Adaptive CAD model (re–)construction with THB–splines

Gábor Kiss[a,*], Carlotta Giannelli[b], Urška Zore[a], Bert Jüttler[a], David Großmann[c], Johannes Barner[c]

[a] *Institute of Applied Geometry, Johannes Kepler University of Linz, Austria*
[b] *INdAM c/o Dipartimento di Matematica e Informatica "U. Dini", Università degli Studi di Firenze, Italy*
[c] *MTU Aero Engines AG, Munich, Germany*

Abstract

Computer Aided Design (CAD) software libraries rely on the tensor-product NURBS model as standard spline technology. However, in applications of industrial complexity, this mathematical model does not provide sufficient flexibility as an effective geometric modeling option. In particular, the multivariate tensor–product construction precludes the design of adaptive spline representations that support local refinements. Consequently, many patches and trimming operations are needed in challenging applications. The investigation of generalizations of tensor–product splines that support adaptive refinement has recently gained significant momentum due to the advent of Isogeometric Analysis (IgA) [2], where adaptivity is needed for performing local refinement in numerical simulations. Moreover, traditional CAD models containing many small (and possibly trimmed) patches are not directly usable for IgA. Truncated hierarchical B–splines (THB–splines) provide the possibility of introducing different levels of resolution in an adaptive framework, while simultaneously preserving the main properties of standard B–splines. We demonstrate that surface fitting schemes based on THB–spline representations may lead to significant improvements for the geometric (re–)construction of critical turbine blade parts. Furthermore, the local THB–spline evaluation in terms of B–spline patches can be properly combined with commercial geometric modeling kernels in order to convert the multilevel spline representation into an equivalent – namely, *exact* – CAD geometry. This software interface fully integrates the adaptive modeling tool into CAD systems that comply with the current NURBS standard. It also paves the way for the introduction of isogeometric simulations into complex real world applications.

Keywords: Truncated hierarchical B-splines, adaptive refinement, CAD interfaces, turbine blades.

1. Introduction

Computer Aided Design (CAD) software usually provide a graphical and interactive user interface that allows designers and engineers to shape and manipulate industrial models by using a set of processing operations associated to the geometric modeling kernel. The CAD standard related to spline representations relies on the B–spline technology and its non–uniform rational extension (NURBS). The key properties of the B–spline model meet the requirements of CAD applications. In particular, in virtue of the convex hull property, the parametric spline representation is manipulated through a simpler modeling tool (*control net*), which reflects the shape of the underlying geometry. Unfortunately, the extension of univariate B–splines to the multivariate setting is based on the tensor-product approach, thus excluding adaptive mesh refinements. The resulting uniform distribution of the degrees of freedom

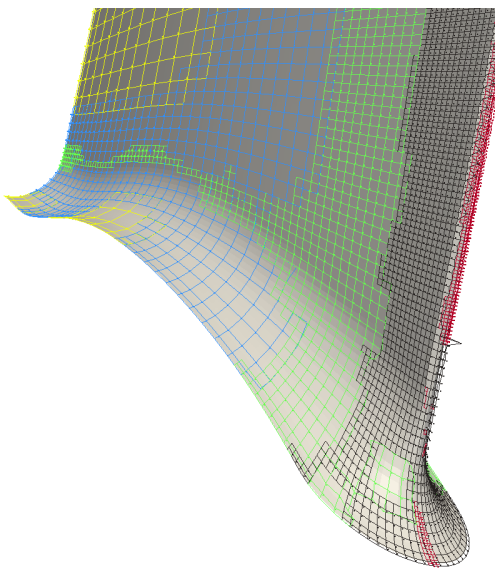


Figure 1: Turbine blade fillet represented by truncated hierarchical B– (THB–) splines. The different meshes represent control points at different levels of the THB–spline hierarchy.

*Corresponding author
Email addresses: gabor.kiss@dk-compmath.jku.at (Gábor Kiss), carlotta.giannelli@gmail.com (Carlotta Giannelli), Urska.Zore@jku.at (Urška Zore), bert.juettler@jku.at (Bert Jüttler), David.GROSSMANN@mtu.de (David Großmann), Johannes.Barner@mtu.de (Johannes Barner)

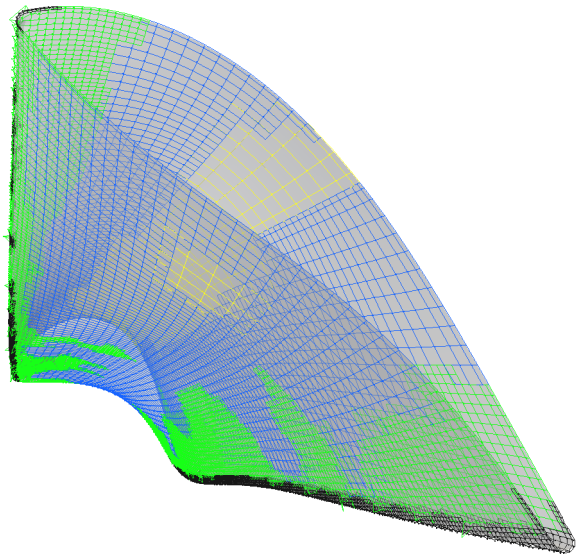


Figure 2: Adaptive airfoil model based on THB–spline representations. Note the different resolution of the hierarchical meshes.

may also cause unwanted oscillations in the fitting of complex non–uniform data sets.

Interpolation or approximation of measured data is required in many industrial applications, as for example, instrument calibration, data analysis or reverse engineering. Automatic surface fitting of complex data with high precision is a non–trivial issue that should properly address many critical steps, see e.g., [31]. These include, for example, a satisfactory treatment of noisy and unevenly distributed point clouds, as well as the construction of a suitable parametrization. In addition, when considering a fitting algorithm based on tensor–product B–spline surfaces, the limit of representing solely rectangular regions naturally appears, while measured data usually cover more general shapes.

The integration of adaptive approximation schemes into an industrial geometric modeling environment can provide new and more flexible design capabilities. The same, and in some critical cases even better, accuracy can be obtained by significantly reducing the number of design parameters. Figures 1 and 2 show two adaptive models related to the reconstruction of turbine blade geometries for the modeling process of aero engines.

Approximation techniques based on different kinds of adaptive spline models are currently receiving particular attention — not only in computer aided geometric design, but also for the solution of partial differential equations in isogeometric analysis. To promote a deeper integration between CAD boundary representations and isogeometric simulations, one demanding challenge is also the automatic construction of (adaptive) solid models that are able to exactly preserve a given spline boundary representation [32]. Extensions of classical tensor–product B–splines that offer adequate local refinement include: *hierarchical B–splines* [8, 16], *T–splines* [23, 24], polynomial splines over hierarchical T–meshes *(PHT–splines)* [3, 4] and *LR–splines* [1, 5]. All of them can be applied for surface modeling and geometry reconstruction. Hierarchical B–splines were considered already in [9, 14], T–splines have recently been used in [28, 29, 30], and applications of PHT–splines were explored in [4, 27].

By exploiting an underlying tensor–product structure at different refinement levels, *hierarchical B–splines* inherit several desirable properties (linear independence, local support, non–negativity, completeness of the spline space), which are fundamental to define an effective spline representation for uniform and non–uniform refinement, for different degrees and smoothness as well as in the multivariate setting [11, 18]. Its potential in isogeometric analysis has recently been demonstrated [20, 26]. Additional results concerning the partition of unity property, approximation power and strong stability have been derived by considering the truncated basis for the hierarchical B–spline space (*THB–splines*) [12, 13, 15, 25]. Generalizations of the truncated bases to wider classes of spline spaces and an extension to cover arbitrary topologies have also been presented [22, 33].

The remaining three adaptive spline constructions:

- allow the modeling of objects with general topology as they support extraordinary vertices in the mesh with geometric continuity (T–splines, special PHT–splines [17]) and provide a powerful geometric modeling tool, also supported by commercial CAD tools (T–splines); this is currently not available for hierarchical splines;

- do not always guarantee the linear independence and need scaling or rational normalization to achieve the partition of unity property of the basis functions (T–splines, LR–splines); the importance of these features in the context of isogeometric analysis motivated the study of restricted mesh configurations together with more involved refinement procedures (analysis suitable T–splines [21], Bressan's class of LR splines [1]);

- do not provide straightforward multivariate generalizations (T–splines, PHT–splines) and may require a case-by-case analysis for the construction of the basis depending on degrees and smoothness (in particular, PHT–splines have been mainly considered in the case of reduced smoothness, e.g., for C^1 cubics).

This paper explores the use of the THB–spline approximation framework focusing on its successful integration with current CAD systems. The remainder of the paper consists of four main sections. The following section recalls the definition of THB–splines. Sections 3 and 4 present the adaptive surface fitting algorithm and related examples. The fifth section discusses the integration of THB–splines

into a commercial CAD environment. Finally, Section 6 concludes the paper.

2. Truncated hierarchical B–splines

We recall the construction of a hierarchical spline space, focusing on truncated hierarchical B–splines (THB–splines), a recently introduced basis [12]. We present the construction in the bivariate case of bidegree (p, p) only, though we can define THB–splines in a general multivariate setting and consider different degrees in any dimension.

2.1. Nested spline spaces

We consider adaptive refinement of a tensor–product B–spline surface of bidegree $\mathbf{p} = (p, p)$ that is defined on the domain $[0, 1]^2 \subset \mathbb{R}^2$. To allow for adaptivity, we consider a nested sequence of $L + 1$ B–spline spaces, defined by nested knot vectors. More precisely, the space at level $\ell = 0, \dots, L$ is defined with respect to two non–decreasing knot vectors

$$0 = u_0^\ell \le \cdots \le u_{m^\ell}^\ell = 1, \quad 0 = v_0^\ell \le \cdots \le v_{n^\ell}^\ell = 1,$$

where m^ℓ and n^ℓ denote the number of knots in the parameter directions u and v at level ℓ. We require that the knots at level $\ell - 1$ are also present at level ℓ at least with the same multiplicity.

The boundary knots 0 and 1 have multiplicity $p + 1$, but multiplicities of the inner knots do not exceed p. The non–empty Cartesian products of the knot intervals

$$(u_i^\ell, u_{i+1}^\ell) \times (v_j^\ell, v_{j+1}^\ell)$$

will be denoted as *cells of level* ℓ.

We define the *knot-index sets* of level ℓ,

$$\mathcal{U}^\ell = \{0, \dots, m^\ell\}, \quad \mathcal{V}^\ell = \{0, \dots, n^\ell\}$$

and $\mathcal{K}^\ell = \mathcal{U}^\ell \times \mathcal{V}^\ell$.

Later, we will use the knot indices to identify the B–splines of level $\ell + 1$ that are obtained by refining a given B–spline of level ℓ. In order to do so, we define a simple relation between the index sets that also takes into account the multiplicity of the knots: for any two indices $i \in \mathcal{U}^\ell$, $j \in \mathcal{U}^k$ (and similarly for indices from $\mathcal{V}^\ell, \mathcal{V}^k$), we say that i *precedes* j and denote this by $(i, \ell) \preccurlyeq (j, k)$, if $u_i^\ell < u_j^k$ or if $u_i^\ell = u_j^k$ and one of the following conditions concerning the knot multiplicities and levels is satisfied:

$\ell < k$ and $\max\{r |\ u_{i+r}^\ell = u_i^\ell\} \ge \max\{r |\ u_{j+r}^k = u_j^k\}$,

or

$\ell > k$ and $\max\{r |\ u_{i-r}^\ell = u_i^\ell\} \le \max\{r |\ u_{j-r}^k = u_j^k\}$.

This definition is extended to the knot-index sets $\mathcal{K}^\ell$. More precisely, $\mathbf{i} = (i, i') \in \mathcal{K}^\ell$ precedes $\mathbf{j} = (j, j') \in \mathcal{K}^k$, denoted by $(\mathbf{i}, \ell) \preccurlyeq (\mathbf{j}, k)$, if $(i, \ell) \preccurlyeq (j, k)$ and $(i', \ell) \preccurlyeq (j', k)$.

For each level ℓ we consider the set of (tensor-product) B–splines of bidegree $\mathbf{p}$ that are defined on the given knot vectors. The B–spline $\beta_{\mathbf{i}}^\ell$ with index $\mathbf{i} = (i, i')$ (that corresponds to the two smallest knots in its support) is defined by the pair

$$(u_i^\ell, \dots, u_{i+p+1}^\ell), \quad (v_{i'}^\ell, \dots, v_{i'+p+1}^\ell) \tag{1}$$

of local knot vectors. The indices of all B–splines of level ℓ form the set

$$\mathcal{N}^\ell = \{0, \dots, m^\ell - p - 1\} \times \{0, \dots, n^\ell - p - 1\}.$$

Due to the assumption concerning nested knot sequences, any B–spline of level ℓ admits a representation as a unique linear combination of B–splines of any higher level k, the coefficients of which can be computed with the knot insertion algorithm. We define the *refinement relation* $\lessdot$ between indices $\mathbf{i} \in \mathcal{N}^\ell$ and $\mathbf{j} \in \mathcal{N}^k$ that identifies the nonzero (and even positive) coefficients in this representation.

More precisely, $(\mathbf{i}, \ell) \lessdot (\mathbf{j}, k)$ if and only if the representation of $\beta_{\mathbf{i}}^\ell$ with respect to B–splines of level $k > \ell$ has a positive coefficient for $\beta_{\mathbf{j}}^k$. We then say that the B–spline $\beta_{\mathbf{i}}^\ell$ *refines to* the B–spline $\beta_{\mathbf{j}}^k$.

This relation can be characterized with the help of $\preccurlyeq$,

$$(\mathbf{i}, \ell) \lessdot (\mathbf{j}, k) \quad \Leftrightarrow$$
$$(\mathbf{i}, \ell) \preccurlyeq (\mathbf{j}, k) \ \text{ and } \ (\mathbf{j} + \mathbf{p} + \mathbb{1}, k) \preccurlyeq (\mathbf{i} + \mathbf{p} + \mathbb{1}, \ell),$$

where $\mathbb{1} = (1, 1)$. Clearly, $(\mathbf{i}, \ell) \lessdot (\mathbf{j}, k)$ implies $\operatorname{supp} \beta_{\mathbf{i}}^\ell \supseteq \operatorname{supp} \beta_{\mathbf{j}}^k$ but not vice versa, if multiple knots are present.

2.2. Hierarchical splines

The construction of hierarchical splines goes back to Forsey and Bartels [8]. They proposed to perform local modifications of a B–spline surface by adding contributions of functions from the B–spline spaces of higher levels. Formally, the extension of the B–spline space is achieved by splitting each index set $\mathcal{N}^\ell$ of level ℓ into

$$\mathcal{N}^\ell = \mathcal{P}^\ell \,\dot\cup\, \mathcal{S}^\ell, \tag{2}$$

where $\dot\cup$ denotes the disjoint union, i.e., the two sets on either side of the symbol have no members in common. The B–splines with indices from $\mathcal{P}^\ell$ are said to be *passive*, since we do not include them into our space, and the B–splines with indices from $\mathcal{S}^\ell$ have been *selected*. All functions of the coarsest level have been selected, $\mathcal{S}^0 = \mathcal{N}^0$ and $\mathcal{P}^0 = \emptyset$.

The hierarchical spline space is defined by

$$\mathcal{H} = \left\{ \sum_{\ell=0}^{L} \sum_{\mathbf{i} \in \mathcal{S}^\ell} c_{\mathbf{i}}^\ell \, \beta_{\mathbf{i}}^\ell \mid c_{\mathbf{i}}^\ell \in \mathbb{R}^3 \right\}.$$

Based on the index sets $\mathcal{S}^\ell$ we identify the *subdomains* $\Omega^\ell \subseteq [0, 1]^2$ of the parameter domain that correspond to the various levels of refinement,

$$\Omega^\ell = \bigcup_{\mathbf{i} \in \mathcal{S}^\ell} \operatorname{supp} \beta_{\mathbf{i}}^\ell.$$

The choice of $\mathcal{S}^\ell$ is subject to the first two *index set constraints:*

(i) among all index sets $\mathcal{S}^\ell$ defining the same subdomain Ω^ℓ, we always use the *largest* one:

$$\forall \mathbf{i} \in \mathcal{N}^\ell : \quad \text{supp } \beta_{\mathbf{i}}^\ell \subseteq \Omega^\ell \;\Rightarrow\; \mathbf{i} \in \mathcal{S}^\ell;$$

(ii) the subdomains are *nested*, $\Omega^\ell \supseteq \Omega^{\ell+1}$:

$$\forall \mathbf{i} \in \mathcal{N}^\ell \; \forall \mathbf{j} \in \mathcal{S}^{\ell+1} : \; (\mathbf{i}, \ell) \Subset (\mathbf{j}, \ell+1) \;\Rightarrow\; \mathbf{i} \in \mathcal{S}^\ell.$$

The set of B–splines with indices from $\bigcup_{\ell=0}^{L} \mathcal{S}^\ell$ may be linearly dependent.

2.3. Bases of hierarchical splines

A basis for the hierarchical space $\mathcal{H}$ has been constructed by Kraft [16] (and reconsidered under weaker assumptions in [26]) by eliminating those B–splines that can be represented as linear combinations of selected B–splines of higher levels. More precisely, the index set $\mathcal{S}^\ell$ is split into two disjoint subsets,

$$\mathcal{S}^\ell = \mathcal{A}^\ell \dot\cup \mathcal{R}^\ell.$$

The functions with indices from $\mathcal{A}^\ell$ are called *active*, since we will keep them in the basis. The remaining functions are called *refined*, since they have been replaced by linear combinations of selected functions from higher levels. The subsets $\mathcal{A}^\ell$ and $\mathcal{R}^\ell$ are defined by the *third index set constraint:*

(iii) the set $\mathcal{R}^\ell$ contains exactly the indices of those functions that can be represented by the selected functions of the next higher level,

$$\forall \mathbf{i} \in \mathcal{S}^\ell : \; \mathbf{i} \in \mathcal{R}^\ell \Leftrightarrow \bigcup \left\{ \mathbf{j} \in \mathcal{N}^{\ell+1} |\; (\mathbf{i}, \ell) \Subset (\mathbf{j}, \ell+1) \right\} \;\subseteq\; \mathcal{S}^{\ell+1}.$$

Kraft's basis of $\mathcal{H}$ is obtained by collecting the active B–splines (i.e., with indices in $\mathcal{A}^\ell$) from all levels.

The truncated hierarchical B–splines (THB–splines) form another basis of the same hierarchical spline space but with improved properties, see [12]. The idea is to modify Kraft's basis functions in order to obtain a basis that forms a partition of unity. Based on the nested nature of the spaces and the refinement properties of the B-spline basis, this is achieved by *truncation* of the active functions from a coarser level with respect to the selected functions from the finer levels. More precisely, in their representations with respect to higher levels, we remove the contributions of the selected functions. As a consequence, the functions in the resulting hierarchical basis have reduced supports, thereby also forming a basis with better sparsity properties.

We define the THB-basis and compare it with the earlier approaches by presenting and analyzing the evaluation algorithm.

The algorithm evaluates the value $f(u, v)$ of a function $f \in \mathcal{H}$. Proceeding from coarse to fine levels it constructs the B–spline representation of f at level ℓ from the previous one via knot insertion (except for $\ell = 0$) and modifies it according to the chosen representation. Finally, $f(u, v)$ is found with the help of standard B–spline evaluation at level L. In each step, it considers only the $(p+1)^2$ coefficients of those splines that take non-zero values at (u, v).

By choosing one of the two lines marked with †, and similarly for ‡, the algorithm performs the evaluation of the representation of f

- with respect to the *full* system of selected B–splines (which was used in the definition of $\mathcal{H}$),
- with respect to the *Kraft's basis*, and
- with respect to the *THB–splines:*

```
algorithm EVALUATE_{T}HB_SURFACE(*coeffs c[], int
L, float u, float v, *indexset S, *indexset A)
  \\ c[] are the coefficients, L is L, (u,v) specifies the
  \\  evaluation point, S[l] and A[l] are S^l and A^l.
  for l from 0 to L do {
    Identify the set I[l] of the (p+1)^2 indices of level l
    B–splines with supports containing (u,v) and create
    temporary variables d[l] for their coefficients.
    for all i in I[l] do {
      if l == 0 then { d[l][i] = 0 }
      else { compute d[l] by applying knot insertion
             to d[l-1] }
      †if i in S[l] then { \\  full
      †if i in A[l] then { \\  Kraft/THB
        ‡d[l][i] = d[l][i] + c[l][i] \\  full/Kraft
        ‡d[l][i] = c[l][i] \\ THB
  } } }
  Use standard B–spline evaluation to compute x(u,v)
  from the coefficients d[L] and return(x).
end
```

2.4. Properties of THB–splines

The evaluation algorithm for *Kraft's basis* requires to modify the coefficients of level ℓ by *adding* the coefficients of the active basis functions to them. In contrast, the *THB* evaluation proceeds by simply *replacing* them with the coefficients of the active basis functions. This is equivalent to the construction of the basis in [12].

A detailed analysis of algorithms for THB–splines has been presented in [15]. Figure 3 shows an example of THB–splines defined on a hierarchical mesh consisting of three levels of detail.

Compared to earlier approaches, THB–splines possess the following advantageous properties.

- THB–splines form a *non-negative partition of unity.* This property makes them attractive for geometric modeling applications. The examples in Figures 1 and 2 show the control meshes (which consist of all control points that are associated with active B–splines) of the various levels.

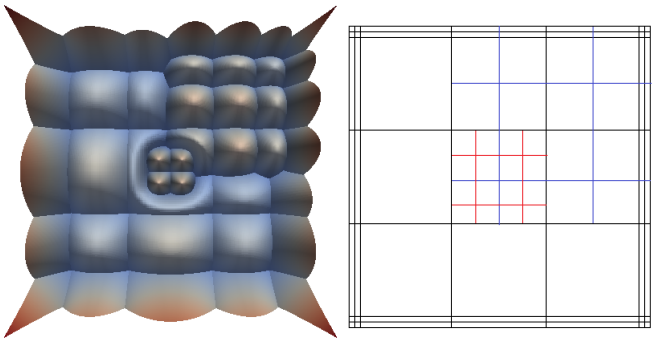


Figure 3: Truncated hierarchical B–splines of bidegree $(2, 2)$ with three refinement levels (left). The hierarchical mesh in the parameter domain is also shown (right), where the colors indicate the contributions of the different levels.

- Compared to Kraft's basis, THB–splines have the same or a smaller support. Thus, they *improve the sparsity properties* of the Kraft's basis.
- Under reasonable assumptions on the given knot configuration, THB–splines are *strongly stable with respect to the maximum norm*, where the constants appearing in the stability inequalities are independent of the selection of the subdomains and of the number of levels [13].
- THB–splines span the full space of piecewise polynomial functions with the smoothness specified by given knot configuration on the considered polynomial grid, provided that the subdomains satisfy certain geometric conditions. See [11] for the case of single knots and [18] for the general setting.

3. Adaptive THB–spline approximation

Surface fitting algorithms based on hierarchical B–spline methods were previously considered in [9, 10, 14]. More recently, a basic framework for approximation with THB–splines was also presented [12]. We extend the basic framework by integrating a smoothing term, which is essential for successful applications in problems of industrial complexity, and by considering different refinement strategies. The examples presented in Section 4 and 5 will show how the proposed hierarchical fitting scheme outperforms standard B–spline approximations, not only with respect to a reduced number of degrees of freedom, but also concerning the quality of the computed solution.

3.1. Least-squares approximation

We consider a surface reconstruction method from measured data by computing a least-squares approximation

$$\mathbf{s}(u, v) = \sum_{\ell=0}^{L} \sum_{\mathbf{i} \in \mathcal{A}^\ell} \boldsymbol{c}_\mathbf{i}^\ell \, \tau_\mathbf{i}^\ell(u, v), \quad (u, v) \in [0, 1]^2, \qquad (3)$$

where the basis functions $\tau_\mathbf{i}^\ell$ are THB–splines. We consider given data

$$\boldsymbol{x} = (\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}) = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m)^T \in \mathbb{R}^{m \times 3},$$

where $\boldsymbol{x}_i = (x_{i1}, x_{i2}, x_{i3})$ are the Cartesian coordinates of the measured (or sampled) points with associated parameter values

$$(u_1, v_1), \ldots, (u_m, v_m) \in [0, 1]^2.$$

Figure 4 shows a typical data set, which contains points on a fillet of a turbine blade. In all our examples we used standard parameterization methods to generate the associated parameter values, see [6, 7]. Due to the shape of the fillet and the fact that our parameter domain is simply the unit square, we obtain a highly non-uniform distribution of the parameter values, see Figure 4. As we shall see later, this fact makes it difficult to deal with these data when using standard tensor-product spline representations.

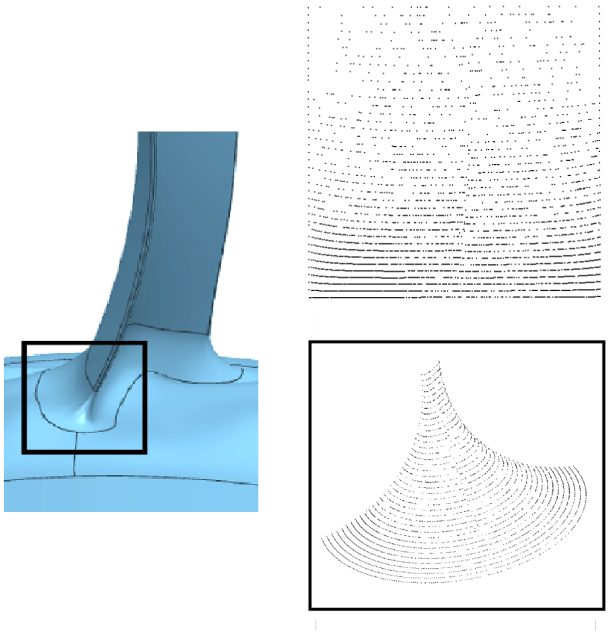


Figure 4: A turbine blade with highlighted area of the fillet (left). The measurement of the fillets often produce non–uniformly sampled data sets where the upper part has significantly less measured data points as the bottom (bottom–right). The corresponding distribution of parameter values is also shown (top–right).

We seek the vector of coefficients (control points)

$$\mathbf{c} = (\ldots, \boldsymbol{c}_\mathbf{i}^\ell, \ldots)^T \in \mathbb{R}^{n \times 3},$$

where $n = \sum_{\ell=0}^{L} |\mathcal{A}^\ell|$ is the number of THB–splines from all levels, for the THB–spline representation (3) that minimizes the objective function

$$F(\mathbf{c}) = \sum_{k=1}^{m} ||\mathbf{s}(u_k, v_k) - \boldsymbol{x}_k||^2 + \lambda J(\mathbf{c}),$$

where $J(\mathbf{c})$ is the smoothing (or regularization) term and λ is a positive weight that controls the influence of the smoothness (regularization) term. Enlarging the value of λ increases the *fairness* of the approximating surface, while simultaneously increasing the approximation error.

Without regularization, the linear system obtained from the objective function would easily become singular, e.g., if the number of local degrees of freedom in a certain region of the surface exceeds the number of available data points in that region. In particular, we use the thin plate spline energy

$$J(\mathbf{c}) = \int\int_{[0,1]^2} s_{uu}^2 + 2\, s_{uv}^2 + s_{vv}^2 \mathrm{d}u\mathrm{d}v \tag{4}$$

as regularization term.

The objective function F is quadratic with respect to the unknown coefficients. The solution of the optimization problem is computed by solving the sparse linear system

$$(A^T A + \lambda E)\mathbf{c}^{(i)} = A^T \boldsymbol{x}^{(i)} \qquad (i = 1, 2, 3)$$

for each of the three columns $\mathbf{c}^{(i)}$ of the vector of coefficients, where the k-th row of $A = (\tau_{\mathbf{i}}^{\ell}(u_k, v_k))_{k,(\mathbf{i},\ell)}$ contains the values of the THB–splines at (u_k, v_k) and the matrix E is contributed by the regularization term.

3.2. Assembling the system

The matrix $A^T A$ has the elements

$$a_{(\mathbf{i},\ell),(\mathbf{i}',\ell')} = \sum_{k=1}^{m} \tau_{\mathbf{i}}^{\ell}(u_k, v_k)\, \tau_{\mathbf{i}'}^{\ell'}(u_k, v_k),$$
$$(\ell, \ell' = 0, \ldots, L;\ \mathbf{i} \in \mathcal{A}^{\ell};\ \mathbf{i}' \in \mathcal{A}^{\ell'}).$$

This matrix is similar to a mass matrix in numerical simulation. In that case, the evaluation points are the Gauss nodes of the elements. In our case, the evaluation points are the parameter values of the given data. In order to efficiently assemble the matrix $A^T A$ (and similarly the right-hand side of the system) we proceed as follows. First, for each index k, we identify all THB–splines that contribute a non-zero value at (u_k, v_k),

$$\bigcup_{\ell=0}^{L} \{(\mathbf{i}, \ell) \mid \mathbf{i} \in \mathcal{A}^{\ell} \text{ and } \tau_{\mathbf{i}}^{\ell}(u_k, v_k) \neq 0\}. \tag{5}$$

When computing this index set, we start with level 0 and increase the level until the indices of all $(p+1)^2$ B–splines of level $\ell + 1$ that do not vanish at this point belong to $\mathcal{P}^{\ell+1}$, i.e., this set of indices correspond to B–splines at level $\ell + 1$ which are all passive.

We simplify the computation of this index set by considering the supports[1] of the original B–splines $\beta_{\mathbf{i}}^{\ell}$, which bound the supports of the THB–splines $\tau_{\mathbf{i}}^{\ell}$. The obtained set (5) is then a superset of the relevant indices.

Second, we evaluate all THB–splines with indices in this set at (u_k, v_k), and third, we create and add the contributions to the matrix elements $a_{(\mathbf{i},\ell),(\mathbf{i}',\ell')}$ for all relevant index pairs.

The regularization term generates the matrix E with the elements

$$e_{(\mathbf{i},\ell),(\mathbf{i}',\ell')} = \int\int_{[0,1]^2} \partial_{uu}\tau_{\mathbf{i}}^{\ell}\, \partial_{uu}\tau_{\mathbf{i}'}^{\ell'} + 2\, \partial_{uv}\tau_{\mathbf{i}}^{\ell}\, \partial_{uv}\tau_{\mathbf{i}'}^{\ell'} + \partial_{vv}\tau_{\mathbf{i}}^{\ell}\, \partial_{vv}\tau_{\mathbf{i}'}^{\ell'}\, \mathrm{d}u\mathrm{d}v$$

which is similar to a stiffness matrix. It is assembled by using Gaussian quadrature on the polynomial pieces of the hierarchical spline functions.

We consider all cells (i.e., the Cartesian products of the knot spans) of level 0 and split them by repeatedly inserting the knots of the higher levels. The refinement stops when we arrive at a cell of level ℓ such that the indices of all B–splines of the next level $\ell + 1$ that do not vanish on this cell belong to $\mathcal{P}^{\ell+1}$. This cell then defines a polynomial piece of the hierarchical spline function. Simultaneously we collect the indices of the THB–splines that do not vanish on this cell in a list, similar to the assembly of the matrix A. We evaluate all THB–splines and their derivatives at the Gauss nodes of the cell and add the contributions to the matrix elements $e_{(\mathbf{i},\ell),(\mathbf{i}',\ell')}$ for all relevant index pairs.

3.3. Refinement strategies

The regularized least-squares approximation is repeated until the error

$$\max_{i=1,\ldots,m} ||\mathbf{s}(u_i, v_i) - \boldsymbol{x}_i||^2$$

is below a fixed tolerance or a prescribed number of iterations is reached. After each step, the accuracy of the approximation result is improved by performing refinement in the critical regions of the surface, thus providing additional degrees of freedom and a higher resolution. A typical requirement in high-end applications — such as turbine blades — is that $90-95\%$ of data points are approximated with an error below $\sigma = 10^{-6}$.

The procedure is initialized by choosing $\mathcal{A}^0 = \mathcal{N}^0$ and $\mathcal{P}^{\ell} = \mathcal{N}^{\ell}$ for $\ell = 1, \ldots, L$, where L is the number of levels in the spline hierarchy.

We investigated two different refinement strategies.

- The *absolute threshold (AT)* approach: the points where the error exceeds a fixed threshold σ are marked for refinement.
- The *relative threshold (RT)* approach: a certain percentage of points with the largest errors is marked for refinement.

[1] The support of a (tensor-product) B–spline is an axis–aligned box in the parameter domain, while the support of a THB–splines can have a more complicated shape.

Both approaches select a set of indices that identify the points $\boldsymbol{x}_k$ with associated parameter values (u_k, v_k) where the THB–spline surface needs to be refined. For each index we first compute the current level of the associated point,

$$\max\{\ell \mid (u_k, v_k) \in \Omega^\ell\},$$

and we identify the cell of level ℓ that contains (u_k, v_k). This cell and a certain number of neighboring cells (determined by the value of an extension parameter, see Fig. 5) is then added to the subdomain $\Omega^{\ell+1}$, provided that $\ell+1 \leq L$. This is done by adding the indices of all B–splines of level $\ell+1$ whose support is contained in this region to the index set $\mathcal{A}^{\ell+1}$ that identifies the active functions.

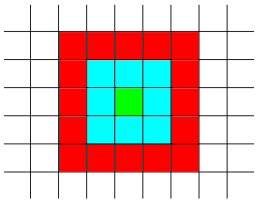


Figure 5: Extension of a selected cell (green) for extension parameters 1 (cyan) and 2 (red).

After modifying the index sets $(\mathcal{A}^\ell)_{\ell=0,\dots,L}$, however, the index set constraints (i-iii) need not be satisfied. The next section describes three algorithms that allow to restore the validity of the three constraints.

3.4. Implementation aspects

In order to implement the hierarchy of truncated B–splines, we use *characteristic matrices* already discussed in [15]. In contrast to the approach in [15], we omit the data structure describing the domain hierarchy. In fact, this data structure is redundant as the matrices implicitly contain the needed information. However, one needs to perform certain operations to maintain the validity of the hierarchy of the subdomains.

For each level $\ell = 0, \dots, L$ we define a characteristic matrix M^ℓ to identify the B–spline basis, defined with respect to the two knot vectors (1) at level ℓ. More precisely, an entry of a matrix M^ℓ with an index $\mathbf{i}$ corresponds to the function with the same index $\mathbf{i} \in \mathcal{N}^\ell$.

In order to store the implicit information about the hierarchy, we describe the partition of $\mathcal{N}^\ell$ by choosing different entries $M^\ell_{\mathbf{i}}$ of the matrix M^ℓ:

$$M^\ell_{\mathbf{i}} = \begin{cases} 0, & \mathbf{i} \in \mathcal{P}^\ell, \\ 1, & \mathbf{i} \in \mathcal{A}^\ell, \\ 2, & \mathbf{i} \in \mathcal{R}^\ell. \end{cases}$$

The characteristic matrix is stored using a sparse data structure, since the number of non-zero entries is generally small.

The matrices M^ℓ represent a valid hierarchy of spaces and domains, provided that the partitions of $\mathcal{N}^\ell$ satisfy the index set constraints (i–iii) described in Section 2. Consequently, the basis for THB–spline space can easily be identified by only considering functions with indices $\mathbf{i} \in \mathcal{A}^\ell$ for each level ℓ, i.e., with entry 1 in the characteristic matrix.

During the refinement procedure, the characteristic matrices are updated by changing the values of certain entries from 0 (passive) to 1 (active), wherever the enlargement of a subdomain involves adding the support of the corresponding non-active B–spline. Proceeding from fine to coarse level we then perform the three maintenance operations: *closing*, *nesting* and *cleaning*, in order to obtain index sets that again satisfy the constraints (i–iii):

```
algorithm MAINTAIN(*mat M, int L)
  \\ M is the list of characteristic matrices
  \\ L is the finest level
  CLOSE(M, L) \\ closing
  for l from L-1 to 0 do {
    NEST(M, l) \\ nesting
    CLOSE(M, l) \\ closing
    CLEAN(M, l) \\ cleaning
  }
end
```

The *closing* operation – which is needed to satisfy the first index set constraint (i) – is performed by the algorithm `CLOSE`. For each passive function at level ℓ, we check whether every cell of its support is covered by selected functions of the same level. If this is the case, then the status of that function is changed to active.

```
algorithm CLOSE(*mat M, int l)
  \\ M is the list of characteristic matrices, l is the level
  for all i with M[l][i] = 0 do {
    M[l][i] = 1
    for all cells c ⊂ supp β^ℓ_i do {
      covered = 0
      for all j with supp β^ℓ_j ⊃ c do {
        if M[l][j] >= 1 then {
          covered = 1
          break \\ for all j
      } }
      if covered == 0 then {
        M[l][i] = 0
        break \\ for all cells
  } } }
end
```

Since the bidegree of the basis is $\mathbf{p}$, the support of the function contains at most $(p+1)^2$ cells of level ℓ. The algorithm can be accelerated by considering only passive functions in the vicinity of functions that changed their status from passive to active since the last call of `CLOSE` for this level. In order to keep the presentation simple, we do not give further details.

The algorithm `NEST` maintains the second index set constraint (ii), i.e., the domain hierarchy. Any passive function of level ℓ that refines only to *selected* functions of level $\ell+1$ (according to the relation $\Subset$) changes its status to *refined*.

The cleaning is performed by the third algorithm `CLEAN`. Dealing with the proper selection of refined functions (iii), it is quite similar to `NEST`. Any active function of level ℓ that refines only to *selected* functions of level $\ell+1$ changes its status to *refined*.

These two algorithms are obtained by choosing either the three lines marked by † or by ‡ below:

```
†algorithm NEST(*mat M, int l)
‡algorithm CLEAN(*mat M, int l)
  \\ M is the list of characteristic matrices, l is the level
  †for all i with M[l][i] == 0 do {
  ‡for all i with M[l][i] == 1 do {
    M[l][i] = 2
    for all j with (i,l) ⩽ (j,l+1) do {
      if M[l+1][j] == 0 then {
        †M[l][i] = 0
        ‡M[l][i] = 1
        break \\  for all j
  } } }
end
```

The number of B–splines of level $\ell+1$ that a function refines to (according to the relation $\leqslant$) depends on the knot configuration. For instance, in the case of single knots and dyadic refinement of the two knot vectors, we have to consider $(p+1)^2$ B–splines of level $\ell+1$. Again, both algorithms can be significantly accelerated by considering only functions in the vicinity of B–splines of level $\ell+1$ that changed their status from passive to selected after their last call for this level.

4. Numerical examples

Our tests were performed on several synthetic data sets — created by uniform sampling of analytical functions — and on more challenging industrial data sets related to turbine blade parts (see Figure 1-2 and Section 5). All tests were initialized with a THB–spline basis of size 8×8 and bi-degree (3,3), resulting in curvature continuous surfaces as the standard case in demanding engineering applications.

The THB–splines and the adaptive fitting algorithm were implemented in `C++`. The presented tests were executed on PC running SUSE Linux Enterprise Desktop 11 (Intel XEON E31240 3.30 GHz, 16 GB RAM, 64 bit).

4.1. Comparison of the refinement strategies

The convergence speed and the resulting domain structure of the THB–spline fitting procedure strongly depends on the refinement strategy considered in the hierarchical approximation framework. In Section 3 we proposed two different approaches to identify the regions with higher errors: the absolute (AT) and the relative threshold (RT).

In Table 1 we compare the number of iterations and degrees of freedom of the absolute (AT) and the relative threshold (RT) refinement strategies for the *3 peak data set*[2] which was already considered in [12], see Figure 6). The data are contained in an axis–aligned box of size $2 \times 2 \times 0.65$. We terminate the refinement procedure when at least 99% of the errors were below the threshold $\sigma = 10^{-6}$ — thereby even exceeding the standard industrial precision requirements mentioned in the previous section — or when the number N of iterations reached 10. By setting the regularization parameter to $\lambda = 10^{-9}$, we obtain the number of iterations and degrees of freedom reported in Table 1. We may observe that, in general, even if both refinement strategies can achieve the same level of accuracy, the AT strategy requires less iterations. Furthermore, the choice of the optimal refinement percentage in case of the RT strategy is not easy and may depend both on the shape and the distribution of the data set.

strategy	# iterations	degrees of freedom	percentage	maximum error
AT	5	5,637	99.94	2.55e-06
RT (1 %)	10	1,981	32.16	5.15e-05
RT (5 %)	8	6,434	99.94	2.24e-06
RT (10 %)	6	5,053	99.94	2.25e-06
RT (20 %)	5	5,569	99.94	2.55e-06

Table 1: Number of iterations and degrees of freedom associated to the absolute (AT) and relative (RT) threshold strategies for the 3 peak data set. The last column specifies the percentage of the number of points that satisfy the error threshold $\sigma = 10^{-6}$.

In addition, as shown in Figure 6, the RT strategy tends to create a more fractal–like domain structure. This fact has negative influence on the complexity of the final surface and the required number of tensor product B–spline patches needed to perform an exact export of THB–splines into a standard CAD format, as described later in Section 5.2. Also, the RT strategy does not lead to a substantially smaller number of control points.

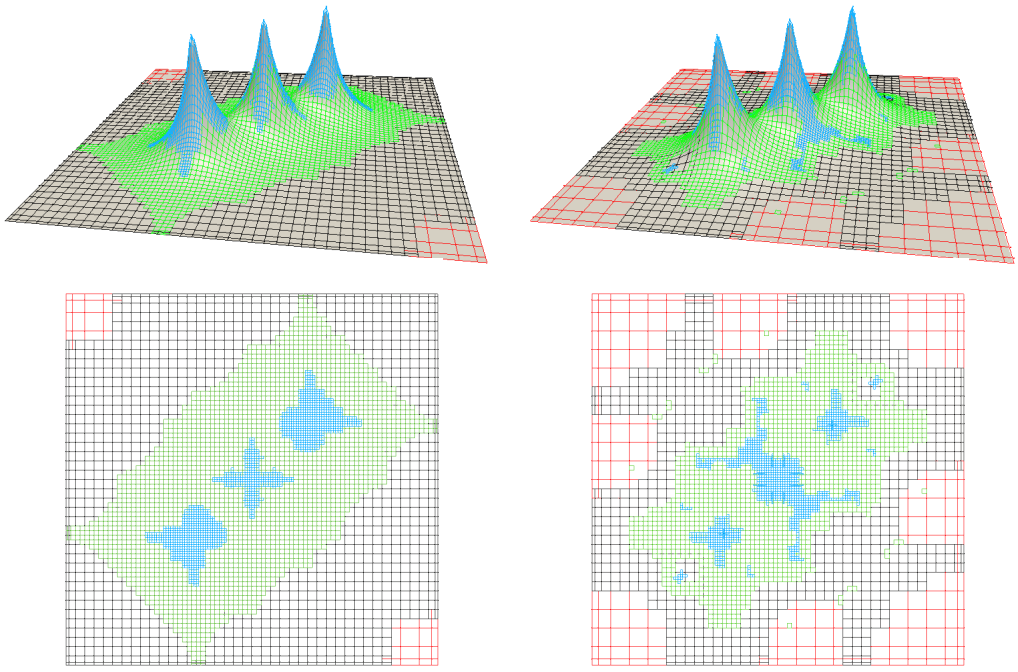
The RT strategy has some advantages if one tries to find the optimal THB–spline surface for a given number of levels. In this situation, one would use a rather small tolerance, and the RT strategy concentrates on the refinement in the regions where the largest errors occur. In contrast, the AT refinement would quickly produce a large number of control points.

However, in all practical test cases with a given error threshold, AT performed better than the RT strategy. Consequently, we will only consider the absolute threshold refinement strategy in the remaining examples.

4.2. Influence of the regularization term

The regularization parameter assures that the solved system of equations is not singular even for high refine-

[2]The data is computed by uniform sampling points of the function $f(x,y) = 1.5(\sqrt{(10x-3)^2+(10y-3)^2})^{-1} + 1.5(\sqrt{(10x+3)^2+(10y+3)^2})^{-1} + 1.5(\sqrt{(10x)^2+(10y)^2})^{-1}$ for $(x,y) \in [-1,1]^2$.



(a) absolute threshold (b) relative threshold

Figure 6: THB–spline approximation with corresponding control mesh for the 3 peak data set. The absolute threshold approach (a) generates a less fractal–like domain structures in comparison to the relative threshold strategy (RT 10%) (b). The top views of the control grids are also shown.

ment levels. At the same time it smooths the resulting surface in case of noisy input data. Naturally, in case of data sets with sharp features, the smoothing effect is in contradiction to the required accuracy of the approximation. For these reasons, the choice of λ is essential for generating accurate results and at the same time minimizing the number of required iterations.

The dependency of the approximation error and the condition number of the matrix on the regularization parameter λ is discussed in Table 2. Our tests show that in the two analyzed[3] cases the regularization parameter should be smaller then 10^{-8} to reach the required accuracy (more than %99 data points with an error below the threshold $\sigma = 10^{-6}$). For larger values of λ the smoothing effect prevents the solution from capturing the fine details of the original surface even for high refinement levels. In general, the choice of $\lambda = 10^{-9}$ provided a satisfactory behavior for all tested data sets. These include the industrial examples shown in Section 5.

(a) 3 peak data set

λ	degrees of freedom	%	condition number	objective function	maximum error
10^{-7}	7,077	98.80	2.70e+09	2.70e-03	2.22e-04
10^{-8}	5,587	99.58	7.48e+10	3.90e-04	2.54e-05
10^{-9}	5,629	99.94	7.20e+11	1.26e-04	2.55e-06
10^{-10}	5,629	100	6.27e+12	9.93e-05	2.55e-07

(b) Rvachev data set

λ	degrees of freedom	%	condition number	objective function	maximum error
10^{-7}	8,501	95.00	2.41e+09	2.39e-02	1.17e-04
10^{-8}	8,959	97.02	8.97e+09	2.57e-03	1.24e-05
10^{-9}	8,833	99.02	2.07e+10	2.59e-04	1.25e-06
10^{-10}	8,759	100	1.94e+11	2.59e-05	1.25e-07

Table 2: The number of degrees of freedom, the condition number and the value of the objective function obtained with different values of the regularization parameters λ for the 3 peak (a) and the Rvachev (b) data set shown in Figure 7. The third column specifies the percentage of points that satisfy the error threshold $\sigma = 10^{-6}$.

4.3. Impact of the extension parameter

The choice of the extension parameter (used in the refinement) may influence the accuracy of the approximation as well as the size of the corresponding THB–spline representation. By enlarging this parameter, the number of newly introduced degrees of freedom increases, thereby providing more flexibility in the neighborhood of the area with high error.

Table 3 compares the number of degrees of freedom and

[3]In addition to the 3 peak data set, previously introduced, we consider the *Rvachev data set*, computed by uniform sampling of the function $f(x,y) = \frac{(x+y)}{2} + \sqrt{\left(\frac{x-y}{2}\right)^2}$ for $[x,y] \in [0,1]^2$.

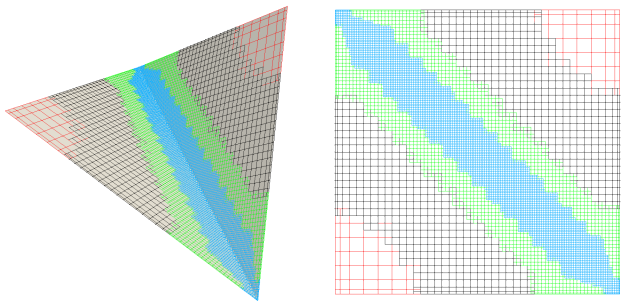


Figure 7: Approximation of the Rvachev function with THB–splines (left) with regularization parameter $\lambda = 10^{-10}$. The top view of the corresponding control mesh is also shown (right).

the approximation errors for different values of the extension parameter, again for the 3 peak data set. As expected, the optimal value of this parameter is $\lceil \frac{p}{2} \rceil$. This corresponds to the smallest possible extension that is required to add at least one new basis function to the THB–spline basis at the next refinement level. For larger values of the extension parameter, we may observe that an increase of the number of coefficients does not always corresponds to significant improvements with respect to the accuracy of the approximation.

extension	# iterations	degrees of freedom	%	objective function	maximum error
2	5	5,629	99.94	1.26e-04	2.55e-06
3	5	6,170	99.94	9.96e-05	2.55e-06
4	5	6,841	99.94	7.08e-05	2.55e-06
5	5	7,396	99.94	5.93e-05	2.55e-06

Table 3: Number of iterations, degrees of freedom and related value for the objective function with respect to different extension parameters obtained by sampling 10^4 data points from the 3 peak data set. The regularization parameter is set to $\lambda = 10^{-9}$ and the required error threshold to $\sigma = 10^{-6}$.

4.4. Global vs. local refinement

The accurate modeling and reconstruction of surfaces with sharp features and small details cause many difficulties for current CAD software. Furthermore, the enormous size of the resulting geometries has a negative influence on all post–processing steps. On the other hand the local refinement of THB–splines provides an effective tool for constructing the detailed structures, while simultaneously minimizing the size of the resulting geometry.

Table 4 and Figure 8 compare the number of degrees of freedom for different number of iterations of the fitting procedure for the Rvachev data set. The growth of the size of the standard B–spline basis is exponential. Thus, the global refinement method reaches its limits after several refinement steps and the solving of the underlying system of equations becomes too expensive in terms of computational time and memory. In contrast to this, the growth of the THB–spline basis remains fairly moderate.

	degrees of freedom		objective function		maximum error	
#	local	global	local	global	local	global
1	169	169	8.69e+00	8.69e+00	1.28e-02	1.28e-02
2	529	529	2.53e+00	2.53+00	6.36e-03	6.36e-03
3	1,729	1,849	7.97e-01	7.97e-01	2.97e-03	2.97e-03
4	4,147	6,889	2.42e-01	2.42e-01	1.02e-03	1.02e-03
5	8,841	26,569	2.59e-04	2.59e-04	1.26e-06	1.26e-06
6	9,233	104,329	2.51e-04	2.36e-04	1.19e-06	1.15e-06
7	9,625	413,449	2.44e-04	2.25e-04	1.16e-06	1.10e-06
8	10,017	n/a	2.43e-04	n/a	1.15e-06	n/a

Table 4: The size of the THB–spline basis remains moderate even for high refinement levels (first column), while it grows much faster for tensor-product B–splines (Rvachev data set).

4.5. Computing times

Each iteration of the presented adaptive fitting procedure consists of three steps:

1. the refinement of the THB–spline basis, i.e., adaptation and maintenance of the sets of active functions;
2. the assembly of the linear system;
3. the solution of the linear system[4].

In our current experimental implementation, the total computing time is dominated by the first two steps. To give an idea, in order to generate the THB–spline approximation of the fillet data set with 38,260 points (Figure 1), where the number of degrees of freedom varies between 169 for 1 level and 20,553 for 7 levels after several iterations, the first two steps need a few seconds up to a few minutes, while the solver time is always less than a second.

We are currently exploring various possibilities to speed up these computations. For example, the B–spline representations of THB–splines is precomputed in order to speed up their evaluation. More precisely, each THB–spline is then represented by its B–spline coefficients at the coarsest possible level. This precomputation is performed by slightly modifying the THB–spline evaluation algorithm. Obviously, a natural way to further optimize the performance for data sets of high complexity relies on parallel computing techniques.

5. CAD integration of THB–splines

The process of converting a measured data set into a CAD object — often referred to as *geometry acquisition*

[4]We used the biconjugate gradient stabilized method solver (BiCGSTAB) from the Eigen library, eigen.tuxfamily.org

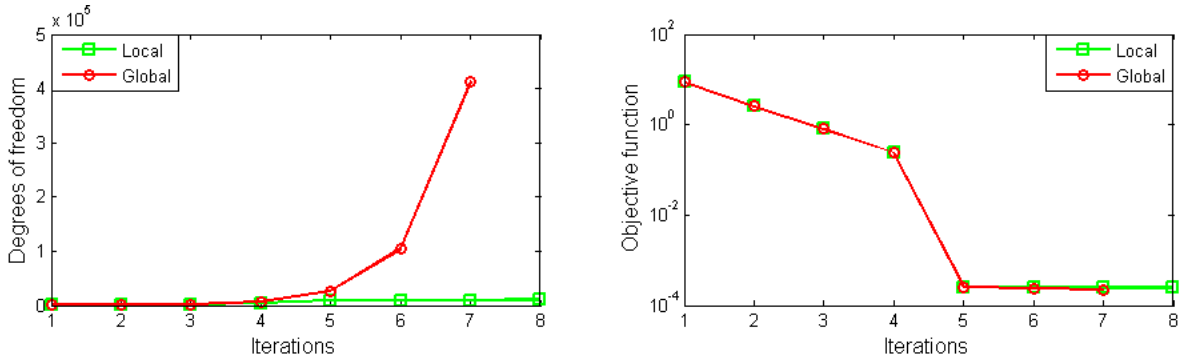


Figure 8: While the size of the globally refined tensor–product basis grows exponentially and the growth of the size of the locally refined basis decreases (left), the difference between the resulting values of the objective function is not significant (right).

and reconstruction — is a crucial part in certain industrial applications, for example, in analyzing manufacturing tolerances with respect to their aerodynamic and structural mechanical impact. The introduced adaptive fitting framework with THB–splines improves the required number of degrees of freedom and the overall stability of the reconstruction process while maintaining the accuracy of the standard fitting technique.

This leads to a dramatic performance enhancement in related industrial applications. This is illustrated in Section 5.1 where the reconstruction process related to a crucial part of an aircraft turbine blade is illustrated. Subsequently, Section 5.2 presents the export of THB–spline geometries into standard tensor–product B–splines, which provides a useful tool for integrating hierarchical spline representations into standard CAD software.

5.1. Adaptive geometry reconstruction

Industrial data sets are often generated by an optical measurement system producing a large amount of non–uniformly distributed points describing the shape of the object. A non–uniform sampling and a strongly varying shape of the data causes several problems during the fitting procedure with standard techniques. This limitation of the tensor–product structure can be eliminated by using the adaptive THB–splines fitting framework.

Therefore, we investigated the reconstruction of the fillet part of a turbine blade as one of the challenging geometrical parts of an aero engine. Figure 4 shows the used point cloud which was parametrized by the technique described in [7]. The fully automatically reconstructed fillet geometry by using B–splines and THB–splines is shown in Figure 9. As the noisy reflection lines show, the tensor–product spline surface suffers from strong oscillations on its upper part, whereas the same region on the hierarchical spline surface is perfectly smooth. This results from the fact that, within the standard technique, no optimal number of degrees of freedom exists to avoid oscillations in the upper fillet part while generating an accurate fitting geometry on the lower fillet part. Note that the problem of oscillations may not be solved with standard B–spline surfaces. Higher values of the regularization parameter does not lead to an effective solution: even if the amplitude and frequence of the oscillations may decrease, the unwanted oscillations will be present until the surface becomes planar, without obtaining the desired smooth curved surface. For this reason, an adaptive scheme is needed to compute a flexible and accurate fitting by exploiting the possibility of identifying different levels of resolution. This demonstrates the superior behavior of the adaptive fitting framework with truncated hierarchical geometries in comparison with the use of standard tensor–product geometries.

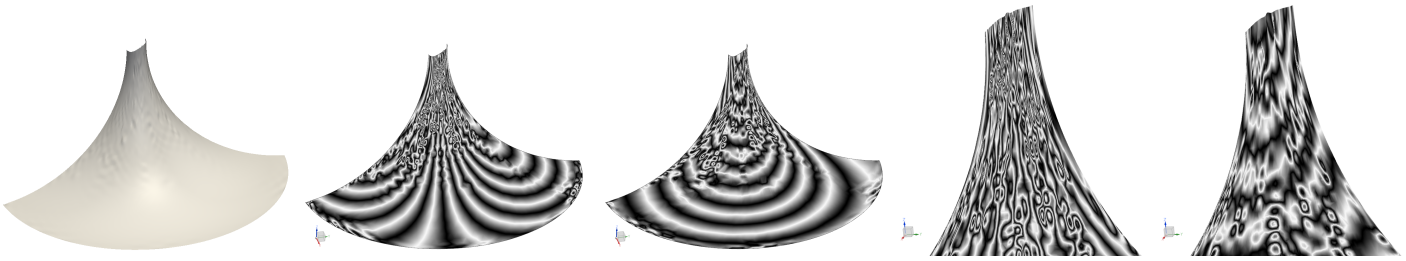
5.2. Conversion to tensor–product patches

Despite the advantages of THB–splines, the current CAD standard relies on tensor–product B–splines or, more generally, on tensor-product NURBS. Therefore, a procedure for exporting THB–spline geometries into the standard format is needed.

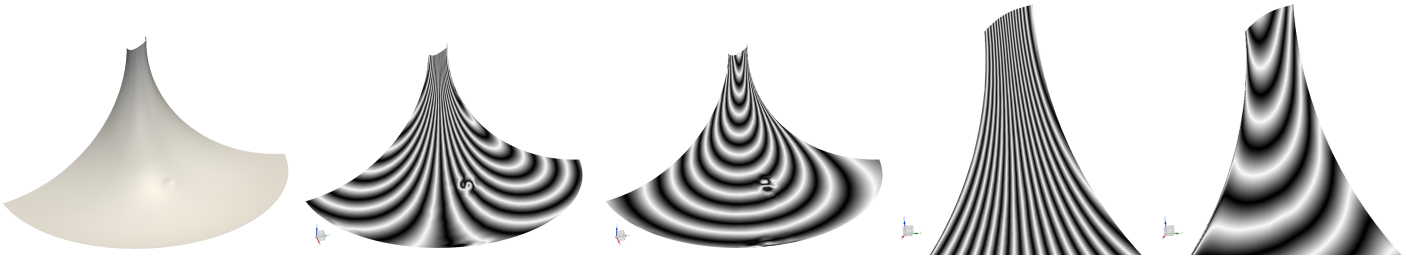
The hierarchical construction of THB–splines offers a natural way to perform this operation by computing the coefficients of the tensor–product representation directly from the control points of the original surface. To execute this conversion efficiently, we need to split the original geometry into several B–spline patches according to the different refinement levels of the THB–spline representation.

The algorithm `EXPORT` performs the conversion of a THB–spline geometry into several B–spline patches using certain splitting techniques, which are described later. The evaluation function `EVALUATE_THB_SURFACE` described in Section 2 can be modified to suit this operation by terminating the iteration at the level of interest. This modified evaluation is indicated as `EVALUATE_THB_SURFACE*` in the algorithm below, where we assume $\Omega^{L+1} = \emptyset$.

```
algorithm EXPORT(mat c, int L)
  \\ c are the THB–spline coefficients
  \\ L is the number of levels
  for l from 0 to L do {
    create the ring r = Ω^l \ Ω^(l+1)
    create the list R of connected components of r
    for i from 1 to |R| do {
```



(a) global refinement using B–splines and 1849 degrees of freedom



(b) local refinement using THB–splines and 475 degrees of freedom

Figure 9: Reconstruction of the fillet part with globally refined B–splines (top row) / locally refined THB–splines (bottom row) using 1849 / 475 degrees of freedom. In both cases the regularization parameter and the error threshold are set to $\lambda = 10^{-9}$ and $\sigma = 10^{-6}$, respectively. The quality of the surfaces (leftmost plots) is visualized using reflection lines. The two rightmost plots in both rows show an enlarged view of the upper part of the fillet. Note that the small distortion of these lines in the central part of the fillet is caused by a measurement error in the provided data.

```
    boxes = SPLIT_TO_BOXES(R[i])
    for j from 1 to |boxes| do {
      Compute the coefficients of the restriction of the
      THB–spline surface to the box boxes[j] using
      EVALUATE_{T}HB_SURFACE* and export the obtained
      tensor–product B–spline surface.
  } } }
end
```

This algorithm relies on the `SPLIT_TO_BOXES` procedure that splits a connected component of one refinement level according to one of the following methodologies.

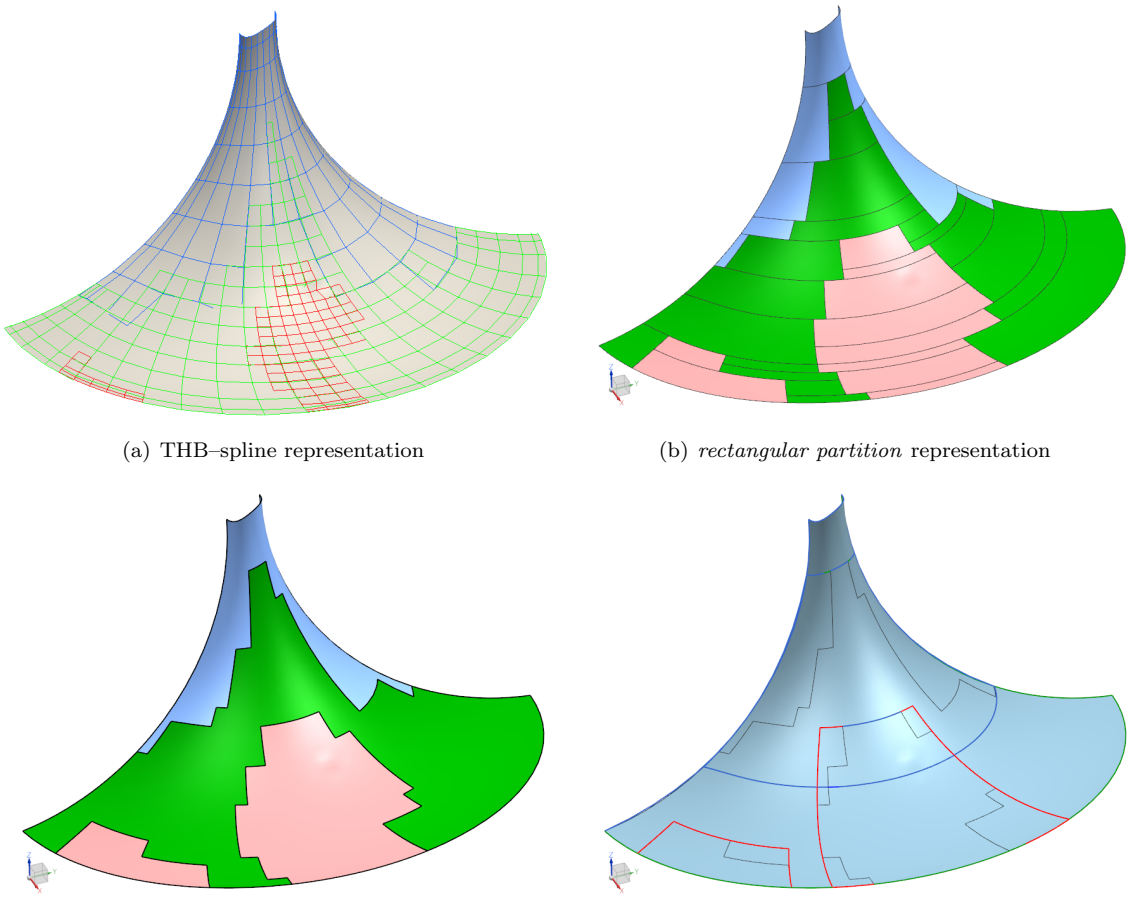
1. *Smallest bounding box*: by using the B–spline representation of the smallest axis aligned bounding box covering the connected component `R[i]`. Its boundary curve is defined in the parameter domain of `R[i]` and used for trimming the tensor–product surface later.
2. *Rectangular partition*: by using the B–spline representation of rectangular boxes completing the connected components `R[i]`.

The first method leads to a small number of patches but increases the required memory storage, since one needs to store additional "phantom" control points in the trimmed areas and the trimming curves themselves. The second approach leads to an increased number of patches depending on the shape of the hierarchical domains. However, in this case the number of degrees of freedom is only increased by storing multiple copies of common control points between adjacent patches.

The domain structure of the THB–splines can be represented by a quadtree data structure which provides us naturally with the collection of boxes, necessary for the rectangular partition procedure, in its leaves. See [15] for a detailed information on this concept. To optimize the number of patches, we applied an additional step where adjacent areas of the same refinement level are joined together creating larger rectangular boxes. The list of boxes stored by the quadtree allows also to compute the smallest rectangular box that covers the connected components of $\Omega^\ell \setminus \Omega^{\ell+1}$, for $\ell = 0, \dots, L$, required for the smallest bounding box procedure.

Finally, we use the capabilities of a CAD system, i.e., Parasolid™ by SIEMENS PLM Software [19], to bridge the gap between THB–splines and standard (commercial) applications. The geometric modeling kernel combines the geometric representations with a topological structure to handle trimmed entities and to build up large complex models based on several (connected) geometries. Therefore, Parasolid combines the generated B–spline patches into a single topological object referred to as a *sheet* without using any approximation, see Figure 10. This provides a straight–forward and geometrically exact integration of THB–splines into standard industrial processes and applications.

Figure 10 presents an example. The THB–spline surface representing the fillet (a) with 475 control points can be split either into 36 patches with 1788 control points (b) or into 4 trimmed patches with 1021 control points (c,d).



(a) THB–spline representation

(b) *rectangular partition* representation

(c) *smallest bounding box* representation

(d) bounding boxes and trimming curves for (c)

Figure 10: Reconstructed fillet part geometry: the approximated THB–spline surface is defined by three refinement levels with a corresponding control grid of 475 control points (a). An optimal *rectangular partition* split into standard B–splines surfaces requires 36 patches with 1788 control points in total (b). The *smallest bounding box* split generates four B–spline patches with 1021 control points in total (c) and the corresponding patch layout is shown in (d). The coloring of patches corresponds to different refinement levels.

6. Conclusion

In this paper we explored the potential benefits of using the recently introduced truncated hierarchical B–splines (THB–splines) in a least–squares approximation framework, focusing on their successful integration with the current mathematical technology of Computer Aided Design. After recalling the definition, the related algorithms and the properties of THB–splines, we presented an adaptive surface fitting framework and we identified two possible strategies for performing the refinement. The presentation also covered implementation aspects and a simple regularization technique.

Based on this approximation framework, we showed how to use THB–splines for demanding real–world applications, in particular, for reconstructing the geometry of core components of aero engines. When compared to the existing tensor–product spline technology, the use of THB–splines significantly improves the quality of the resulting geometric shapes. Moreover, based on standard features of the geometric kernels of CAD systems, such as Parasolid™, we identified two possibilities that allow to export the resulting surfaces as standard CAD geometries/models.

In order to further exploit the successful CAD interface of the THB–spline framework here presented, future work will be devoted to algorithm and efficiency optimization of the related interconnected procedures. This will pave the way for using this new spline technology in further applications, e.g., for performing numerical simulations using the approach of isogeometric analysis. In particular, the use of adaptive generalizations of tensor–product splines (such as THB–splines) becomes mandatory when addressing three–dimensional problems, since the "curse of dimension" makes global refinement even more prohibitive. The truncation mechanism may also provide the possibility — non-trivial in an adaptive/hierarchical spline context — of developing refinement procedures that, by preserving suitable hierarchical domain configurations, may allow to consider explicit bounds for the number of basis functions acting on a single mesh element.

Additionally we have started to explore generalizations of the THB–spline framework to more general spaces of functions, which include box splines and spaces related to subdivision surfaces [33]. Also, recent results regarding the completeness of hierarchical spline spaces allow to generate simple B–spline bases for large classes of spline spaces over box partitions in any dimensions [18]. By relying on sparse representations of geometric objects and of functions defined on them, these spaces have the great potential of providing a viable multivariate approach for demanding applications.

Acknowledgments

The authors were supported by the projects EXAMPLE and INSIST (EC GA nos. 324340, 289361), "Geometry + Simulation" (FWF NFN S117) and DREAMS (MIUR "Futuro in Ricerca" RBFR13FBI3).

References

[1] A. Bressan. Some properties of LR-splines. *Comput. Aided Geom. Design*, 30:778–794, 2013.

[2] J. A. Cottrell, T. J. R. Hughes, and Y. Bazilevs. *Isogeometric Analysis: Toward Integration of CAD and FEA*. John Wiley & Sons, 2009.

[3] J. Deng, F. Chen, and Y. Feng. Dimensions of spline spaces over T–meshes. *J. Comput. Appl. Math.*, 194:267–283, 2006.

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

[5] T. Dokken, T. Lyche, and K. F. Pettersen. Polynomial splines over locally refined box–partitions. *Comput. Aided Geom. Design*, 30:331–356, 2013.

[6] M. Floater and K. Hormann. Surface parameterization: a tutorial and survey. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, Mathematics and Visualization, pages 157–186. Springer, Berlin, Heidelberg, 2005.

[7] M. S. Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3):231–250, 1997.

[8] D. R. Forsey and R. H. Bartels. Hierarchical B-spline refinement. *Comput. Graphics*, 22:205–212, 1988.

[9] D. R. Forsey and R. H. Bartels. Surface fitting with hierarchical splines. *ACM Trans. Graphics*, 14:134–161, 1995.

[10] D. R. Forsey and D. Wong. Multiresolution surface reconstruction for hierarchical B-splines. In W. A. Davis, K. S. Booth, and A. Fournier, editors, *Graphics Interface*, pages 57–64. Canadian Human-Computer Communications Society, 1998.

[11] C. Giannelli and B. Jüttler. Bases and dimensions of bivariate hierarchical tensor–product splines. *J. Comput. Appl. Math.*, 239:162–178, 2013.

[12] C. Giannelli, B. Jüttler, and H. Speleers. THB–splines: the truncated basis for hierarchical splines. *Comput. Aided Geom. Design*, 29:485–498, 2012.

[13] C. Giannelli, B. Jüttler, and H. Speleers. Strongly stable bases for adaptively refined multilevel spline spaces, preprint. *Adv. Comput. Math.*, 2013. DOI 10.1007/s10444-013-9315-2.

[14] G. Greiner and K. Hormann. Interpolating and approximating scattered 3D-data with hierarchical tensor product B-splines. In A. L. Méhauté, C. Rabut, and L. L. Schumaker, editors, *Surface Fitting and Multiresolution Methods*, pages 163–172. Vanderbilt University Press, Nashville, TN, 1997.

[15] G. Kiss, C. Giannelli, and B. Jüttler. Algorithms and data structures for truncated hierarchical B-splines. In M. Floater et al., editors, *Mathematical Methods for Curves and Surfaces*, volume 8177 of *Lecture Notes in Computer Science*, pages 304–323. Springer, 2014.

[16] R. Kraft. Adaptive and linearly independent multilevel B–splines. In A. Le Méhauté, C. Rabut, and L. L. Schumaker, editors, *Surface Fitting and Multiresolution Methods*, pages 209–218. Vanderbilt University Press, Nashville, 1997.

[17] X. Li, J. Deng, and F. Chen. Polynomial splines over general T-meshes. *Visual Comput.*, 26(4):277–286, 2010.

[18] D. Mokriš, B. Jüttler, and C. Giannelli. On the completeness of hierarchical tensor-product B-splines. Technical Report 8, NFN Geometry + Simulation, 2013. Available at www.gs.jku.at.

[19] Parasolid 3D geometric modeling engine. Siemens PLM Software Inc., 2013. www.plm.automation.siemens.com/en_us/products/open/parasolid.

[20] D. Schillinger, L. Dedé, M. A. Scott, J. A. Evans, M. J. Borden, E. Rank, and T. J. R. Hughes. An isogeometric design-through-analysis methodology based on adaptive hierarchical refinement of NURBS, immersed boundary methods, and T-spline CAD surfaces. *Comput. Methods Appl. Mech. Engrg.*, 249:116 – 150, 2012.

[21] M. A. Scott, X. Li, T. W. Sederberg, and T. J. R. Hughes. Local refinement of analysis-suitable T-splines. *Computer Methods in Applied Mechanics and Engineering*, 213–216:206–222, 2012.

[22] M. A. Scott, D. C. Thomas, and E. J. Evans. Isogeometric spline forests. *Comput. Methods Appl. Mech. Engrg.*, 269:222–264, 2014.

[23] T. W. Sederberg, D. L. Cardon, G. T. Finnigan, N. S. North, J. Zheng, and T. Lyche. T-spline simplification and local refinement. *ACM Trans. Graphics*, 23:276 – 283, 2004.

[24] T. W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-splines and T-NURCCS. *ACM Trans. Graphics*, 22:477–484, 2003.

[25] H. Speleers and C. Manni. Effortless quasi-interpolation in hierarchical spaces. Preprint, 2013.

[26] A.-V. Vuong, C. Giannelli, B. Jüttler, and B. Simeon. A hierarchical approach to adaptive local refinement in isogeometric analysis. *Comput. Methods Appl. Mech. Engrg.*, 200:3554–3567, 2011.

[27] J. Wang, Z. Yang, L. Jin, J. Deng, and F. Chen. Parallel and adaptive surface reconstruction based on implicit PHT-splines. *Comput. Aided Geom. Design*, 28:463–474, 2011.

[28] W. Wang, Y. Zhang, M. Scott, and T. Hughes. Converting an unstructured quadrilateral mesh to a standard T-spline surface. *Computational Mechanics*, 48:477–498, 2011.

[29] W. Wang, Y. Zhang, G. Xu, and T. Hughes. Converting an unstructured quadrilateral/hexahedral mesh to a rational T-spline. *Computational Mechanics*, 50:65–84, 2012.

[30] Y. Wang and J. Zheng. Curvature-guided adaptive T-spline surface fitting. *Comput. Aided Design*, pages 1095–1107, 2013.

[31] V. Weiss, L. Andor, G. Renner, and T. Varády. Advanced surface fitting techniques. *Comput. Aided Geom. Design*, 19:19–42, 2002.

[32] Y. Zhang, W. Wang, and T. Hughes. Conformal solid T-spline construction from boundary T-spline representations. *Comput. Mech.*, 51:1051–1059, 2013.

[33] U. Zore and B. Jüttler. Generalized hierarchical spline spaces. Technical Report 9, NFN Geometry + Simulation, 2013. Available at www.gs.jku.at.