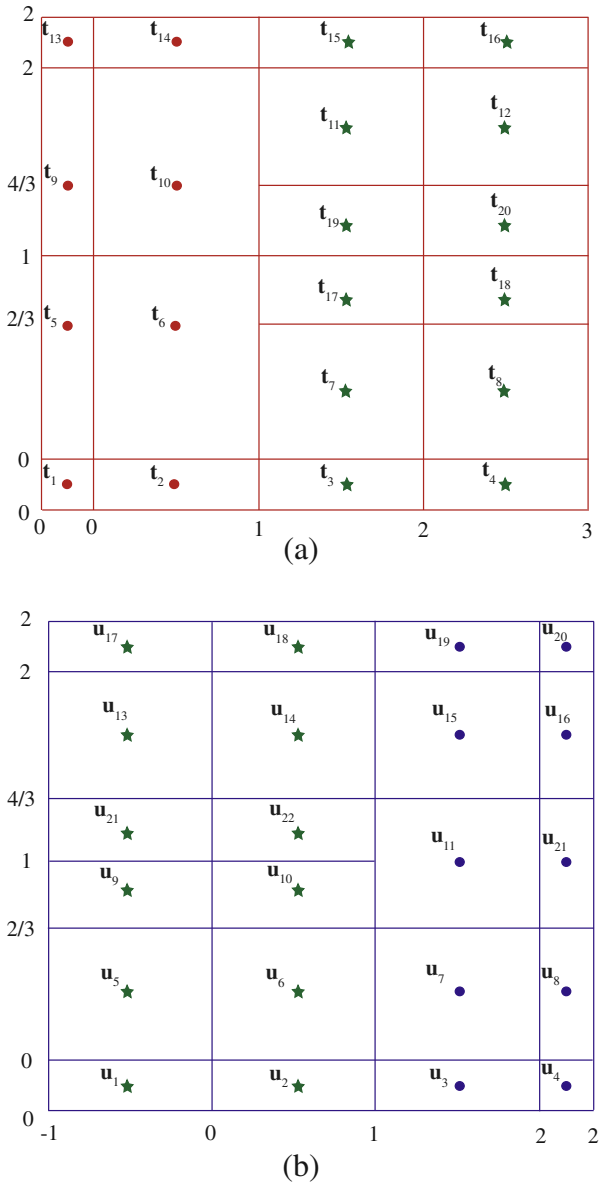


Fig. B.5. Local refinements of T-meshes from Fig. B.4 to arrive at matching patches. Control points of anchors designated by stars are adjusted such that the boundaries of the two geometries match.

inheritance is described in Table B.1. Note that during the merging process, the original geometry may be slightly altered as to arrive at a watertight geometry. In this particular case, a C^0 line is created.

Alternatively, suppose we wish to merge the two quadratic NURBS patches shown in Fig. B.1a and b such that the new geometry is C^1-continuous and is represented by a single T-spline patch. For this case, the given patches must be first extended such that there is an overlap region. To create the overlap region, we extend the patch in Fig. B.1a to the right such that the length of the final knot interval of this patch is equal to the length of the second knot interval of the patch in Fig. B.1b. Similarly, we extend the patch in Fig. B.1b to the left such that the length of the first knot interval of this patch is equal to the length of the second to last knot interval of the patch in Fig. B.1a. The result of such an extension is shown in Fig. B.4. To preserve the original geometry on the two elongated patches, the location of new control points must be recomputed to be consistent with the extension of the parametric domain.

Following this elongation, we locally refine the final two columns of the T-mesh in Fig. B.4a and the first two columns of the T-mesh in Fig. B.4b as demonstrated in Fig. B.5. Then, we adjust, if necessary, the control points associated with the second to last column of the T-mesh in Fig. B.5a and the first column of the T-mesh in Fig. B.5b such that they are equal and the control points associated with the last column of the T-mesh in Fig. B.5a and the second column of the T-mesh in Fig. B.5b such that they are equal. The geometry represented by these two T-meshes is now globally C^1-continuous. Note, by construction, there is an overlap between the resulting T-spline patches. The final geometry can be represented by the single T-mesh given in Fig. B.6 with appropriately chosen control points by removing two repeated columns of control points.

The above merging process, while originally designed for sewing bicubic T-spline patches, is extendable to patches of arbitrary order and dimension. For a C^n merging, $n+1$ columns of control points on one patch must correspond to $n+1$ columns of control points on the other patch.

The procedures for smoothly merging more topologically complex configurations is more complicated. These will be described in future work.

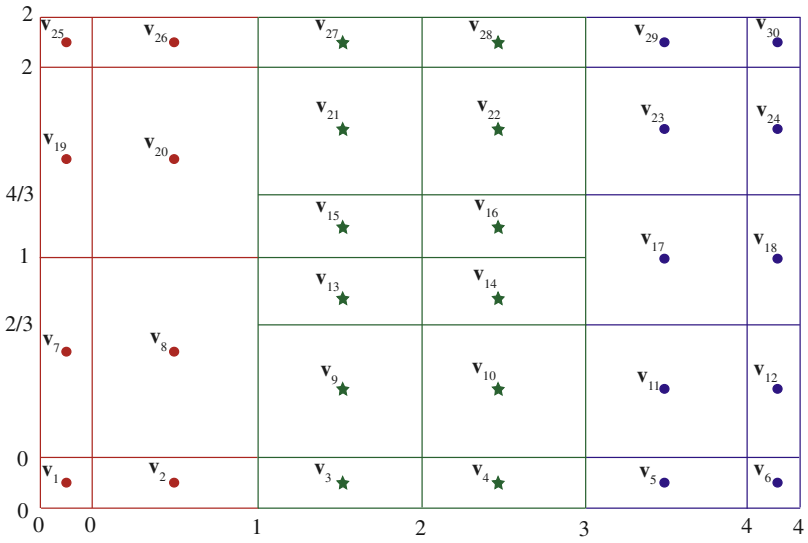


Fig. B.6. Merged C^1 geometry represented by a single T-mesh.

References

[1] Workshop on Mathematical Foundations of Computer Aided Design, Mathematical Sciences Research Institute, Berkeley, CA, 1999. June.

[2] I. Akkerman, Y. Bazilevs, V. Calo, T.J.R. Hughes, S. Hulshoff, The role of continuity in residual-based variational multiscale modeling of turbulence, Comput. Mech. 41 (2008) 371–378.

[3] F. Auricchio, L.B. da Veiga, A. Buffa, C. Lovadina, A. Reali, G. Sangalli, A fully "locking-free" isogeometric approach for plane linear elasticity problems: a stream function formulation, Comput. Methods Appl. Mech. Engrg. 197 (2007) 160–172.

[4] Y. Bazilevs, L. Beirao de Veiga, J.A. Cottrell, T.J.R. Hughes, G. Sangalli, Isogeometric analysis: approximation, stability and error estimates for h-refined meshes, Math. Models Methods Appl. Sci. 16 (2006) 1031–1090.

[5] Y. Bazilevs, V.M. Calo, J.A. Cottrell, T.J.R. Hughes, A. Reali, G. Scovazzi, Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows, Comput. Methods Appl. Mech. Engrg. 197 (2007) 173–201.

[6] Y. Bazilevs, V.M. Calo, T.J.R. Hughes, Y. Zhang, Isogeometric fluid–structure interaction: theory, algorithms, and computations, Comput. Mech. 43 (2008) 3–37.

[7] Y. Bazilevs, V.M. Calo, Y. Zhang, T.J.R. Hughes, Isogeometric fluid–structure interaction analysis with applications to arterial blood flow, Comput. Mech. 38 (2006) 310–322.

[8] Y. Bazilevs, T.J.R. Hughes, NURBS-based isogeometric analysis for the computation of flows about rotating components, Comput. Mech. 43 (2008) 143–150.

[9] Y. Bazilevs, C. Michler, V.M. Calo, T.J.R. Hughes, Isogeometric variational multiscale modeling of wall-bounded turbulent flows with weakly enforced boundary conditions on unstretched meshes, Comput. Methods Appl. Mech. Engrg., doi:10.1016/j.cma.2008.11.020.

[10] T. Belytschko, H. Stolarski, W.K. Liu, N. Carpenter, J.S.-J. Ong, Stress projection for membrane and shear locking in shell finite elements, Comput. Methods Appl. Mech. Engrg. 51 (1985) 221–258.

[11] A.N. Brooks, T.J.R. Hughes, Streamline upwind/Petrov–Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier–Stokes equations, Comput. Methods Appl. Mech. Engrg. 32 (1982) 199–259.

[12] S.B. Brunnermeier, S.A. Martin, Interoperability cost analysis of the US automotive supply chain, National Institute of Standards and Technology, NIST Planning Report 99-1, 1999.

[13] V.M. Calo, N.F. Brasher, Y. Bazilevs, T.J.R. Hughes, Multiphysics model for blood flow and drug transport with applications to patient-specific coronary artery flow, Comput. Mech. 43 (2008) 161–177.

[14] E. Catmull, J. Clark, Recursively generated B-spline surfaces on arbitrary topological meshes, Comput. Aided Des. 10 (1978) 350–355.

[15] F. Cirak, M. Ortiz, Fully C^1-conforming subdivision elements for finite deformation thin shell analysis, Int. J. Numer. Methods Engrg. 51 (2001) 813–833.

[16] F. Cirak, M. Ortiz, P. Schröder, Subdivision surfaces: a new paradigm for thin shell analysis, Int. J. Numer. Methods Engrg. 47 (2000) 2039–2072.

[17] F. Cirak, M.J. Scott, E.K. Antonsson, M. Ortiz, P. Schröder, Integrated modeling, finite-element analysis, and engineering design for thin-shell structures using subdivision, Comput. Aided Des. 34 (2002) 137–148.

[18] J.A. Cottrell, T.J.R. Hughes, A. Reali, Studies of refinement and continuity in isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 196 (2007) 4160–4183.

[19] J.A. Cottrell, A. Reali, Y. Bazilevs, T.J.R. Hughes, Isogeometric analysis of structural vibrations, Comput. Methods Appl. Mech. Engrg. 195 (2006) 5257–5296.

[20] M. Dorfel, B. Juttler, B. Simeon, Adaptive isogeometric analysis by local h-refinement with T-splines, Comput. Methods Appl. Mech. Engrg., this issue.

[21] T. Elguedj, Y. Bazilevs, V.M. Calo, T.J.R. Hughes, $\bar{B}$ and $\bar{F}$ projection methods for nearly incompressible linear and non-linear elasticity and plasticity using higher-order NURBS elements, Comput. Methods Appl. Mech. Engrg. 197 (2008) 2732–2762.

[22] J.A. Evans, Y. Bazilevs, I. Babuška, T.J.R. Hughes, N-widths, sup–infs, and optimality ratios for the k-version of the isogeometric finite element method, Comput. Methods Appl. Mech. Engrg. 198 (2009) 1726–1741.

[23] G.E. Farin, Curves and Surfaces for CAGD, A Practical Guide, fifth ed., Morgan Kaufman Publishers, San Francisco, 1999.

[24] G.E. Farin, NURBS Curves and Surfaces: From Projective Geometry to Practical Use, second ed., A.K. Peters, Ltd., Natick, MA, 1999.

[25] C.A. Felippa, Course notes for advanced finite element methods. <http://caswww.colorado.edu/Felippa.d/FelippaHome.d/Home.html>.

[26] T. Finnigan, Arbitrary degree T-splines, Master's thesis, Department of Computer Science, Brigham Young University, 2008.

[27] H. Gomez, V.M. Calo, Y. Bazilevs, T.J.R. Hughes, Isogeometric analysis of the Cahn–Hilliard phase-field model, Comput. Methods Appl. Mech. Engrg. 197 (2008) 4333–4352.

[28] J. Hoschek, D. Lasser, Fundamentals of Computer Aided Geometric Design, A.K. Peters, Wellesley, Massachusetts, 1993.

[29] T.J.R. Hughes, The Finite Element Method: Linear Static and Dynamic Finite Element Analysis, Dover Publications, Mineola, NY, 2000.

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

[31] T.J.R. Hughes, L.P. Franca, A mixed finite element formulation for Reissner–Mindlin plate theory: uniform convergence of all higher order spaces, Comput. Methods Appl. Mech. Engrg. 67 (1988) 223–240.

[32] T.J.R. Hughes, A. Reali, G. Sangalli, Duality and unified analysis of discrete approximations in structural dynamics and wave propagation: comparison of p-method finite elements with k-method NURBS, Comput. Methods Appl. Mech. Engrg. 197 (2008) 4104–4124.

[33] T.J.R. Hughes, A. Realli, G. Sangalli, Efficient quadrature for NURBS-based isogeometric analysis, Comput. Methods Appl. Mech. Engrg., this issue.

[34] D.J. Kasik, W. Buxton, D.R. Ferguson, Ten CAD model challenges, IEEE Comput. Graph. Appl. 25 (2) (2005).

[35] X. Li, X. Guo, H. Wang, Y. He, X. Gu, H. Qin, Harmonic volumetric mapping for solid modeling applications, in: Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling, Beijing China, 2007.

[36] G.G. Lorentz, Bernstein Polynomials, Chelsea Publishing Co., New York, 1986.

[37] L. Piegl, W. Tiller, The NURBS Book (Monographs in Visual Communication), second ed., Springer-Verlag, New York, 1997.

[38] H. Prautzsch, W. Boehm, M. Paluszny, Bézier and B-spline Techniques, Springer, New York, NY, 2002.

[39] E. Rank, A. Düster, V. Nübel, K. Preusch, O.T. Bruhns, High order finite elements for shells, Comput. Methods Appl. Mech. Engrg. 194 (21–24) (2005) 2494–2512.

[40] D.F. Rogers, An Introduction to NURBS With Historical Perspective, Academic Press, San Diego, CA, 2001.
[41] T.W. Sederberg, D.C. Anderson, R.N. Goldman, Implicit representation of parametric curves and surfaces, Comput. Vis. Graph. Image Process. 28 (1984) 72–84.
[42] T.W. Sederberg, D.L. Cardon, G.T. Finnigan, N.S. North, J. Zheng, T. Lyche, T-spline simplification and local refinement, ACM Trans. Graph. 23 (3) (2004) 276–283.
[43] T.W. Sederberg, G.T. Finnigan, X. Li, H. Lin, Watertight trimmed NURBS, ACM Trans. Graph. 27 (3) (2008). Article No. 79.
[44] T.W. Sederberg, J. Zheng, A. Bakenov, A. Nasri, T-splines and T-NURCCSs, ACM Trans. Graph. 22 (3) (2003) 477–484.
[45] T-Splines, Inc. <http://www.tsplines.com/maya/>.
[46] T-Splines, Inc. <http://www.tsplines.com/rhino/>.
[47] W.P. Thurston, Three-dimensional manifolds, Kleinian groups and hyperbolic geometry, Bull. Am. Math. Soc. (New Series) 6 (1982) 357–381.
[48] W.P. Thurston, Three-Dimensional Geometry and Topology, vol. 1, Princeton University Press, 1997.
[49] W.A. Wall, M.A. Frenzel, C. Cyron, Isogeometric structural shape optimization, Comput. Methods Appl. Mech. Engrg. 197 (2008) 2976–2988.
[50] H. Wang, Y. He, X. Li, X. Gu, H. Qin, Polycube splines, Comput. Aided Des. 40 (2008) 721–733.
[51] J. Warren, H. Weimer, Subdivision Methods for Geometric Design, Morgan Kaufman Publishers, San Francisco, 2002.
[52] Y. Zhang, Y. Bazilevs, S. Goswami, C. Bajaj, T.J.R. Hughes, Patient-specific vascular NURBS modeling for isogeometric analysis of blood flow, Comput. Methods Appl. Mech. Engrg. 196 (2007) 2943–2959.