Efficient alternating and joint distance minimization methods for adaptive spline surface fitting

Carlotta Giannelli, Sofia Imperatore, Angelos Mantzaflaris, Dominik Mokriš

Efficient alternating and joint distance minimization methods for adaptive spline surface fitting

Carlotta Giannelli[a], Sofia Imperatore[a,*], Angelos Mantzaflaris[b], Dominik Mokriš[c]

[a] *Dipartimento di Matematica e Informatica "Ulisse Dini", Università degli Studi di Firenze, Viale Morgani 67/A, Firenze, 50137, Italy*
[b] *Inria Centre at Université Côte d'Azur, 2004 route des Lucioles - BP 93, Sophia Antipolis, 06902, France*
[c] *MTU Aero Engines AG, Dachauer Strasse 665, Munich, 80995, Germany*

Abstract

We propose a new paradigm for scattered data fitting with adaptive spline constructions based on the key interplay between parameterization and adaptivity. Specifically, we introduce two novel adaptive fitting schemes that combine moving parameterizations with adaptive spline refinement, for highly accurate CAD models reconstruction from real-world scattered point clouds. The first scheme alternates surface fitting and data parameter optimization. The second scheme jointly optimizes the parameters and the surface control points. To combine the proposed fitting methods with adaptive spline constructions, we present a key treatment of boundary points. Industrial examples show that updating the parameterization, within an adaptive spline approximation framework, significantly reduces the number of degrees of freedom needed for a certain accuracy, especially if spline adaptivity is driven by suitably graded hierarchical meshes. The numerical experiments employ THB-splines, thus exploiting the existing CAD integration within the considered industrial setting, nevertheless, any adaptive spline construction can be chosen.

Keywords: Adaptive surface fitting, Parameter correction, L-BFGS, THB-splines

1. Introduction

We consider the reverse engineering problem of reconstructing highly accurate Computer-Aided Design (CAD) models, mostly of aircraft engine components. Note that the considered data, due to the acquisition process, i. e. through optical scans, are scattered and affected by noise. Nevertheless, the reconstructed geometric models are required to capture meaningful geometric features of the engine parts while being smooth and compact. In a parametric surface fitting algorithm we have two groups of unknowns that may be adjusted when constructing a surface: the spline control points and the point parameters. In this work we propose non-linear approximation schemes for each of these two groups of unknowns, enhanced with local control point refinement and iterative parameter updating by means of hierarchical splines. The ultimate goal of this article consists of developing automatic, highly-accurate, and efficient methods for the representations of the 3D input discrete data via an adaptive spline surface model.

Splines are piecewise polynomial functions that feature a certain global regularity. The spline representation most commonly used in CAD software libraries relies on B-splines [12] and their non-uniform rational extension, i. e. NURBS [44]. Until recently, the multivariate case for standard CAD spline representations relied on the *tensor-product* structure. This construction poses several limitations when one wants to properly include *local* properties in the geometric model. In particular, tensor-product constructions allow only *global* refinement. Hence, unnecessary degrees of freedom may be added in regions far away from the one of interest. The need to overcome their limits and achieve enhanced flexibility and efficient computation of high-quality approximations led to the development of new *adaptive* spline spaces defined on T-meshes, which are generalizations of tensor-product constructions allowing T-junctions and hence enabling local refinement. Spline adaptivity in this setting can be achieved with

*Corresponding author
Email addresses: carlotta.giannelli@unifi.it (Carlotta Giannelli), sofia.imperatore@unifi.it (Sofia Imperatore), angelos.mantzaflaris@inria.fr (Angelos Mantzaflaris), dominik.mokris@mtu.de (Dominik Mokriš)

different approaches, and several proposals can be found in the literature. Among others, we mention: T-splines [51, 50], and related modified versions, see e.g., [33, 55]; PHT-splines [13]; LR B-splines [14]. LR B-splines have also been employed for GIS terrain and bathymetry data surface reconstruction [28]. Finally, Hierarchical B-splines (HB-splines) [17, 22, 30] are multilevel B-spline extensions that preserve the tensor-product structure at any level of the hierarchy. Their truncated formulation, i. e. Truncated Hierarchical B-splines (THB-splines) [21], is based on HB-splines and relies on the definition of a certain truncation operator. THB-splines are a more flexible tool than HB-splines since they reproduce some characteristic properties of tensor-product B-splines, such as non-negativity and partition of unity. Moreover, smaller supports reduce basis function overlapping and improve numerical conditioning, see, e.g., [21, 20] and influence also admissible mesh grading, see [7].

The main focus of this paper is to analyze the interplay between (scattered) data *parameterization* and spline *adaptivity*. In particular, we consider THB-spline constructions, which represent a desirable tool for building flexible geometric models, because of their local tensor-product structure, their key properties, and their ease of implementation, see again [21, 20, 7]. Moreover, THB-splines are also the mathematical tools used to perform the CAD integration into Parasolid [1] within the industrial environment of this paper. We refer to [29] for a detailed discussion of this software interface in the considered industrial setting. Nevertheless, we remark that any other different adaptive spline construction might also have been chosen in the framework of our enhanced data fitting schemes.

The THB-spline fitting problem can be stated as follows. Given a scattered data set of the form

$$\mathcal{P} = \{\boldsymbol{p}_i \in \mathbb{R}^D \,|\, i = 1, \dots, m\}, \tag{1}$$

and an error tolerance $\epsilon > 0$, find a THB-spline $\boldsymbol{s} : \Omega \subseteq \mathbb{R}^N \to \mathbb{R}^D$ and the parameter values $\boldsymbol{u}_i \in \Omega, i = 1, \dots, m$, so that

$$\operatorname{dist}\left(\boldsymbol{s}(\boldsymbol{u}_i), \boldsymbol{p}_i\right) \leq \epsilon \quad \text{for each } i = 1, \dots, m. \tag{2}$$

Solving (2) implies the solution of two essential problems:

(a) Find the data parameterization, i. e. assign a parameter value $\boldsymbol{u}_i \in \Omega \subset \mathbb{R}^N$ to each point $\boldsymbol{p}_i \in \mathbb{R}^D$, for $i = 1, \dots, m$, and define the set

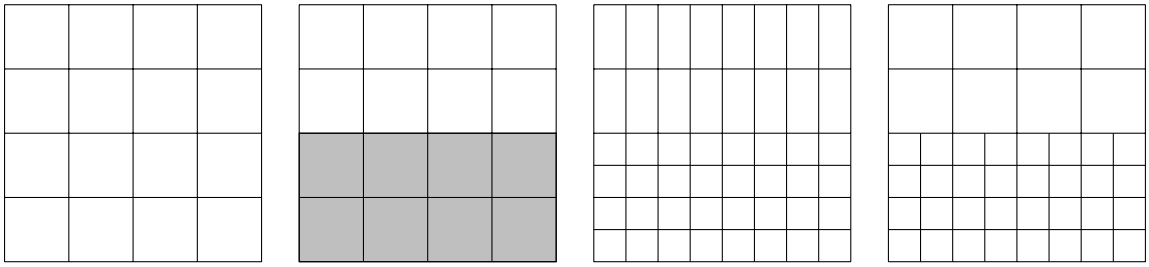
$$\mathcal{U} := \{\boldsymbol{u}_i \in \Omega \subset \mathbb{R}^N \,|\, i = 1, \dots, m\}. \tag{3}$$

(b) Construct the THB-spline approximant in a suitable hierarchical spline space.

The straightforward approach consists of first generating the parameter values, i. e. solving (a), and then constructing the spline approximant in (b) for those *fixed* values. In a more sophisticated approach, the point parameters may be considered as variables that can be adjusted to optimize the surface. This leads to a non-linear optimization problem. In particular, such approaches can be naturally grouped into two categories: *alternating* and *joint-optimization* methods.

In this article, we propose different solutions to problem (2) for scattered data of industrial complexity by developing novel adaptive THB-spline approximation schemes that address the parameterization problem (a) and the fitting scheme (b), either *separately* or *simultaneously*. More specifically, we extend the *alternating* and *joint* optimization methods developed for B-spline curves and tensor-product B-spline surfaces to the THB-spline setting. Furthermore, we exploit the potential of involving admissible locally refined meshes [9] for fitting problems. To the best of our knowledge, adaptive spline approaches for these schemes are not fully addressed in the literature, and, moreover, beyond commonly used least squares fitting, surprisingly few publications [4, 26] treat the B-spline surface reconstruction problem, using classical tensor-product B-splines.

The current paper builds on top of the work presented in [18], where the potential of constructing an alternating adaptive method with THB-splines has been demonstrated using point distance minimization. In this paper, we extend this approach in different directions considering more effective error metrics, such as tangent [3, Chapter 6.2] and hybrid [4] distance minimization. In addition, we propose an adaptive joint optimization method, which was previously considered only in the B-spline curve [58] and tensor-product B-spline surface [26] cases. A selection of real-data examples shows that the update of the parameterization, within an adaptive spline approximation framework, can lead to a significant reduction of the number of degrees of freedom needed to obtain a certain accuracy. This



(a) initial tensor-product grid (b) marked elements (c) tensor-product refinement (d) hierarchical refinement

Figure 1: Given an initial tensor-product grid (a) and a set of elements marked for refinement (b), the refined tensor-product grid (c) propagates the refinement beyond the marked areas, while the hierarchical model enables local refinement capabilities (d).

advantage can be even more pronounced if hierarchical spline adaptivity driven by suitably graded hierarchical meshes is considered. Finally, to properly combine the alternating and joint distance minimization methods with THB-spline constructions, we present a key treatment of boundary points, while simultaneously addressing the interplay of moving parameterization and THB-spline refinement. It should be noted that the proposed fitting schemes with moving parameterization can be naturally exploited also in combination with different (adaptive) spline constructions such as T-splines or LR B-splines.

The structure of the paper is the following. Section 2 introduces some preliminary notions on THB-spline constructions. The adaptive THB-spline fitting loop is then presented in Section 3 in its general framework. The novel adaptive alternating and joint optimization methods are detailed in Section 4 and 5, respectively. Finally, a selection of numerical examples related to industrial applications is presented in Section 6 and Section 7 concludes the paper with some final remarks.

2. Preliminaries

Tensor-product B-splines and their rational extensions have been largely adopted for geometric modeling applications because of their properties and the simplicity of their construction, see e. g., [15, 44]. The main limitation of the tensor-product model arises when a local refinement strategy has to be considered, since any knot insertion in one parametric direction propagates the refinement over the other directions to generate the refined tensor-product grid. Consequently, strictly localized refinement is precluded in this setting and additional degrees of freedom may be inserted even far away from the area marked for refinement, as shown in Figure 1 (a–c). Hierarchical spline refinement can be considered to address this issue by localizing the refinement area, as in Figure 1 (d). In this Section, we provide a brief introduction to the hierarchical spline model with a focus on truncated hierarchical B-splines [21].

2.1. Hierarchical B-splines

Local spline refinement enables the possibility of achieving flexible and accurate models by strongly reducing the total number of control points if compared with standard tensor-product B-spline representations. Hierarchical B-splines (HB-splines) [17] naturally support this kind of local mesh refinement by introducing tensor-product B-splines on several hierarchical levels. Specifically, a selection mechanism to identify a suitable basis was presented in [30]. We now briefly recall this construction.

Let Ω be a hypercube of $\mathbb{R}^N$ and $\boldsymbol{d} = (d_1, \dots, d_N)$, $\boldsymbol{k} = (k_1, \dots, k_N) \in \mathbb{N}^N$ be the polynomial multi-degree and -order, respectively, and

$$V^0 \subset \dots \subset V^{L-1} \tag{4}$$

be a nested sequence of L tensor-product B-spline spaces defined on $\Omega \subset \mathbb{R}^N$. More precisely, for each level $\ell = 0, \dots, L-1$, let $\Gamma^\ell_{\boldsymbol{k}}$ be the set of indices for the tensor-product B-spline basis of level ℓ, hence

$$\mathcal{B}^\ell = \left\{ \beta^\ell_j \mid j \in \Gamma^\ell_{\boldsymbol{k}} \right\}, \quad \text{and} \quad V^\ell = \operatorname{span} \left\{ \beta^\ell_j \,|\, j \in \Gamma^\ell_{\boldsymbol{k}} \right\}. \tag{5}$$

Furthermore, let G^ℓ be the tensor-product mesh associated to each V^ℓ, and let q be their non-empty quadrilateral elements, commonly addressed as mesh elements of level ℓ, for each $\ell = 0, \dots, L-1$. In

addition to the tensor-product spline spaces V^ℓ and the corresponding tensor-product B-spline bases $\mathcal{B}^\ell$, consider a nested sequence of closed subsets of Ω, i. e.

$$\Omega \equiv \Omega^0 \supset \ldots \supset \Omega^L = \emptyset, \tag{6}$$

where each Ω^ℓ is the union of a selection of elements q of the tensor-product mesh G^ℓ. Thus, each domain boundary $\partial\Omega^\ell$ is aligned with the knot-lines of V^ℓ, for each $\ell = 0, \ldots, L-1$. Define

$$\mathbb{G}^\ell := \left\{ q \in G^\ell \mid q \subset \Omega^\ell \setminus \Omega^{\ell+1} \right\}$$

to be the set of active elements of level $\ell = 0, \ldots, L-1$. Consequently, the hierarchical mesh is defined as

$$\mathcal{M} := \left\{ q \in \mathbb{G}^\ell \mid \ell = 0, \ldots, L-1 \right\}.$$

Finally, given a function f defined on Ω, we address with $\operatorname{supp}(f)$ the intersection of the support of f with Ω.

Given a sequence of tensor-product B-spline spaces and bases as in (4) and (5), as well as a sequence of closed subsets as in (6), the HB-spline basis can be recursively defined by replacing any B-spline of level ℓ with support completely contained in $\Omega^{\ell+1}$ by B-splines at successively finer levels. Specifically, define

$$A^\ell_{\boldsymbol{k}} := \left\{ j \in \Gamma^\ell_{\boldsymbol{k}} \mid \operatorname{supp}\left(\beta^\ell_j\right) \subseteq \Omega^\ell \wedge \operatorname{supp}\left(\beta^\ell_j\right) \not\subseteq \Omega^{\ell+1} \right\}$$

the set of indices of *active functions* at level ℓ. Then the HB-spline basis and space are respectively

$$\mathcal{H}_{\boldsymbol{k}}(\mathcal{M}) := \left\{ \beta^\ell_j \mid j \in A^\ell_{\boldsymbol{k}},\ \ell = 0, \ldots, L-1 \right\}, \text{ and } V := \operatorname{span}\left\{ \mathcal{H}_{\boldsymbol{k}} \right\}. \tag{7}$$

Note that, the elements of the HB-basis are non-negative, have local support, but they *do not* form a partition of unity. Moreover, the HB-basis is characterized by several overlaps of coarse and fine B-spline supports.

2.2. Truncated hierarchical B-splines

Truncated hierarchical B-splines (THB-splines) were introduced in [21] to construct an alternative basis for the hierarchical spline space V in (7) with enhanced properties, i. e. smaller supports, reduced interaction between basis functions at different levels and partition of unity. The construction of THB-splines is based on the so-called truncation mechanism. In particular, let $s \in V^\ell \subset V^{\ell+1}$ be represented in terms of the tensor-product B-spline basis $\mathcal{B}^{\ell+1}$, i. e.

$$s(\boldsymbol{u}) = \sum_{j \in \Gamma^{\ell+1}_{\boldsymbol{k}}} c^{\ell+1}_j(s) \beta^{\ell+1}_j(\boldsymbol{u}), \quad \text{for } \boldsymbol{u} \in \Omega, \tag{8}$$

with suitable coefficients $c^{\ell+1}_j(s)$ for each $j \in \Gamma^{\ell+1}_{\boldsymbol{k}}$. The truncation of $s \in V^\ell$ at level $\ell+1$ is defined as the sum of the terms in (8), related to the basis functions whose supports are not contained in $\Omega^{\ell+1}$, namely,

$$\operatorname{trunc}^{\ell+1}(s) := \sum_{\substack{j \in \Gamma^{\ell+1}_{\boldsymbol{k}} \\ \operatorname{supp}\left(\beta^\ell_j\right) \not\subseteq \Omega^{\ell+1}}} c^{\ell+1}_j(s)\, \beta^{\ell+1}_j.$$

In addition, the cumulative truncation with respect to all finer levels is defined as

$$\operatorname{Trunc}^{\ell+1}(s) := \operatorname{trunc}^{L-1}\left(\operatorname{trunc}^{L-2}\left(\ldots\left(\operatorname{trunc}^{\ell+1}(s)\right)\ldots\right)\right), \text{ with } \operatorname{Trunc}^L(s) \equiv s, \text{ for } s \in V^{L-1}.$$

The THB-spline basis can be then defined as

$$\mathcal{T}_{\boldsymbol{k}}(\mathcal{M}) := \left\{ \tau^\ell_j = \operatorname{Trunc}^{\ell+1}\left(\beta^\ell_j\right) \mid j \in A^\ell_{\boldsymbol{k}},\ \ell = 0, \ldots, L-1 \right\},$$

where the B-spline β^ℓ_j is the *mother* B-spline of the truncated τ^ℓ_j. Note that THB-splines are non-negative, have local supports, and form a partition of unity. Moreover, the truncation mechanism ensures that the THB-splines have the same or smaller support than HB-splines, as can be noticed in Figure 2. More precisely, by considering the hierarchical grid of Figure 1 (d), the coarser HB-basis functions with bi-degree $\boldsymbol{d} = (2, 2)$ that overlap the refinement interface are shown in Figure 2

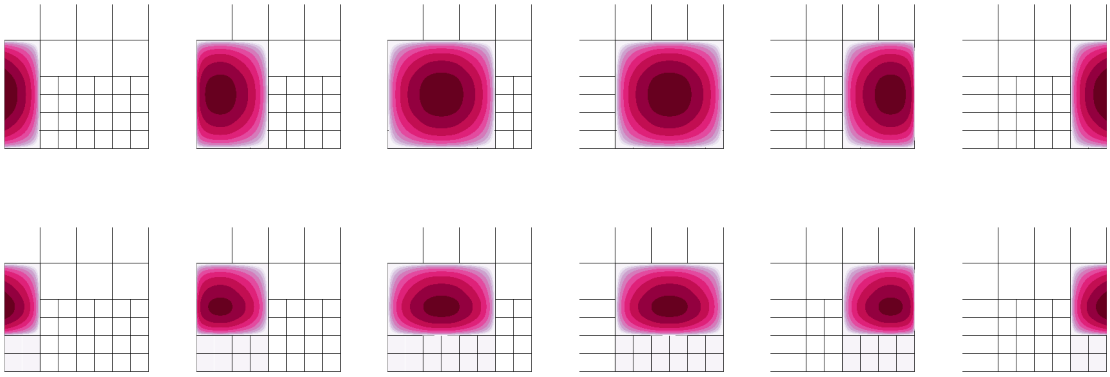


Figure 2: The HB-basis functions with bi-degree $\boldsymbol{d} = (2, 2)$ of level 0 overlapping the refinement interface (top) and their truncated version (bottom); the underlying hierarchical grid with 2 levels is also shown.

(top), and their truncation across 1 level is illustrated in Figure 2 (bottom). According to [21], the THB-splines form an alternative basis for the hierarchical spline space, namely

$$V = \operatorname{span}\{\mathcal{H}_{\boldsymbol{k}}\} \equiv \operatorname{span}\{\mathcal{T}_{\boldsymbol{k}}\}.$$

THB-splines have been proven to be a valuable tool both for geometric modeling and isogeometric analysis, see e. g., [20, 6], among others.

We may consider a suitable global index mapping for the THB-basis, $(j, \ell) \mapsto J$, with $j \in A^{\ell}_{\boldsymbol{k}}$, $\ell = 0, \ldots, L-1$ and $J = 0, \ldots, M$. Therefore, a general THB-spline model takes the form

$$\boldsymbol{s}(\boldsymbol{u}) = \sum_{J=0}^{M} \boldsymbol{c}_J \tau_J(\boldsymbol{u}), \quad \boldsymbol{u} \in \Omega, \tag{9}$$

with $\boldsymbol{c}_J \in \mathbb{R}^D$ for $J = 0, \ldots, M$. If $N = 1$ a planar ($D = 2$) or spatial ($D = 3$) THB-spline curve can be specified, whereas if $N = 2$ and $D = 3$ a THB-spline surface can be assembled. In this paper, we address industrial applications in the case $N = 2$ and $D = 3$. Moreover, we choose $\Omega = [0, 1]^2$. In the remainder of the paper we restrict our discussion to this setting.

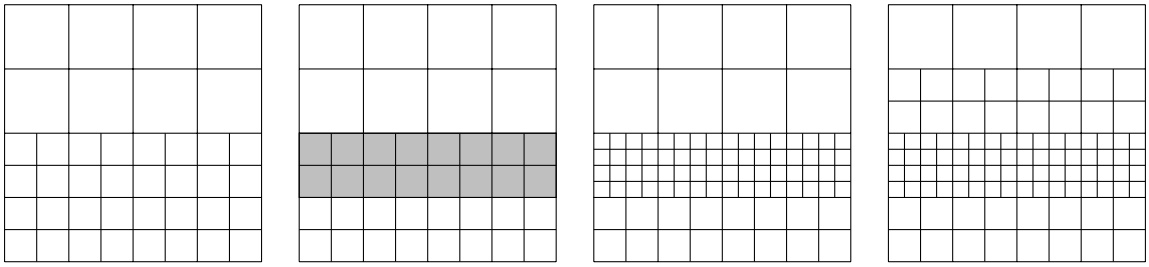
2.3. Admissible hierarchical refinement

In the theory of adaptive methods based on (truncated) hierarchical splines, suitably graded mesh configurations with a bounded number of non-zero basis functions on any mesh element need to be considered. A family of *admissible* meshes with this property was introduced in [9] to prove the convergence of adaptive isogeometric methods with THB-splines.

A hierarchical mesh $\mathcal{M}$ is admissible of class μ if the non-zero truncated functions in $\mathcal{T}_{\boldsymbol{k}}(\mathcal{M})$ over any mesh element $q \in \mathcal{M}$ belong to at most μ successive levels. To guarantee this, with every element marked for refinement a certain sets of (coarser) additional elements has to be refined as well. For example, when an element q of level ℓ has to be split into refined elements of level $\ell+1$, any THB-spline of level $\ell - \mu + 1$ acting on q (i. e. with q in its support) has to be zero on these refined elements. A refinement algorithm that enables the construction of this kind of admissible meshes for THB-splines was introduced in [9]. An example of hierarchical refinement and admissible hierarchical refinement for bi-degree $\boldsymbol{d} = (2, 2)$ and $\mu = 2$ is shown in Figure 3.

3. Adaptive THB-spline surface fitting with moving parameters

The core problem of this work is the fitting problem presented in Section 1. The *adaptive* fitting problem with THB-splines consists of finding a parametric THB-spline surface $\boldsymbol{s} : [0, 1]^2 \to \mathbb{R}^3$ as in (9), that "best" approximates the point cloud $\mathcal{P}$, according to (2), by iteratively updating the THB-spline geometry $\boldsymbol{s}$ and enriching the THB-spline basis and space. More specifically, the adaptive approximation procedure is characterized by four main steps that are successively repeated after selecting an initial parameter and mesh configuration.



(a) (b) (c) (d)

Figure 3: Given a hierarchical grid with 2 refinement levels (a), a set of elements marked for further refinement (b), the refined hierarchical grid with 3 levels obtained by dyadic splitting of the marked elements (c), and the refined hierarchical grid with 3 levels obtained by imposing admissibility, that propagates the refinement also in a suitable set of neighboring elements (d).

1. Solve. This step computes an approximating surface on the current mesh. In particular, by collecting the control points of $\boldsymbol{s}$ in a set

$$\mathcal{C} \coloneqq \{\boldsymbol{c}_J \mid J = 0, \dots, M\},$$

and the parametric values associated with the points $\mathcal{P}$ in

$$\mathcal{U} \coloneqq \left\{\boldsymbol{u}_i = (u_i, v_i) \in [0,1]^2 \mid i = 1, \dots, m\right\}, \tag{10}$$

we define a general objective function that we aim to minimize as

$$f(\mathcal{C}, \mathcal{U}) = \sum_{i=1}^{m} \operatorname{dist}^2 \left(\boldsymbol{s}(\boldsymbol{u}_i) - \boldsymbol{p}_i\right) + \lambda E(\boldsymbol{s}). \tag{11}$$

The term E is a regularization function, whose influence is tuned by a weight $\lambda \geq 0$. Energy functionals are commonly considered in spline geometric modeling problems, in particular when reconstructing a spline geometry from scattered or sparse data. They are usually introduced to compensate for robustness issues, such as oscillations of the fitted surface or the conditioning of the fitting matrix. In our case, we define E as the thin-plate energy functional, expressed as

$$E(\boldsymbol{s}) = \int_{[0,1]^2} \left(\left\| \frac{\partial^2 \boldsymbol{s}}{\partial u \partial u} \right\|_2^2 + 2 \left\| \frac{\partial^2 \boldsymbol{s}}{\partial u \partial v} \right\|_2^2 + \left\| \frac{\partial^2 \boldsymbol{s}}{\partial v \partial v} \right\|_2^2 \right) \mathrm{d}u\mathrm{d}v. \tag{12}$$

In particular we adopt the thin-plate energy (12) consistently with [29]. Moreover, we control its influence with a suitably chosen constant value for λ that is kept fixed during the entire fitting process. This choice allows for a subsequent fair comparison with existing adaptive scattered data THB-spline fitting schemes.

In addition to the thin-plate energy, other smoothing terms could be considered, such as the exact thin plate energy based on the principle curvatures of each parametric direction [23], or the data dependent thin plate energy defined in [41], which involves the second order derivative of the (re-)constructed surface by mean of the trace of the corresponding hessian. Finally, non-constant regularization weight functions for data fitting via least-squares tensor-product splines have been recently proposed in [40, 31], see also the references therein, e.g., [11, 53, 24].

Note that the minimization problem in (11) is non-linear and addresses the update of the surface approximation by computing the new control points $\mathcal{C}$ as well as the parameter values $\mathcal{U}$ associated with the input points. Therefore, the adaptive model construction deals with a *moving* parameterization and this choice enables the definition of adaptive THB-spline fitting with moving parameters.

2. Estimate. This step estimates the approximation error. The THB-spline approximant resulting from the first step is evaluated at the parameter values $\mathcal{U}$, also obtained in the first step. The Euclidean point-wise error $\|\boldsymbol{s}(\boldsymbol{u}_i) - \boldsymbol{p}_i\|_2$ for $i = 1, \dots, m$, is computed.

3. Mark. In this step we mark the regions of the parametric domain, where more degrees of freedom are desirable to satisfy the prescribed accuracy. This is done by marking any elements containing a parameter $\boldsymbol{u}_i \in \mathcal{U}$ such that $\|\boldsymbol{s}(\boldsymbol{u}_i) - \boldsymbol{p}_i\|_2 \geq \epsilon$.

4. Refine. Finally, we refine the marked elements, potentially together with some of their neighbors, which enriches the approximation space for the next iteration. We restrict ourselves to dyadic refinement and consider the following two extension choices, illustrated in Figure 4.

- *4.a* We refine the collected elements together with a certain number of surrounding rings of elements in the hierarchical mesh.
- *4.b* We refine the collected elements together with coarse mesh elements in their neighborhood to enforce the admissibility of the successively refined mesh.

After the refinement step above, a new iteration of the adaptive loop begins.

More precisely, when strategy *4.a* takes place, for any marked element of level ℓ, $\ell = 0, \dots, L-1$, we consider a certain number of surrounding rings of neighboring elements (whose value depends on the choice of an extension parameter) with respect to the grid of level $\ell+1$ and mark for refinement all elements of coarser levels which intersect this region. Figure 4 (a–d) shows an example for bi-degree $\boldsymbol{d} = (2,2)$ with the extension parameter equals to two. This choice, also considered in the numerical examples, guarantees that for any marked element one or more B-splines of level $\ell+1$ is activated on the resulting refined mesh up to bi-degree $\boldsymbol{d} = (5,5)$. This strategy is inspired by [29].

Strategy *4.b*, instead generates admissible meshes of class μ by recursively refining all the elements in a certain neighborhood of any marked element. Given an element q of level ℓ its neighborhood with respect to the chosen class of admissibility μ is defined as

$$\mathcal{N}(q,\mu) := \left\{ q' \in \mathbb{G}^{\ell-\mu+1} : q' \cap S(q, \ell-\mu+2) \neq \emptyset \right\},$$

where the multilevel support extension of q with respect to level k is given by

$$S(q,k) := \bigcup \left\{ \operatorname{supp}\beta^k : \beta^k \in \mathcal{B}^k \text{ and } q \cap \operatorname{supp}\beta^k \neq \emptyset \right\}.$$

The neighborhood collects the set of active elements of level $\ell-\mu+1$ (in the current hierarchical mesh) that intersect the multilevel support extension of q with respect to level $\ell-\mu+2$. The choice of level $k = \ell-\mu+2$ for the support extension yields the smallest value that is needed for preserving the class of admissibility of the mesh when subdividing any marked element. Note, however, that depending on the current hierarchical mesh configuration, THB-splines could also be truncated at different intermediate levels. Figure 4 (a, e–g) shows an example for bi-degree $\boldsymbol{d} = (2,2)$ and $\mu = 2$. The admissible refinement strategy naturally encapsulates a certain structure in the hierarchical mesh configuration, leveraging the support of hierarchical B-splines to limit the interaction of basis functions introduced at different hierarchical levels. This key property, obtained by iteratively marking the elements in the neighborhood of an initial set of marked elements until this neighborhood set is empty, is missing in strategy *4.a*. For additional details on admissible refinement and related algorithms, we refer to [9].

Figure 5 illustrates these two refinement strategies when the diagonal element of the current hierarchical mesh, together with the neighboring elements on the upper and lower diagonal are successively marked for refinement. We start with an initial 4×4 tensor-product mesh and consider bi-degree $\boldsymbol{d} = (2,2)$. The hierarchical meshes obtained by applying the refinement strategies *4.a* and *4.b* are shown on top and bottom of Figure 5, respectively, after $1, \dots, 5$ refinement steps (from left to right).

In the following, we employ this general procedure and we explicit each step in the process, with special focus on the first step. In particular, we propose novel adaptive fitting schemes with THB-splines based on different error metrics and compare the behavior of different optimization problems for the critical task of the control points and parameters.

Non-linear methods require an initial fitting surface and point parameterization to begin with. In several works such as [4, 58], the initial guess is a simple predicate that is chosen manually. The key influence of a good initial approximation for adaptive surface fitting with moving parameters has been analyzed in [18]. Similarly, the critical role of a good initial parameterization for adaptive surface fitting has been investigated in [19].

Throughout the rest of this work, we will make a further assumption concerning the input point cloud. More precisely, given a scattered point-cloud as in (1), we assume $\mathcal{P}$ to be the disjoint union between interior and boundary points, i. e.

$$\mathcal{P} = \mathcal{P}_I \dot{\cup} \mathcal{P}_B, \tag{13}$$

with

$$\mathcal{P}_I := \{\boldsymbol{p}_i \in \mathbb{R}^3 \,|\, i = 1, \dots, n\}, \quad \text{and} \quad \mathcal{P}_B := \{\boldsymbol{p}_i \in \mathbb{R}^3 \,|\, i = n+1, \dots, m\}.$$

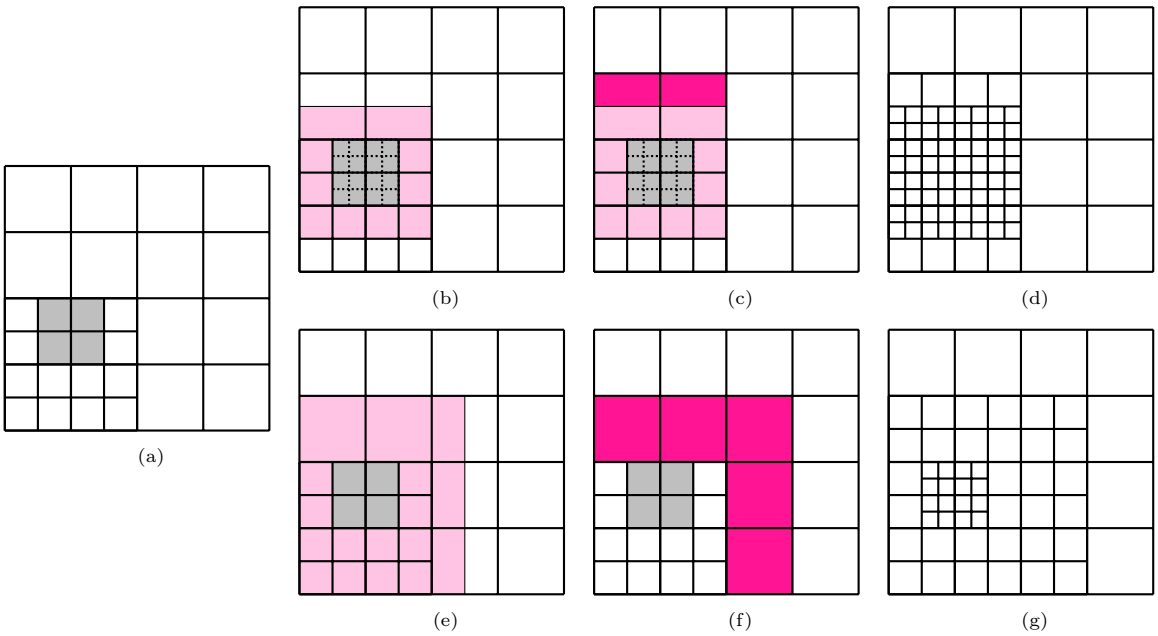


Figure 4: Refinement strategies *4.a* (top) and *4.b* (bottom). We consider bi-degree $\boldsymbol{d} = (2, 2)$, open knot vectors in both directions and internal knots with multiplicity one. For the light gray elements of level ℓ (a) marked for refinement, we plot in light pink the part of the domain considered in the refinement strategies *4.a* and *4.b*, respectively (b,e). We plot in dark pink the additional marked elements, after alignment with the grid lines of the corresponding levels (c,f). The final refined meshes are shown in (d) and (g).

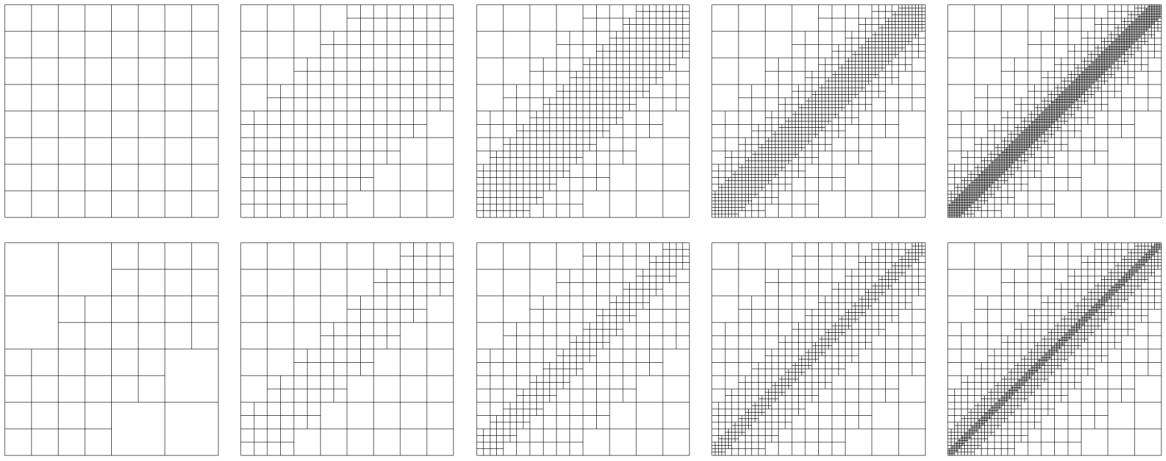


Figure 5: Hierarchical meshes obtained by refinement strategy *4.a* (top) and *4.b* (bottom) after $1, \ldots, 5$ refinement steps (from left to right) with bi-degree $\boldsymbol{d} = (2, 2)$. For this illustration we give every time as marked elements to the refinement routine all elements touching the diagonal of the domain.

4. Adaptive alternating fitting schemes

Typical Computer-Aided Geometric Design methods usually decouple the non-linear problem in (11) into two smaller sub-problems, treated separately, i. e. the computation of the parameters $\mathcal{U}$ and the computation of the control points $\mathcal{C}$. The idea of adapting the parameterization to the fitted geometry was introduced in [25], in the context of B-spline curve approximation. Specifically, starting from a certain geometric approximation, the so-called parameter correction (PC) method is an iterative procedure consisting of projecting the fitted points $\mathcal{P}$ onto the current geometry and considering the projected foot-points as the corrected parameter values. The control points of the B-spline geometry are subsequently re-computed with the new parameter values. Usually, each re-fit will significantly improve the reconstructed geometric model, hence only a few PC steps are enough to improve the accuracy of the final approximation. The core of the PC method relies on efficient foot-point projection, which

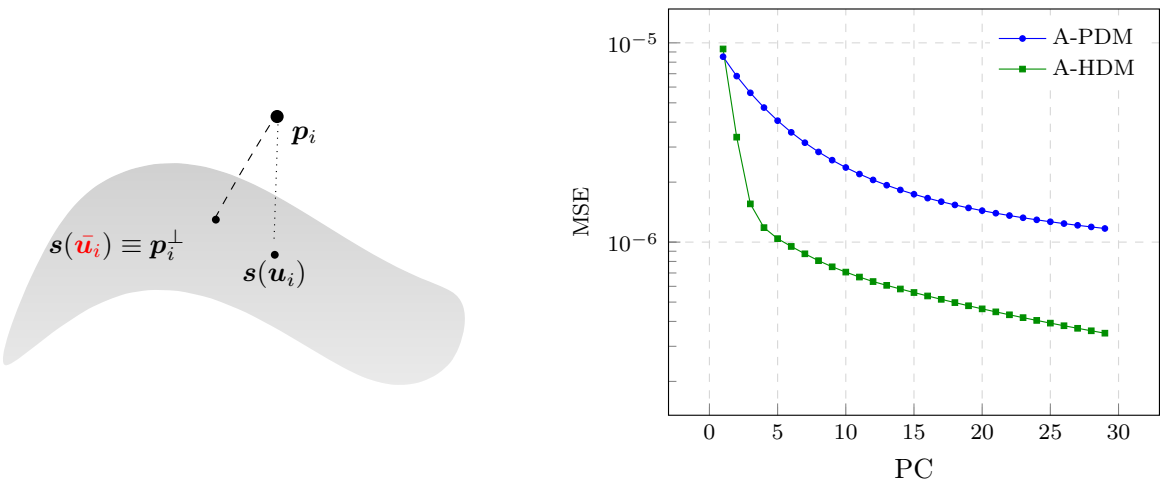


Figure 6: Left: Foot-point projection. Right: comparison of MSE evolution through 30 PC steps for A-PDM and A-HDM, with the weights of [38], for fitting a ship hull point cloud model (with 9384 points) using tensor-product B-splines of bi-degree $\boldsymbol{d} = (2, 2)$ on a 5×2 tensor-product mesh and initial standard meshless parameterization.

can be reduced to a non-linear minimization problem. The iterative approach of [25] for finding the foot-point has been revisited in [49], where a Newton-like method is proposed for the problem.

For a fixed spline space, given an initial parameterization and an initial geometric approximation, the alternating methods iterate over between the following two steps:

1. foot-point projection and consequent parameter update;
2. control points computation.

We refer to the accomplishment of both 1. and 2. as performing one step of PC.

4.1. Foot-point projection and parameter update

Foot-point projection is the process of finding for each data point the closest point on the surface and its parameter. It can be computed, e. g., using an iterative non-linear solver. More precisely, given a point cloud $\mathcal{P}$, its parameterization $\mathcal{U}$, and a surface $\boldsymbol{s} : [0, 1]^2 \to \mathbb{R}^3$, the surface closest point problem consists in solving the following minimization problem,

$$\min_{\boldsymbol{u}_i \in [0,1]^2} \frac{1}{2} \left\| \boldsymbol{s}\left(\boldsymbol{u}_i\right) - \boldsymbol{p}_i \right\|_2^2, \qquad \text{for each } i = 1, \ldots, m. \tag{14}$$

This two-dimensional non-linear problem can be explicitly formulated as

$$\left(\boldsymbol{s}\left(\boldsymbol{u}_i\right) - \boldsymbol{p}_i\right)^T \nabla \boldsymbol{s}\left(\boldsymbol{u}_i\right) = 0, \text{ for each } i = 1, \ldots, m, \tag{15}$$

where $\nabla \boldsymbol{s}$ indicates the gradient of the surface $\boldsymbol{s}$, and solved by employing a suitable optimizer. In view of (15), the vector connecting the data point $\boldsymbol{p}_i$ to the surface point $\boldsymbol{s}(\boldsymbol{u}_i)$ has to be orthogonal to the tangent plane of the surface, and $\boldsymbol{s}\left(\boldsymbol{u}_i\right)$ is then usually called the *foot-point* of $\boldsymbol{p}_i$ over $\boldsymbol{s}$ for each $i = 1, \ldots, m$. The updated parameterization $\bar{\boldsymbol{u}}_i$ is defined as the solution of (15), for $i = 1, \ldots, m$. Figure 6 (left) illustrates the projection of a point $\boldsymbol{p}_i$ and its foot-point over a surface $\boldsymbol{s}$. The connection between the data point and the surface point is represented by a dotted line, whereas the connection between the data point and its foot-point over the surface is represented by a dashed line. Finally, the updated parameter $\bar{\boldsymbol{u}}_i$ to be associated with $\boldsymbol{p}_i$ is highlighted in red.

We refer to [18] for a more comprehensive study of the interaction between foot-point projection and adaptive spline configurations, as well as the influence and importance of the optimizer used.

4.2. Control point computation

Depending on the strategy employed to compute the control points, different methods have been developed. In this work, we compare methods characterized by the solution of a minimization problem for the computation of the control points of the spline geometry. However, there are no constraints on the control-point update scheme to be used for their definition. E. g., in [18] for the second step of the alternating scheme a two-stage quasi-interpolation problem is also considered.

4.2.1. Alternating Point Distance Minimization

Minimizing at each adaptive iteration the sum of the point-wise squared distance between each data point and the corresponding point on the spline model, i. e.

$$\min_{\mathcal{C}} \frac{1}{2} \sum_{i=1}^{m} \left\| \sum_{J=0}^{M} \boldsymbol{c}_J \tau_J(\boldsymbol{u}_i) - \boldsymbol{p}_i \right\|_2^2 + \lambda E\left(\mathcal{C}\right), \tag{16}$$

with respect to the coefficients $\mathcal{C}$ leads to the formulation of the alternating point distance minimization (A-PDM) method, introduced for the first time in [25]. This method is widely used because of its simplicity; as observed in [2], from an optimization viewpoint, A-PDM exhibits linear convergence.

The linear system of normal equations to solve at each step of A-PDM is

$$\left(B^\top B + \lambda G\right) \boldsymbol{c} = B^\top \boldsymbol{p}, \tag{17}$$

where B is the collocation matrix

$$B = \begin{pmatrix} \tau_0(\boldsymbol{u}_1) & \dots & \tau_M(\boldsymbol{u}_1) \\ \vdots & \ddots & \vdots \\ \tau_0(\boldsymbol{u}_m) & \dots & \tau_M(\boldsymbol{u}_m) \end{pmatrix} \in \mathbb{R}^{m \times (M+1)}, \tag{18}$$

G is the matrix representing the contribution of the functional E, i. e.

$$G_{I,J} = \int_{[0,1]^2} \left(\frac{\partial^2 \tau_I}{\partial u \partial u} \frac{\partial^2 \tau_J}{\partial u \partial u} + 2 \frac{\partial^2 \tau_I}{\partial u \partial v} \frac{\partial^2 \tau_J}{\partial u \partial v} + \frac{\partial^2 \tau_I}{\partial v \partial v} \frac{\partial^2 \tau_J}{\partial v \partial v} \right) \mathrm{d}u\mathrm{d}v, \tag{19}$$

for $I, J = 0, \dots, M$, $\boldsymbol{p} \in \mathbb{R}^{m \times 3}$ is the matrix containing the points of $\mathcal{P}$, and $\boldsymbol{c} \in \mathbb{R}^{(M+1) \times 3}$ is the matrix containing the unknown control points.

We now introduce a different arrangement of the linear system of equations in (17), which will be useful for the following discussions, as it enables expressing A-PDM in the same format as the methods from Sections 4.2.2 and 4.2.3. Let $\widetilde{B} \in \mathbb{R}^{3m \times 3(M+1)}$ be the matrix, where the main-diagonal blocks B consist of the collocation matrix (18), and similarly, let $\widetilde{G} \in \mathbb{R}^{3(M+1) \times 3(M+1)}$ be the matrix, where the main-diagonal blocks correspond to the matrix G as in (19), representing the functional E. Hence,

$$\widetilde{B} = \begin{pmatrix} B & 0 & 0 \\ 0 & B & 0 \\ 0 & 0 & B \end{pmatrix}, \quad \widetilde{G} = \begin{pmatrix} G & 0 & 0 \\ 0 & G & 0 \\ 0 & 0 & G \end{pmatrix}. \tag{20}$$

Finally, let $\widetilde{\boldsymbol{p}} \in \mathbb{R}^{3m}$ be the reshaped vector containing the data points ordered as follows,

$$\widetilde{\boldsymbol{p}} = \left(p_1^{(1)}, \dots, p_m^{(1)}, p_1^{(2)}, \dots, p_m^{(2)}, p_1^{(3)}, \dots, p_m^{(3)} \right)^\top. \tag{21}$$

Hence, the problem in (17) can be rewritten as

$$\left(\widetilde{A}_{\mathrm{pdm}} + \lambda \widetilde{G} \right) \boldsymbol{c} = \boldsymbol{b}_{\mathrm{pdm}}, \tag{22}$$

where $\widetilde{A}_{\mathrm{pdm}} \in \mathbb{R}^{3(M+1) \times 3(M+1)}$ and $\boldsymbol{b}_{\mathrm{pdm}} \in \mathbb{R}^{3(M+1)}$, so that

$$\widetilde{A}_{\mathrm{pdm}} = \widetilde{B}^\top \widetilde{B}, \quad \boldsymbol{b}_{\mathrm{pdm}} = \widetilde{B}^\top \widetilde{\boldsymbol{p}},$$

and $\boldsymbol{c} \in \mathbb{R}^{3(M+1)}$ is the vector of unknowns, i. e.

$$\boldsymbol{c} = \left(c_0^{(1)}, \dots, c_M^{(1)}, c_0^{(2)}, \dots, c_M^{(2)}, c_0^{(3)}, \dots, c_M^{(3)} \right)^\top. \tag{23}$$

4.2.2. Alternating Tangent Distance Minimization

The Alternating Tangent Distance Minimization (A-TDM) method, introduced in [3, Chapter 6.2] for B-spline template curve fitting, and further developed in [54, 36] for B-spline curve fitting to scattered data, minimizes the squared distance from a point to the tangent plane at its foot-point on the current fitting spline model. More precisely, it is characterized by the following update rule for the control points,

$$\min_{\mathcal{C}} \frac{1}{2} \sum_{i=1}^{m} \left[\left(\sum_{J=0}^{M} \boldsymbol{c}_J \tau_J(\boldsymbol{u}_i) - \boldsymbol{p}_i \right)^T \cdot \boldsymbol{n}_i \right]^2 + \lambda \boldsymbol{E}(\mathcal{C}), \tag{24}$$

where

$$\boldsymbol{n}_i = \left(n_i^{(1)}, n_i^{(2)}, n_i^{(3)} \right)^\top \in \mathbb{R}^3$$

is the unit normal vector at the surface point $\boldsymbol{s}(\boldsymbol{u}_i)$, for each $i = 1, \ldots, m$. In particular, the term

$$\left[\left(\sum_{J=0}^{M} \boldsymbol{c}_J \tau_J(\boldsymbol{u}_i) - \boldsymbol{p}_i \right)^T \cdot \boldsymbol{n}_i \right]^2 = \left[\left(\boldsymbol{s}(\boldsymbol{u}_i) - \boldsymbol{p}_i \right) \cdot \boldsymbol{n}_i \right]^2,$$

defines the squared distance between the data point $\boldsymbol{p}_i$ and the tangent plane at $\boldsymbol{s}(\boldsymbol{u}_i)$, for each $i = 1, \ldots, m$.

Remark 1. Note that the problem in (24) is *quadratic* in $\mathcal{C}$ under the assumption that $\boldsymbol{n}_i$ does not depend on $\mathcal{C}$, since we assume $\boldsymbol{n}_i = \boldsymbol{n}_i(\mathcal{U})$ for $i = 1, \ldots, m$.

We now turn to the derivation of the linear system of equations to be solved at each step of the A-TDM method. For each physical direction $k = 1, 2, 3$, let $N_k \in \mathbb{R}^{m \times m}$ be the diagonal matrix containing the normal components at each point along direction k, i. e.

$$N_k = \begin{pmatrix} n_1^{(k)} & 0 & \ldots & 0 \\ 0 & n_2^{(k)} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & n_m^{(k)} \end{pmatrix},$$

and let $\widetilde{N} \in \mathbb{R}^{3m \times m}$ be a block matrix with blocks corresponding to N_k, i. e.

$$\widetilde{N} = \begin{pmatrix} N_1 \\ N_2 \\ N_3 \end{pmatrix} \in \mathbb{R}^{3m \times m}.$$

The problem (24) can be rewritten in matrix form as

$$\min_{\boldsymbol{c}} \frac{1}{2} \left[\left(\widetilde{B}\boldsymbol{c} - \widetilde{\boldsymbol{p}} \right)^\top \widetilde{N} \right] \left[\left(\widetilde{B}\boldsymbol{c} - \widetilde{\boldsymbol{p}} \right)^\top \widetilde{N} \right]^\top + \lambda \widetilde{G},$$

with $\widetilde{B}$, $\widetilde{G}$, $\widetilde{\boldsymbol{p}}$ and $\boldsymbol{c}$ defined as in (20), (21) and (23), respectively. Therefore, the normal equations with respect to the unknowns $\boldsymbol{c}$ are

$$\left(\widetilde{A}_{\text{tdm}} + \lambda \widetilde{G} \right) \boldsymbol{c} = \boldsymbol{b}_{\text{tdm}}, \tag{25}$$

where $\widetilde{A}_{\text{tdm}} \in \mathbb{R}^{3(M+1) \times 3(M+1)}$ and $\boldsymbol{b}_{\text{tdm}} \in \mathbb{R}^{3(M+1)}$ so that

$$\widetilde{A}_{\text{tdm}} = \widetilde{B}^\top \widetilde{N} \widetilde{N}^\top \widetilde{B}, \quad \boldsymbol{b}_{\text{tdm}} = \widetilde{B}^\top \widetilde{N} \widetilde{N}^\top \widetilde{\boldsymbol{p}}.$$

It has been observed that A-TDM does not show stable performance near high curvature regions, see, e. g., [54]. Moreover, A-TDM has been proven to be equivalent to Gauss-Newton minimization without a step-size control, see again [54], thus a regularization term needs to be added to improve stability. In particular, it is common to apply the Levenberg–Marquardt regularization, see [54, 58], by modifying (25) as

$$\left(\widetilde{A}_{\text{tdm}} + \gamma I + \lambda \widetilde{G} \right) \boldsymbol{c} = \boldsymbol{b}_{\text{tdm}}, \tag{26}$$

by adding the term γI, where I is the $3(M+1) \times 3(M+1)$ identity matrix and $\gamma \in \mathbb{R}, \gamma \geq 0$. This A-TDM variant is usually addressed in the literature as A-TDMLM, see e. g., [4, 58].

4.2.3. Alternating Hybrid Distance Minimization

The alternating hybrid distance minimization method (A-HDM) can be defined by combining the A-PDM and A-TDM methods, see, e. g., [38, 4]. In this case, the distance metric at a sample $\boldsymbol{s}(\boldsymbol{u}_i)$ for each $i = 1, \ldots, m$ takes into account both A-PDM and A-TDM metrics, i. e.

$$\gamma_i \left\| \boldsymbol{s}(\boldsymbol{u}_i) - \boldsymbol{p}_i \right\|_2^2 + \delta_i \left[\left(\boldsymbol{s}(\boldsymbol{u}_i) - \boldsymbol{p}_i \right)^\top \cdot \boldsymbol{n}_i \right]^2, \tag{27}$$

with the weights $\gamma_i, \delta_i \geq 0$ satisfying $\gamma_i + \delta_i = 1$. The objective function to be minimized over the control points is in this case

$$\min_{\mathcal{C}} \frac{1}{2} \sum_{i=1}^{m} \left\{ \gamma_i \left\| \sum_{J=0}^{M} \boldsymbol{c}_J \tau_J(\boldsymbol{u}_i) - \boldsymbol{p}_i \right\|_2^2 + \delta_i \left[\left(\sum_{J=0}^{M} \boldsymbol{c}_J \tau_J(\boldsymbol{u}_i) - \boldsymbol{p}_i \right)^T \cdot \boldsymbol{n}_i \right]^2 \right\} + \lambda \boldsymbol{E}(\mathcal{C}).$$

Remark 2. If $\gamma_i = 1$ and $\delta_i = 0$ for each $i = 1, \ldots, m$, we recover the A-PDM method. Similarly, if $\gamma_i = 0$ and $\delta_i = 1$, then we obtain the A-TDM method.

The authors in [38] suggest the following choice of the blending weights,

$$\gamma_i = \frac{\mathrm{d}_i}{2 \max_{j=1,\ldots,m} \mathrm{d}_j} \text{ and } \delta_i = (1 - \gamma_i), \text{ for } i = 1, \ldots, m,$$

for which (27) results to be a weighted combination of the A-PDM and A-TDM error terms, depending on the point-wise distance

$$\mathrm{d}_i = \left\| \boldsymbol{s}(\boldsymbol{u}_i) - \boldsymbol{p}_i \right\|_2.$$

To exploit the curvature information from the point cloud $\mathcal{P}$, the authors in [4] propose as blending weights

$$\gamma_i = \frac{\mathrm{d}_i}{\mathrm{d}_i + \rho_i} \quad \text{and} \quad \delta_i = (1 - \gamma_i) = \frac{\rho_i}{\mathrm{d}_i + \rho_i},$$

with,

$$\rho_i = \frac{1}{\max\left\{ |c_{i,1}|, |c_{i,2}| \right\}},$$

where $c_{i,1}, c_{i,2}$ are the principal curvature values at every data point $\boldsymbol{p}_i \in \mathcal{P}$, for $i = 1, \ldots, m$. Hence, (27) becomes a weighted combination of the A-PDM and A-TDM error terms, with weights depending on the surface discrete curvature as well.

Independently of the specific choice of the blending weights, the linear system of equations to be solved at each iteration of the A-HDM method has the following form,

$$\left(\widetilde{A}_{\mathrm{pdm}} + \widetilde{A}_{\mathrm{tdm}} + \lambda \widehat{G} \right) \boldsymbol{c} = \left(\boldsymbol{b}_{\mathrm{pdm}} + \boldsymbol{b}_{\mathrm{tdm}} \right), \tag{28}$$

where the matrices and right-hand sides defined in (17) and (25) include the blending weight matrices

$$\Gamma = \mathrm{diag}\{\gamma_1, \ldots, \gamma_m\} \in \mathbb{R}^{m \times m}, \ \widetilde{\Gamma} = \begin{pmatrix} \Gamma & 0 & 0 \\ 0 & \Gamma & 0 \\ 0 & 0 & \Gamma \end{pmatrix} \in \mathbb{R}^{3m \times 3m}, \ D = \mathrm{diag}\{\delta_1, \ldots, \delta_m\} \in \mathbb{R}^{m \times m}, \tag{29}$$

so that

$$\widetilde{A}_{\mathrm{pdm}} = \widetilde{B}^\top \widetilde{\Gamma} \widetilde{B}, \qquad \widetilde{A}_{\mathrm{tdm}} = \widetilde{B}^\top \widetilde{N} D \widetilde{N}^\top \widetilde{B}, \qquad \boldsymbol{b}_{\mathrm{pdm}} = \widetilde{B}^\top \widetilde{\Gamma} \widetilde{\boldsymbol{p}}, \qquad \boldsymbol{b}_{\mathrm{tdm}} = \widetilde{B}^\top \widetilde{N} D \widetilde{N}^\top \widetilde{\boldsymbol{p}}.$$

For completeness, we recall that the A-HDM method is a variant of the Alternating Squared Distance Minimization (A-SDM) method, based on a suitable *local* quadratic approximant of the squared distance function of a surface to a point. A-SDM has been developed in [46, 45] for active contour models to fit a surface to a target shape, subsequently adjusted in [54] for the problem of B-spline curve fitting to point clouds, and in [35] for the problem of surface fitting and registration.

Finally, note that A-HDM can be considered a regularized version of A-TDM. While for A-TDMLM the regularization term is a diagonal matrix γI as in (26), in this case, the role of regularization is

played by the matrix $\tilde{A}_{\text{pdm}}$ as in (28). Moreover, from all the numerical results of [4], we can infer that A-HDM with curvature-based weights and A-TDMLM can achieve the same accuracy. We also experienced that, independently of the choice of the weights, the A-HDM system matrix behaves more numerically stable than the A-TDMLM and hence is better suited for numerical computations, in particular for complex, e. g., adaptive, constructions. For this reason, we choose A-HDM over A-TDMLM to conduct the analysis developed in the following sections. Note, however, that A-PDM is nevertheless the method whose system matrix has lower condition number, being numerically even more stable than A-HDM.

4.3. Influence of the number of PC steps

A key choice for our adaptive alternating fitting schemes resides in the number of PC iterations to perform before proceeding with the next adaptive loop. Specifically, it is fundamental to find a value that provides a suitable trade-off between the approximation accuracy and computational costs.

As a preliminary study, we investigate and compare in a tensor-product B-spline framework the accuracy trend of the A-PDM and A-HDM methods up to 30 PC iterations. Specifically, we consider a point cloud sampled from a ship hull geometry, consisting of 9384 scattered data with initial standard shape preserving meshless parameterization [16]. We then set a tensor-product B-spline space, characterized by bi-degree $\boldsymbol{d} = (2, 2)$, and a 5×2 tensor-product mesh. The comparison of the mean squared error (MSE) between A-PDM and A-HDM with the weights of [38], both with a penalization weight $\lambda = 1\text{e} - 7$, for 30 PC iterations is illustrated in Figure 6 (right). We can observe that the biggest accuracy gain, i. e. more than 1 order of magnitude, is obtained in the range between 5 and 10 PC steps. This result is in line with the observation in [4]. As we proceed with the number of PC iterations, both methods tend to plateau and the difference between the MSE diminishes. This motivates us to work with 5 to 10 PC steps in adaptive frameworks. This observation will be further validated on data of industrial complexity in Section 6.

4.4. Boundary constraints for (adaptive) alternating fitting schemes

In this section we detail the boundary treatment for (adaptive) alternating fitting schemes. Specifically, we analyze the instability that might arise on the boundary of the approximation, due to parameter correction as well as the evaluation of the tangent distance at the boundary points, and provide a solution to these issues. Note that, in the literature concerning alternating fitting schemes, these kind of boundary effects were also present in the final approximation, see e. g., [4], while they do not appear if closed geometries were considered, see e. g., [58].

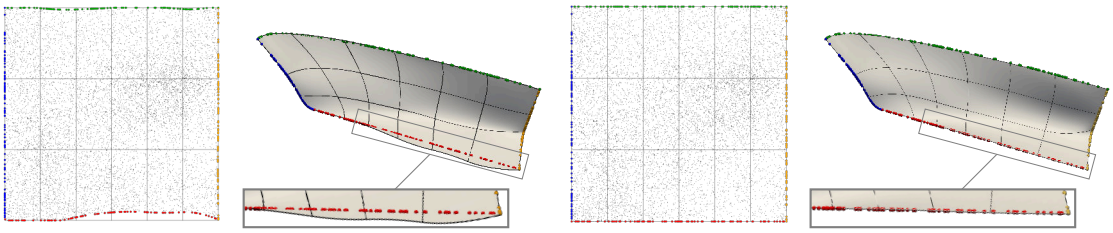
We assume the point cloud $\mathcal{P}$ to be a disjoint union of interior and boundary points, i. e. $\mathcal{P} = \mathcal{P}_I \dot{\cup} \mathcal{P}_B$, consequently the parameterization $\mathcal{U}$, defined in (10), is the disjoint union of interior and boundary parameters, i. e. $\mathcal{U} = \mathcal{U}_I \dot{\cup} \mathcal{U}_B$. Specifically, to ensure a suitable boundary approximation when moving the parameter values, we constrain the boundary parameters $\mathcal{U}_B$ to lay only on the boundary curve they belong to, hence we decouple the problem in (14) accordingly.

In particular, if $\boldsymbol{p}_i \in \mathcal{P}_I$, i. e. for interior points, we solve the problem in (14). On the other hand, if if $\boldsymbol{p}_i \in \mathcal{P}_B$, i. e. for boundary points, we solve

$$\min_{\boldsymbol{u}_i \in \partial[0,1]^2} \frac{1}{2} \left\| \boldsymbol{s}\left(\boldsymbol{u}_i\right) - \boldsymbol{p}_i \right\|_2^2, \text{ for each } i = n+1, \ldots, m,$$

where $\partial[0,1]^2$ indicates the boundary of the parametric domain $[0,1]^2$. Moreover, if the boundary of the domain is composed of more than one curve, we need to further decouple the optimization problem by constraining each boundary point to be projected only on the boundary curve it belongs to.

In Figure 7, we report an illustrative comparison for the reconstruction of a point cloud sampled from a ship hull geometry, consisting of 9384 scattered data with initial shape preserving meshless parameterization [16]. We consider a tensor-product B-spline space, characterized by bi-degree $\boldsymbol{d} = (2, 2)$, and a 5×2 tensor-product mesh, and iterate 30 times the A-PDM method without and with boundary constraints. The final parameter values, together with the tensor-product mesh, and the geometric approximations, together with the input point cloud, are shown respectively in Figure 7 (a–b), and Figure 7 (c–d). In addition, $\mathcal{U}_I$ are represented in black, whereas $\mathcal{P}_B$ and $\mathcal{U}_B$ are divided into 4 different boundaries and colored in red (south), orange (east), green (north) and blue (west), respectively. In particular, Figure 7 (a–b) shows that not constraining the boundary parameters and performing 30 PC steps affects the quality of the boundary curves and, consequently, of the one of the



(a) (b) (c) (d)

Figure 7: Tensor-product approximation of a ship hull point cloud consisting of 9384 scattered data initially parameterized with standard meshless parameterization methods. By performing the joint optimization fitting scheme without boundary constraints (a–b) the boundary values might move in the interior part of the parametric domain (a) and the boundary curves of the final approximations are affected (b). By imposing the boundary constraints (c–d) the parameter values are properly distributed over the parametric domain (c) and lead to suitable geometric approximation (d).

approximation. Moreover, the boundary curves are not well represented even with additional steps of refinement, therefore post-processing of the reconstructed geometry is necessary to recover a good approximation, e. g., via trimming. Alternatively, constraining the boundary parametric values enables us to produce boundary fitted models, as shown in Figure 7 (c–d), which can be employed in the next step of the adaptive loop.

In addition, when dealing with the boundaries of geometries given by a point cloud $\mathcal{P} = \mathcal{P}_I \dot{\cup} \mathcal{P}_B$, instability might arise from the evaluation of the tangent distance at the boundary points $\mathcal{P}_B$. When the A-TDM method is unstable, its contribution to the A-HDM is not effective. To reduce this unstable behavior, we modify the A-TDM method by decoupling the computation of the unknowns related to the interior of the surface from the computation of the unknowns for the boundary curves. In particular, $\boldsymbol{c} = \boldsymbol{c}_I \dot{\cup} \boldsymbol{c}_B$, where $\boldsymbol{c}_I$ is the solution of (28) built on the interior parameters $\mathcal{U}_I$ and points $\mathcal{P}_I$, whereas $\boldsymbol{c}_B$ is the solution of a different more stable problem, e. g., (17), built on the boundary parameters $\mathcal{U}_B$ and points $\mathcal{P}_B$. As an illustrative example, we consider the approximation of a sequence of 100 points, parameterized with chord-length method, with an open B-spline curve of degree $d = 3$, clamped knots and 5 interior uniform knots. Thus, we compare the two different scenarios for the computation of the control points when performing 1 step of the alternating methods. Specifically, we impose end-point-interpolation condition to handle the extremity of the reconstructed curve. Despite the simplicity of this instance, in Figure 8 (left) we can observe that A-TDM without any boundary treatment (blue) has heavy artefacts in correspondence to the extremity of the data sequence; meanwhile A-TDM with boundary constraints (red) suitably approximates the data. Moreover, A-TDM contributes to the definition of A-HDM, see (27), however, as shown in Figure 8 (right), if we consider A-TDM with no boundary treatment, also A-HDM is affected by artefacts towards the extremity of the geometry (blue), whereas if we suitably handle the boundary, then A-HDM results in a good approximation of the input points (red). Finally, both A-TDM and A-HDM with no boundary treatment are not suitable for more complex problem, i. e. the surface approximation of scattered data.

5. Adaptive fitting with joint-optimization

Moving parameterization methods treat the point parameters as variables that might be adjusted during the fitting process in order to optimize the final geometry. The optimization problem proper of the alternating methods discussed in Section 4 is linear with respect to the control points $\mathcal{C}$ of the spline surface $\boldsymbol{s}$ and non-linear with respect to the point parameters $\mathcal{U}$. Going one step further, we propose in this section a Joint Optimization approach to solve simultaneously the parameterization and fitting problems for a fixed spline space, i. e. to minimize the objective function (11). The simultaneous optimization of parameters and control points for B-spline curve approximation was proposed in [52], by exploiting the Levenberg-Marquardt (LM) optimization method [32, 39]. Subsequently, a faster version of this method for B-spline curve approximation was proposed in [58], where the L-BFGS optimization method [42, 10] is proposed to speed up the computational time. Finally, the extension of the joint optimization for tensor-product B-spline surface approximation was developed in [26] with different BFGS variants.

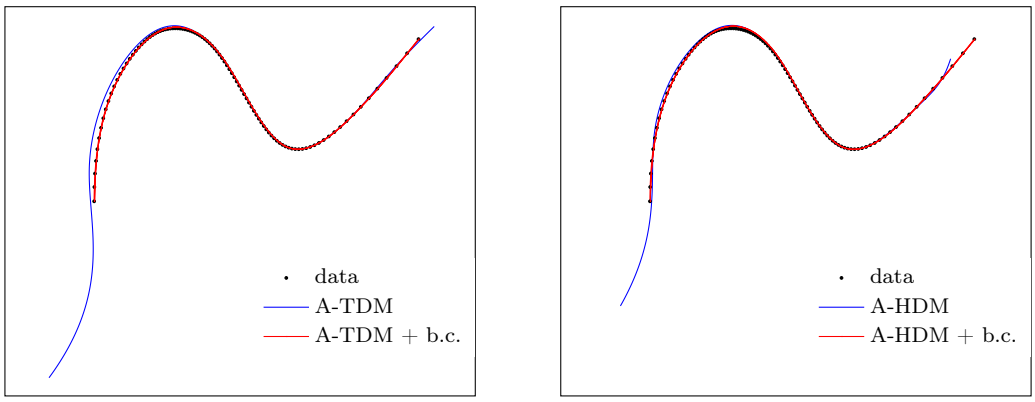


Figure 8: Left: A-TDM B-spline curve approximation without (blue) and with (red) boundary conditions. Right: A-HDM B-spline curve approximation built on A-TDM with no boundary condition (blue) and built on A-TDM with boundary condition (red). The data to be approximated are also displayed.

We extend this surface fitting optimization method to the adaptive spline framework. More precisely, we address the parameterization problem within the first step (1. Solve) of the adaptive loop by jointly, i. e. simultaneously, computing the optimal parameter sites $\mathcal{U}$ for $\mathcal{P}$ and control points $\mathcal{C}$ for $\boldsymbol{s}$. Therefore, with this method, we avoid solving a linear system of equations and performing PC routines at every adaptive iteration. We refer to this method as Joint Point Distance Minimization (J-PDM). The spline fitting problem is here addressed as a non-linear minimization problem, whose Jacobian can be explicitly computed by exploiting the properties of the (TH)B-spline basis functions.

In particular, the explicit computation of the Jacobian matrix enables the use of numerical optimization techniques. Specifically, since the explicit calculation of the Hessian matrix is costly, the BFGS optimization algorithm [8] approximates the Hessian matrix by constructing a sequence of matrices based only on Jacobian evaluations. On the other hand, for large dimensional problems, the storage of these matrices is computationally very expensive both in time and memory.

Therefore, we consider an optimization framework based on the L-BFGS optimizer [42, 10], which exploits the BFGS for the Hessian corrections, with the difference that these corrections are stored separately and, when the available storage is complete, the oldest correction is deleted to make space for the new one. The benefits of exploiting L-BFGS-based methods can be particularly appreciated to solve problems in which the Hessian matrices are large, unstructured, dense, or unavailable.

5.1. Limited memory BFGS optimization

In the following, we provide some background knowledge on the L-BFGS method, used in our fitting method to minimize (11). Let $f : \mathbb{R}^n \to \mathbb{R}$ be a non-linear function, and let $g : \mathbb{R}^n \to \mathbb{R}^n$ be its *known* gradient. The L-BFGS method is a quasi-Newton algorithm for solving large, i. e. $n \gg 50$ [10], non-linear optimization problems of the form

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}), \tag{30}$$

with respect to $\boldsymbol{x} \in \mathbb{R}^n$. The L-BFGS method approximates the inverse of the Hessian matrix of the objective function H_k by a sequence of gradient vectors from previous iterations. The user specifies the number ν of BFGS corrections that are to be kept and provides an initial sparse symmetric and positive definite matrix H_0^0, which approximates the inverse of the Hessian of f. During the first ν iterations, the L-BFGS method is identical to the BFGS method. For $k > \nu$, H_k is obtained by applying ν BFGS updates to H_k^0 using information from the ν previous iterations. More precisely, at each iteration k, the current iterate $\boldsymbol{x}_k$, the function value f_k, the gradient $\boldsymbol{g}_k$, and the approximation of the Hessian H_k are given; therefore, this allows us to compute

$$\boldsymbol{s}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k, \quad \boldsymbol{y}_k = \boldsymbol{g}_{k+1} - \boldsymbol{g}_k, \quad \rho_k = \frac{1}{\boldsymbol{y}_k^\top \boldsymbol{s}_k}, \quad M_k = I - \rho_k \boldsymbol{y}_k \boldsymbol{s}_k^\top, \quad H_k^0 = \frac{\boldsymbol{s}_{k-1}^\top \boldsymbol{y}_{k-1}}{\boldsymbol{y}_{k-1}^\top \boldsymbol{y}_{k-1}} I$$

and, consequently,

$$
\begin{aligned}
H_k = &\left(M_{k-1}^\top \dots M_{k-m}^\top\right) H_k^0 \left(M_{k-m} \dots M_{k-1}\right) \\
&+ \rho_{k-m} \left(M_{k-1}^\top \dots M_{k-m+1}^\top\right) \boldsymbol{s}_{k-m} \boldsymbol{s}_{k-m}^\top \left(M_{k-m+1} \dots M_{k-1}\right) \\
&+ \rho_{k-m+1} \left(M_{k-1}^\top \dots M_{k-m+2}^\top\right) \boldsymbol{s}_{k-m+1} \boldsymbol{s}_{k-m+1}^\top \left(M_{k-m+2} \dots M_{k-1}\right) \\
&\vdots \\
&+ \rho_{k-1} \boldsymbol{s}_{k-1} \boldsymbol{s}_{k-1}^\top.
\end{aligned}
$$

Subsequently, at the end of the k-th iteration of L-BFGS the variables are updated by

$$
\boldsymbol{d}_k = -H_k \boldsymbol{g}_k, \qquad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \gamma_k \boldsymbol{d}_k,
$$

where γ_k is a scalar variable controlling the step-size, which satisfies the Wolfe conditions [56, 57], i. e.

$$
f\left(\boldsymbol{x}_k + \gamma_k \boldsymbol{d}_k\right) \leq f\left(\boldsymbol{x}_k\right) + c_1 \gamma_k \boldsymbol{g}_k^\top \boldsymbol{d}_k, \qquad \boldsymbol{g}\left(\boldsymbol{x}_k + \gamma_k \boldsymbol{d}_k\right)^\top \boldsymbol{d}_k \geq c_2 \boldsymbol{g}_k^\top \boldsymbol{d}_k,
$$

with $0 < c_1 < c_2 < 1$. Finally, the L-BFGS algorithm terminates when the gradient of the objective function is smaller than a specified input tolerance, i. e. $\|\nabla f\| < \eta$ or a maximum number of iterations $K_{\max}$ is reached.

Remark 3. The L-BFGS algorithm uses a sequence of ν gradient vectors to approximate the inverse of the Hessian matrix. Thus, a large value of ν can potentially lead to a more accurate guess while simultaneously increasing the computational costs. According to the literature, the default suggested value to be used is $\nu = 20$ [43, 58].

5.2. Adaptive spline fitting with L-BFGS

We employ the L-BFGS algorithm to address the problem of spline surface fitting. In particular, given a point cloud $\mathcal{P}$, so that $\mathcal{P} = \mathcal{P}_I \dot{\cup} \mathcal{P}_B$, i. e. disjoint union between interior points and boundary points, for a fixed THB-spline space $V = \operatorname{span}\{\tau_0, \dots, \tau_M\}$ of dimension $M+1$, our goal consists in finding a spline model $\boldsymbol{s} \in V$ and suitable parameter values $\mathcal{U}$, which minimize

$$
f\left(\mathcal{C}, \mathcal{U}\right) = \frac{1}{2} \sum_{i=1}^{m} \left\| \sum_{J=0}^{M} \boldsymbol{c}_J \tau_J(\boldsymbol{u}_i) - \boldsymbol{p}_i \right\|_2^2 + \lambda E\left(\mathcal{C}\right), \tag{31}
$$

where $E\left(\mathcal{C}\right)$ is the functional defined in (12). The outline of the proposed optimization approach consists in the following three steps:

1. define an initial parameterization $\mathcal{U}$;
2. define the initial control points, hence the initial spline geometry $\boldsymbol{s}_0 \in V$;
3. run the L-BFGS optimizer until convergence.

Note that there are no requirements concerning the initial guesses in steps 1 and 2. Nevertheless, both the chosen parameterization and spline geometry need to be *good enough* to guarantee fast convergence to a more accurate solution. As detailed in Section 5.1, the convergence of the L-BFGS algorithm in step 3 means that either the gradient of the objective function is smaller than a specified input tolerance, or a maximum number of iterations is reached.

The unknowns of this problem are both the coefficients $\mathcal{C}$ of the THB-spline model and the parametric sites $\mathcal{U}$ associated with the data points. We then arrange them in the matrices

$$
\boldsymbol{c} \in \mathbb{R}^{(M+1)\times 3} \quad \text{and} \quad \boldsymbol{u} \in \mathbb{R}^{m\times 2},
$$

respectively. Moreover, we consider the collocation matrix $B = B\left(\mathcal{U}\right)$ as in (18), and we obtain the matrix form of the objective function (31), i. e.

$$
f(\boldsymbol{c}, \boldsymbol{u}) = \|B\boldsymbol{c} - \boldsymbol{p}\|_F^2 + \lambda E\left(\boldsymbol{c}\right) = \operatorname{trace}\left\{ \left[B\boldsymbol{c} - \boldsymbol{p}\right] \left[B\boldsymbol{c} - \boldsymbol{p}\right]^\top \right\} + \lambda E\left(\boldsymbol{c}\right), \tag{32}
$$

where $\|\cdot\|_F$ represents the Frobenius norm, $\boldsymbol{p} \in \mathbb{R}^{m\times 3}$ is a matrix containing the data points $\mathcal{P}$, and $E\left(\boldsymbol{c}\right)$ represents the contribution of the smoothing term. In order to exploit the L-BFGS optimizer, we

need to provide the gradient of the objective function (32), namely to compute the (partial) derivative of our fitting spline object with respect to each coefficient as well as with respect to each parametric site. More precisely,

$$\nabla f(\boldsymbol{c}, \boldsymbol{u}) = \left(\frac{\partial f}{\partial c_0^{(1)}}, \dots, \frac{\partial f}{\partial c_M^{(1)}}, \frac{\partial f}{\partial c_0^{(2)}}, \dots, \frac{\partial f}{\partial c_M^{(2)}}, \frac{\partial f}{\partial c_0^{(3)}}, \dots, \frac{\partial f}{\partial c_M^{(3)}}, \frac{\partial f}{\partial u_1}, \dots, \frac{\partial f}{\partial u_m}, \frac{\partial f}{\partial v_1}, \dots, \frac{\partial f}{\partial v_m} \right), \tag{33}$$

hence, by letting G as in (19), and $\boldsymbol{u} = (u, v) = (u^{(1)}, u^{(2)}) \in [0,1]^2$,

$$\begin{aligned} \nabla_{\boldsymbol{c}} f(\boldsymbol{c}, \boldsymbol{u}) &= \left(B^\top B + \lambda G\right) \boldsymbol{c} - B^\top \boldsymbol{p} \\ \nabla_{u^{(k)}} f(\boldsymbol{c}, \boldsymbol{u}) &= \frac{1}{2} \nabla_{u^{(k)}} \operatorname{trace}\left((B\boldsymbol{c} - \boldsymbol{p}) (B\boldsymbol{c} - \boldsymbol{p})^\top \right) \\ &= \operatorname{diag}\left((B\boldsymbol{c} - \boldsymbol{p}) (\partial_{u^{(k)}} B\boldsymbol{c})^\top \right), \quad k = 1, 2. \end{aligned} \tag{34}$$

The advantage of the proposed fitting method consists in avoiding the computation of the foot-point projections and solving a very large linear system of equations at every time step.

In the following, we extend the joint optimization fitting method to the adaptive THB-spline framework. In this way, we provide a new strategy to embed the moving parameterization within adaptive spline fitting schemes. Given a point cloud $\mathcal{P} = \mathcal{P}_I \dot{\cup} \mathcal{P}_B$, a suitable parameterization $\mathcal{U}$, and a THB-spline geometry $\boldsymbol{s} \in V$ as in (9), belonging to a hierarchical spline space V, at the beginning of each adaptive routine (1. Solve), we perform the joint optimization fitting algorithm for the current fixed spline space V, using as initial guesses the parameter and the control points coming from the previous adaptive loop. Subsequently, the output of the L-BFGS optimizer defines the new control points $\boldsymbol{c}_J \in \mathbb{R}^3$, for $J = 0, \dots, M$, of the spline geometry $\boldsymbol{s} \in V$ and the new parametric sites $\mathcal{U}$ of the data points, which are not kept fixed over the adaptive fitting scheme, but updated at each iteration. The remaining three adaptive steps are similar to the ones implemented for the adaptive alternating methods and detailed in Section 3.

Remark 4. After refinement, the hierarchical space has been updated by the insertion of a new hierarchical level, and a new iteration of the adaptive loop can potentially begin. Note that, in order to initialize the L-BFGS optimizer on the enlarged hierarchical spline space, we need to project the solution from the previous iteration to the new spline space.

5.3. Boundary constraints for (adaptive) fitting with joint optimization

Similarly to Section 4.4, we are again dealing with the disjoint union of interior and boundary points and, consequently, with interior and boundary parameters, i. e. $\mathcal{U} = \mathcal{U}_I \dot{\cup} \mathcal{U}_B$. As before, we need to preserve this distinction when moving te parametric values $\mathcal{U}$, hence to constrain the boundary parameters $\mathcal{U}_B$ to live only on the boundary curve they belong to. This can be achieved by discarding from the optimization variables, and (33), the parametric directions $u_j^{(k)}$, for $j = n+1, \dots, m$ and the coordinate ($k = 1$ or 2) which is perpendicular to the boundary side. Figure 9 illustrates the boundary constraints for the parametric domain $[0,1]^2$ with 4 boundary edges. More precisely, the parametric site $\boldsymbol{u}_i = (u_i, v_i)$ belongs to the west edge of the domain; hence, the only component of $\boldsymbol{u}_i$ that plays a role in the minimization of (31) is v_i, whereas u_i is kept fixed to 0. Similarly, $\boldsymbol{u}_j$ belongs to the south edge of the domain; hence, we optimize u_j while we set $v_j = 0$. Finally, also the domain corners, if any, are fixed.

In addition, we consider as a test case the reconstruction of a ship hull point cloud consisting of 9384 scattered data, initially parameterized with with shape preserving meshless parameterization method [16]. We perform the joint-optimization fitting scheme on an initial 5×2 tensor-product mesh, with bi-degree $\boldsymbol{d} = (2, 2)$, and iterate the adaptive fitting with joint optimization method up to 4 hierarchical levels. If we do not constrain the boundary parameters, the boundary parametric values are moved towards the interior of the domain, see Figure 10 (a), subsequently the boundary curves of the approximation surface are affected and the quality final geometry is ruined, see Figure 10 (b). Moreover, the adaptive refinement routine does not contribute to recover a good parameterization and approximation of the boundary curves, as shown in Figure 9 (e–f). On the other hand, imposing the boundary constraints, see Figure 10 (c), allows for a reasonable tensor-product approximation, see Figure 10 (d), that can be refined to further improve the final accuracy, see see Figure 10 (g–h).

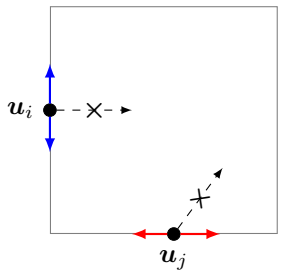


Figure 9: Two parametric sites $\boldsymbol{u}_i$ and $\boldsymbol{u}_j$ respectively constrained to the west and the south boundary edge of the rectangular domain $[0,1]^2$. The only possible directions are highlighted by blue arrows for $\boldsymbol{u}_i$ and red arrows for $\boldsymbol{u}_j$. In addition, not acceptable directions are represented by dashed arrows, marked by a cross symbol.

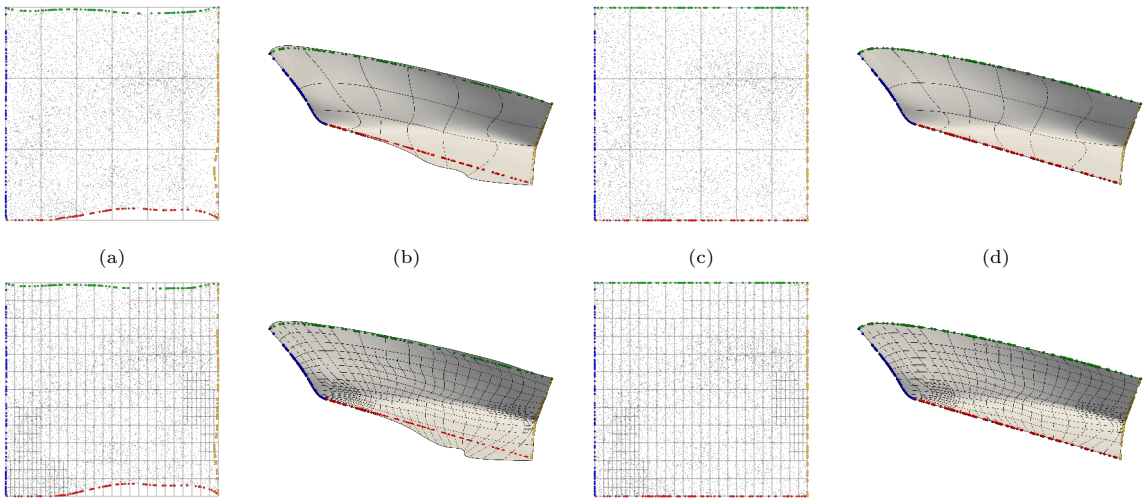


Figure 10: Tensor-product approximation of a ship hull point cloud consisting of 9384 scattered data initially parameterized with standard meshless parameterization methods. By performing the joint optimization fitting scheme without boundary constraints the boundary values might move in the interior part of the parametric domain (a–e) and the boundary curves of the final approximations are affected (b–f). By imposing the boundary constraints the parameter values are properly distributed over the parametric domain (c–g) and lead to suitable geometric approximation (d–h).

6. Numerical results

In this Section, we apply the adaptive fitting schemes with moving parameterization presented in this work for the reconstruction of aircraft engine components from scanned data of industrial complexity, parameterized with standard methods, e. g., [16], whereas the initial approximation is defined in terms of a least squares tensor-product B-spline computed on such parameterization.

We present a comparison between the adaptive A-PDM, A-HDM, and the adaptive J-PDM. Furthermore, we analyze the role of admissible hierarchical meshes within the fitting framework.

We evaluate the goodness of the final reconstructed geometric model by measuring the accuracy of the final approximation at each parameter site, in terms of absolute errors. Furthermore, it is also fundamentally important that the final geometric model is free from artifacts or oscillations, which might arise from overfitting noisy data. Therefore, we require the point-wise error to be below an input tolerance only for a certain percentage of points. This procedure is widespread when dealing with scattered and noisy data such as real-world datasets for industrial applications.

The algorithms related to the adaptive fitting schemes with THB-splines have been implemented in `C++` with the open-source G+Smo library [37]. As regards the alternating methods, we re-wrote and greatly improved the parameter projection routine for making it more robust in order to suitably tackle the industrial models addressed in this paper. As concerns the implementation of the L-BFGS optimization method, we exploit the HLBFGS `C++` library [34], which provides the L-BFGS method, the preconditioned L-BFGS method [27], and the preconditioned conjugate gradient method [48, 47].

6.1. Blade: influence of the number of PC steps

In this example, we study the influence of the number of PC steps between refinements for the adaptive A-PDM and A-HDM with error-based weights proposed in [38]. To this aim, we examine the reconstruction of a blade geometry with the length of about $5 \cdot 10^{-2}$m. In particular, we consider the same physical part as in [18, Example 4.2]; however, in this example, we use a 3D scan with higher sampling consisting of 104138, instead of 27191, scattered data points. Again, we start with a bi-cubic basis, i. e. $\boldsymbol{d} = (3, 3)$, defined on an 8×8 tensor-product mesh and work with the smoothing coefficient $\lambda = 1\mathrm{e} - 8$. In addition, we aim for 99% of data points within the tolerance $\epsilon = 1\mathrm{e} - 5\mathrm{m}$.

We compare the two adaptive fitting methods with refinement strategy *4.a* and varying number of PC steps, from 0 to 5, between refinements. More specifically, A-PDM requires 5, 4, 4, 4, 4, and 4 adaptive refinements for 0 to 5 PC steps, respectively, to reach the tolerance; the A-HDM requires 5, 4, 4, 4, 3, and 3 refinements, for 0 to 5 PC steps, respectively, to reach the tolerance. Note that for 5 PC steps, A-HDM performs 3 adaptive refinements, i. e. one less than A-PDM. This results in a natural reduction of degrees of freedom (DOF) in the final reconstructed geometry. In particular, A-HDM final reconstruction needs 2254 DOF, i. e. about 41% less DOF needed by A-PDM, to satisfy the accuracy requirements.

Figure 11 shows the resulting surfaces for A-PDM (left column) and A-HDM (right column), with 1 PC (top), 3 PC (middle), and 5 PC (bottom) steps. Furthermore, Figure 12 (left) depicts the mean square error (MSE) against the number of DOF needed to satisfy the accuracy requirements. Figure 12 (right) shows the percentage of points satisfying the input tolerance at each adaptive iteration. In particular, we plot the results obtained with A-PDM (blue) and A-HDM (green) with 0, 1, 3, and 5 PC steps.

We can see that, for both A-PDM and A-HDM, increasing the number of PC steps enables reaching a comparable error with fewer degrees of freedom, to the extent that fewer refinement iterations can be required. Aligned with the theory from [54], the improvements in A-HDM are stronger than in A-PDM. In addition, in line with the findings in [18], the first parameter correction step yields the biggest improvement, whereas, for a higher number of adaptive iterations, this influence decreases. On the other hand, 0 PC step implies that for A-HDM the unit normals are not necessarily computed at the foot-point projections of the data onto the surface, as the original definition of the A-HDM method requires.

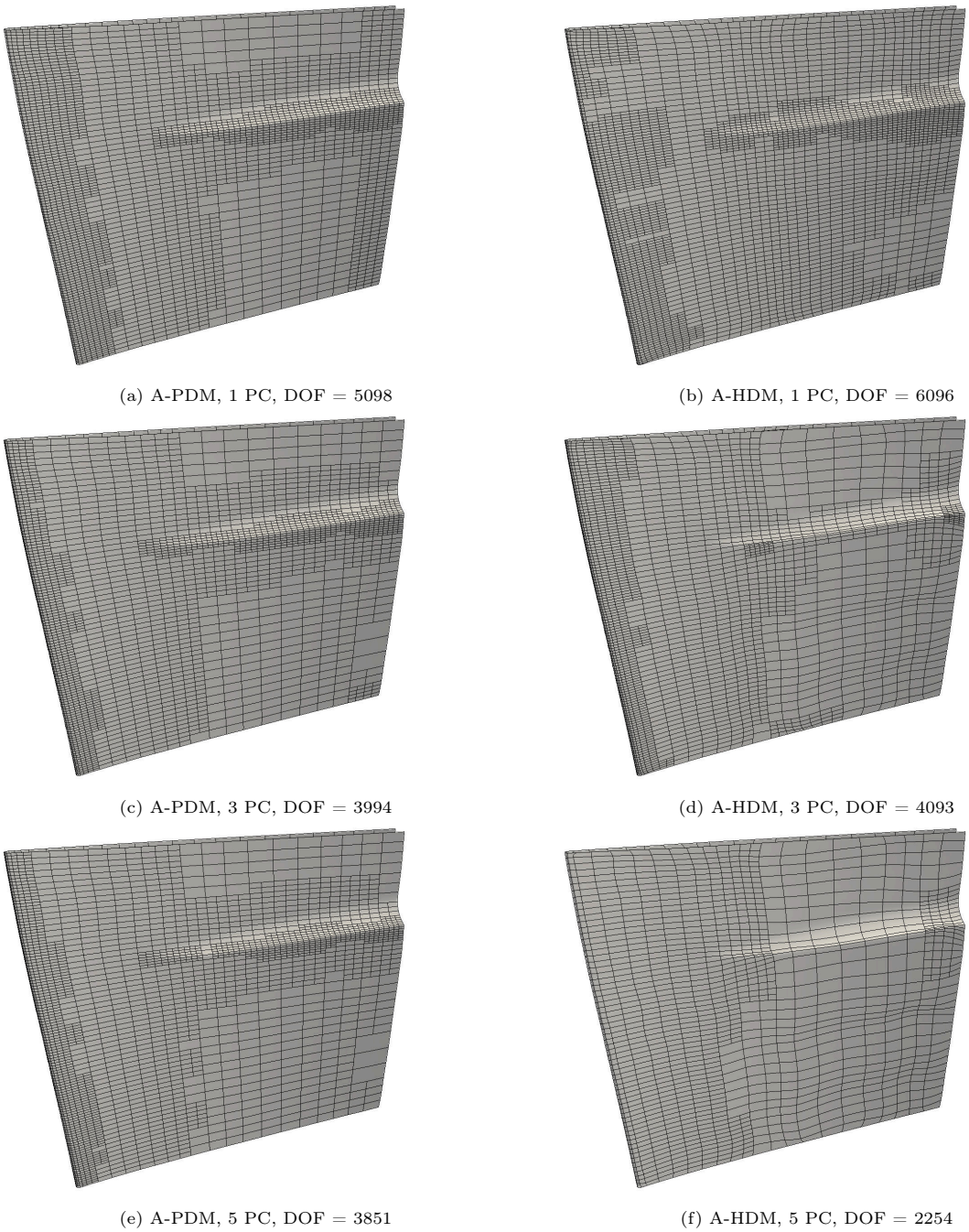
6.2. Blade: influence of the fitting method

In this example, we compare the adaptive alternating fitting methods A-PDM and A-HDM with error-based weights proposed in [38], with the adaptive J-PDM method. Moreover, we employ refinement strategy *4.a*. In particular, we consider the same blade dataset of Section 6.1, we consider the same initial approximation space and ask for the same geometric model accuracy, i. e. $\epsilon = 1\mathrm{e} - 5\mathrm{m}$ for 99% of the data points.

Based on the analysis conducted in Section 6.1, we perform 5 PC steps for A-PDM and A-HDM. As concerns J-PDM, we consider a varying maximum number of L-BFGS optimizer iterations, i. e. $K_{\max} = 100, 200, 500, 1000$. Moreover, we set the error tolerance on the gradient of the objective function to $\eta = 1\mathrm{e} - 9 \ll \epsilon\mathrm{m}$, i. e. sufficiently smaller than the input threshold, so that we have control over the exact number of L-BFGS iterations performed.

Figure 13 illustrates the final reconstructed geometries for A-PDM (top left) and A-HDM (top right) with 5 PC steps and for J-PDM with 100 (middle left), 200 (middle right), 500 (bottom left), and 1000 (bottom right) L-BFGS iterations. Figure 14 (left) depicts the MSE trend against the DOF of the final geometry. Figure 14 (right) illustrates the percentage of points below the input tolerance ϵ with respect to the DOF, for each considered method. Note that the A-PDM and A-HDM geometries are the same as Figure 11 (bottom) from Section 6.1; similarly their MSE error trends are the same as Figure 12 from Section 6.1. We plot these results again in Figure 13 and Figure 14 for a better global overview and a direct comparison between all the considered methods.

From both the qualitative and quantitative analysis, we note that J-PDM results improve with an increasing number of L-BFGS iterations, in the sense that fewer DOF are required to meet the input prescribed accuracy. More specifically, stopping the L-BFGS optimizer after 100 or 200 L-BFGS iterations results in parametric geometries that are still too far away from their optimal configurations. Therefore, to meet the accuracy requirements, several DOF need to be added at each refinement step. By letting the L-BFGS run for 500 iterations, almost the behavior of A-PDM is obtained. Finally,



(a) A-PDM, 1 PC, DOF = 5098

(b) A-HDM, 1 PC, DOF = 6096

(c) A-PDM, 3 PC, DOF = 3994

(d) A-HDM, 3 PC, DOF = 4093

(e) A-PDM, 5 PC, DOF = 3851

(f) A-HDM, 5 PC, DOF = 2254

Figure 11: Blade geometry reconstruction from Section 6.1 for a varying number of PC steps (increasing from top to bottom) for A-PDM (left column) and A-HDM with error-based weights suggested in [38] (right column). The dimension of each geometric model (DOF) is also reported.

running L-BFGS for 1000 iterations enables terminating the adaptive loop already after 3 refinements, i. e. 1 refinement earlier than A-PDM and after the same number of refinements as A-HDM, yet achieving smaller approximation error values.

In this example, we can observe that the adaptive J-PDM fitting scheme has an approximation power higher than adaptive A-PDM and A-HDM. Nevertheless, we recall that the adaptive J-PDM method is more computationally expensive since it necessitates solving a non-linear optimization problem of dimension $3(M+1)+2m$ at each adaptive loop. On the other hand, the adaptive alternating methods solve at each adaptive loop m non-linear problems of dimension 2 that are all independent from each other and a *linear* problem of dimension $3M$.

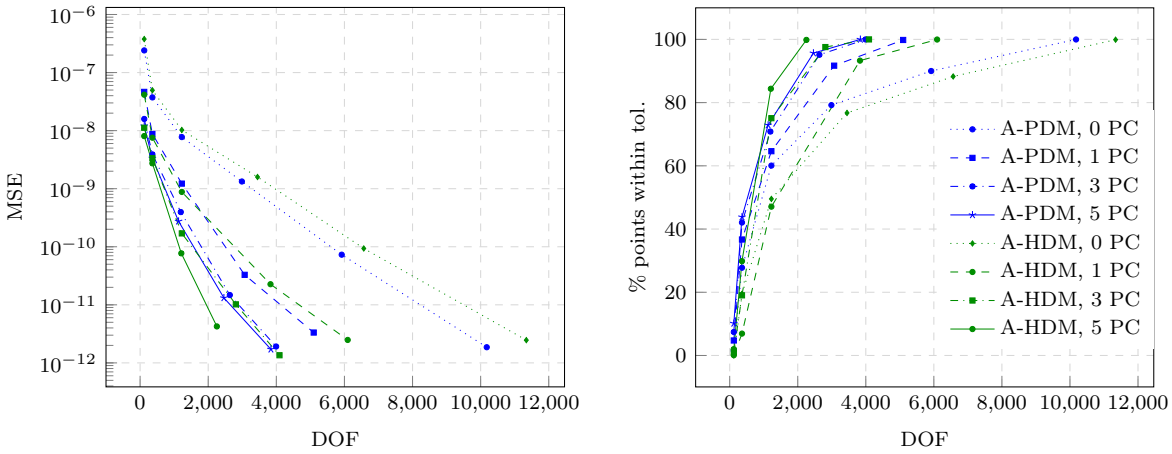


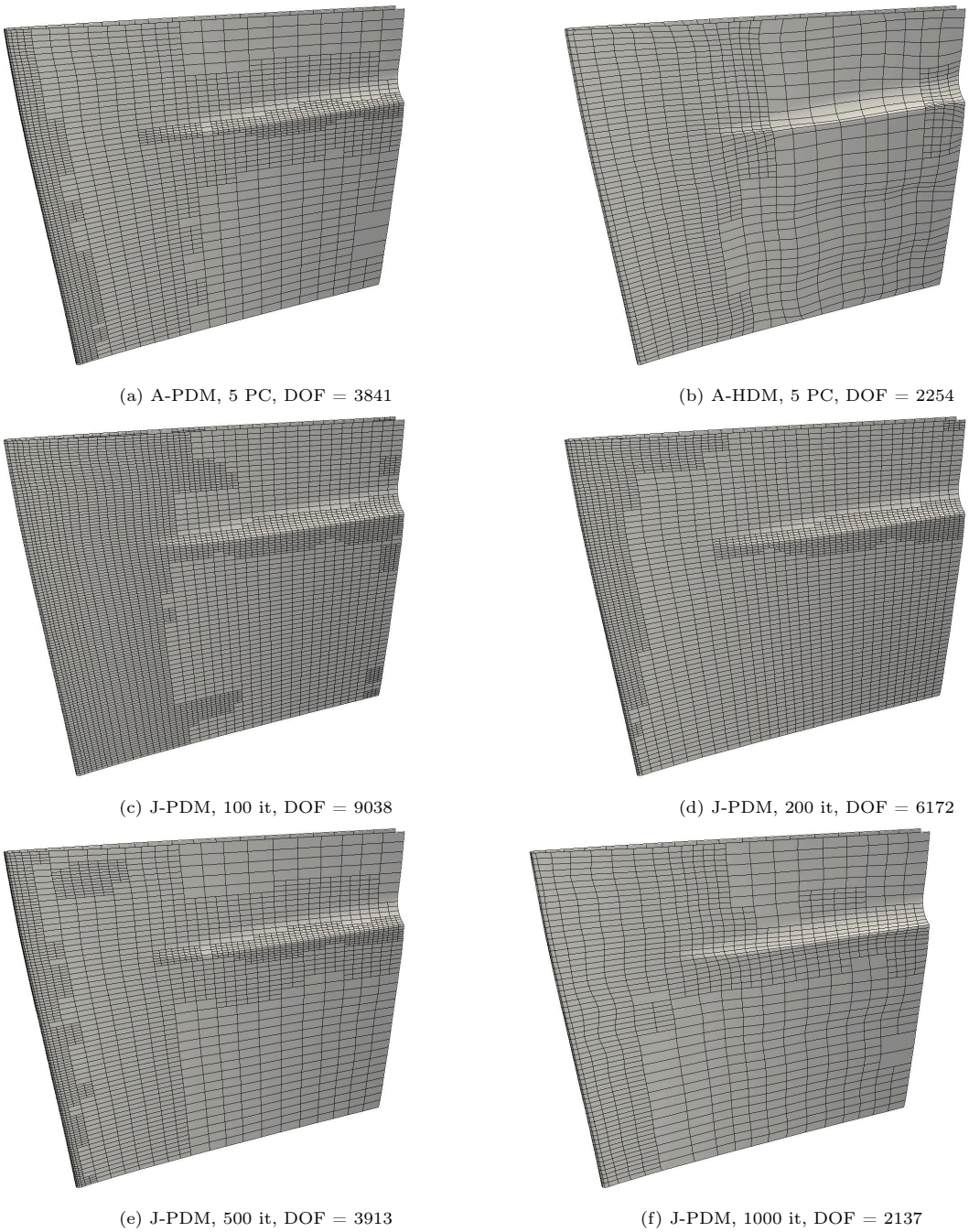
Figure 12: Blade geometry reconstruction from Section 6.1. MSE trend with respect to DOF for A-PDM (blue) and A-HDM with error-based weights suggested in [38] (green) with 0 (dotted), 1 (dashed), 3 (dashdotted), and 5 (solid) PC steps.

6.3. Airfoil: influence of the fitting methods

In this example, we compare the adaptive fitting schemes with moving parameters presented in this work, with refinement strategy *4.a*, for the reconstruction of an airfoil geometry. This aircraft component is the same considered in [5, Example 3.4]; however the initial parameterization has been modified. More precisely, the initial parameterization is shifted by 0.25 to the spot where the mesh was cut open during the parameterization step, and the stitch vertices are now duplicated, once with $u = 0$ and once with $u = 1$. This results in a new point cloud with 20034 instead of 19669 data points.

We then start with a bi-cubic tensor-product basis, i. e. $\boldsymbol{d} = (3, 3)$, on an 8×1 tensor-product mesh. We use stiffness $\lambda = 1\text{e} - 6$ and terminate the approximation when 95% of data points are within the tolerance $\epsilon = 5\text{e} - 5\text{m}$. From this initial configuration, we fit the input scattered point cloud with adaptive A-PDM, and adaptive A-HDM with error-based weights [38], performing 5 PC steps for both alternating methods. Moreover, we reconstruct the THB-spline geometry by running the adaptive J-PDM fitting algorithm with $K_{\max} = 100, 200, 500, 1000$ number of L-BFGS iterations. As in the previous examples, the gradient threshold for the optimizer is $\eta = 1\text{e}-9$, namely low enough not to contribute to the optimizer stopping criterion.

The qualitative results are reported in Figure 15, whereas the quantitative analysis is illustrated in Figure 16. Specifically, Figure 15 shows the resulting THB-spline geometries obtained with adaptive A-PDM and 5 PC steps (top left), A-HDM and 5 PC steps (top right), and J-PDM with 100 (middle left), 200 (middle right), 500 (bottom left) and 1000 (bottom right) L-BFGS iterations. Figure 16 plots the corresponding MSE (left) and percentage of points within the input tolerance (right) evolution over the geometric model DOF. From these results, we can see that the approximation power of the adaptive J-PDM fitting scheme with a sufficient number of L-BFGS iterations, is higher than that of the adaptive A-PDM and A-HDM. In essence, the adaptive J-PDM fitting scheme needs fewer DOF to meet the required input accuracy. Similar to the previous example, adaptive J-PDM with 100 or 200 L-BFGS iterations produce parametric geometries that are still too far away from potential optimal configurations, hence several DOF are added at each refinement step. Adaptive J-PDM with 500 iterations shows an MSE behavior especially aligned with adaptive A-HDM with 5 PC steps, yet with a higher number of points below the input tolerance for each adaptive iteration. These considerations can be inferred by comparing the magenta dash-dotted line (adaptive J-PDM with 500 L-BFGS iterations), with the green solid line (adaptive A-HDM with 5 PC steps) in Figure 16 (left) for the MSE and in Figure 16 (right) for the percentages. Finally, adaptive J-PDM with 1000 L-BFGS iterations achieves the accuracy requirements with the lowest number of DOF. Note that, for coarser meshes, the MSE is reduced by more than one order of magnitude compared to the adaptive alternating methods. The same remark from Figure 6.2, concerning the computational costs of the developed method, holds.



(a) A-PDM, 5 PC, DOF = 3841

(b) A-HDM, 5 PC, DOF = 2254

(c) J-PDM, 100 it, DOF = 9038

(d) J-PDM, 200 it, DOF = 6172

(e) J-PDM, 500 it, DOF = 3913

(f) J-PDM, 1000 it, DOF = 2137

Figure 13: Blade geometry reconstructions from Section 6.2 with A-PDM and 5 PC (top left), A-HDM with the error-based weights [38] and 5 PC (top right), and J-PDM with 100 (middle left), 200 (middle right), 500 (bottom left) and 1000 (bottom right) L-BFGS iterations. The dimension of each geometric model (DOF) is also reported.

6.4. Blade and airfoil geometry reconstruction with admissible meshes

In this example, we show that if spline adaptivity is driven by suitably graded hierarchical meshes, i. e. admissible refined meshes, the final THB-spline reconstruction can be further improved in terms of the trade-off between accuracy and final geometric model dimension. In particular, we revisit the examples of Section 6.2 and Section 6.3 by exploiting the refinement strategy *4.b*, considering $\mu = 2$, so that the meshes remain admissible through their adaptive design.

Blade. We start by considering the reconstruction of the blade geometry presented in Section 6.2 and compare those results with the new ones obtained via admissible refinement. Figure 17 (left) shows some of the resulting geometries from Section 6.2 with their admissible counterparts (right). In

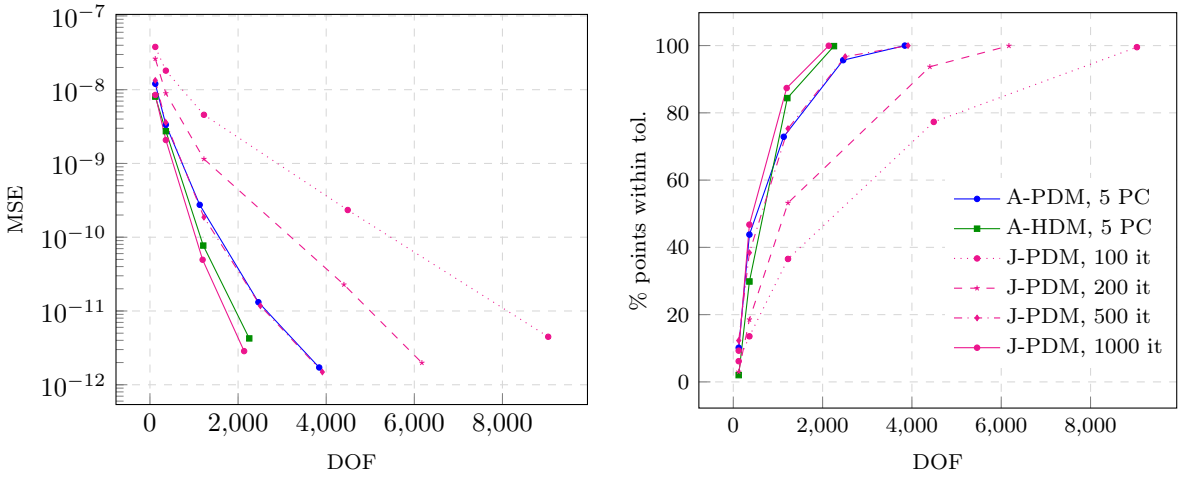


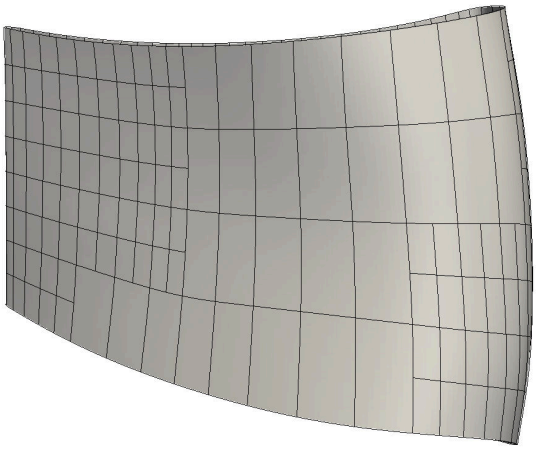
Figure 14: Blade geometry reconstruction from Section 6.2. Trends of the MSE (left) and the percentage of points below the tolerance (right) with respect to the number of DOF, for A-PDM (solid blue), A-HDM with the error-based weights [38] (solid green), both with 5 PC steps, and J-PDM (magenta) with 100 (dotted), 200 (dashed), 500 (dash-dotted) and 1000 (solid) L-BFGS iterations.

particular, we report the geometries fitted with the adaptive A-PDM with 5 PC steps (top), adaptive A-HDM with the error-based weight of [38] and 5 PC steps (middle), and the adaptive J-PDM with 1000 LBDFS iterations (bottom). In Figure 18 we show the MSE – (left) and the percentage of points within the tolerance (right) for all the above-mentioned fitting methods with standard refinement (solid lines) and with admissible refinement routines (dashed lines). From the quantitative results, we can see that enabling the admissible refinement strategy saves DOF for all the considered adaptive fitting methods while attaining similar accuracy. Nevertheless, the behavior of the fitting methods relative to each other remains analogous. In addition, from the qualitative results, we can see that admissibility also preserves the quality of the resulting geometries, i. e. no artifacts or oscillations arise. The best results in terms of a trade-off between accuracy and geometric model complexity are achieved by J-PDM with 1000 L-BFGS iterations and admissible mesh configuration, followed by A-HDM with error-based weights [38], 5 PC steps and admissible mesh refinement, J-PDM with 1000 L-BFGS iterations and standard refinement, A-HDM with error-based weights [38], 5 PC steps and standard refinement, and finally A-PDM 5 PC steps with admissible and then standard meshes. Therefore, we can conclude that involving suitably graded hierarchical meshes pronounces the quality of the final geometric model.

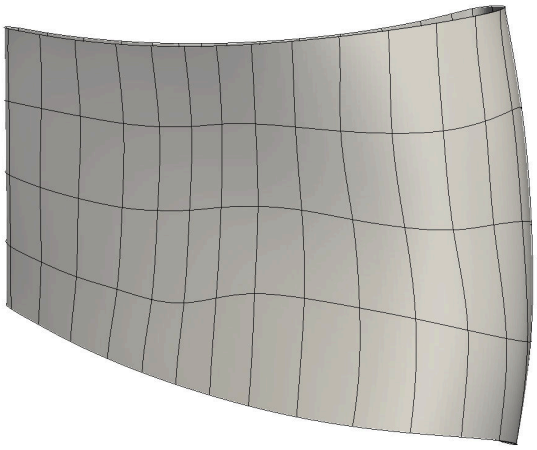
Airfoil. We now revisit the example in Section 6.3, by exploiting admissible mesh refinement strategy. In particular, we consider the adaptive A-PDM with 5 PC steps and the adaptive J-PDM with 200 and 1000 L-BFGS iterations fitting schemes and compare the final geometries reconstructed with standard and admissible refinement. Specifically, Figure 19 shows the reconstructed geometries, whereas Figure 20 plots the MSE (left) and the percentage of points below the input tolerance in relation to the number of DOF (right). Admissibility improves the final reconstructed geometric models, for all the considered fitting schemes, by saving DOF while reaching equivalent approximation accuracy. Notably, the admissible refinement strategy helps to improve the performance of adaptive J-PDM with 200 L-BFGS iterations, by reducing the final number of DOF from 599 to 369 and aligning its approximation power to adaptive A-PDM with 5 PC steps and admissible mesh configurations, see also Figure 19 (b–d). Finally, we remark that adaptive J-PDM with 1000 L-BFGS iterations achieves the best approximation results among all the considered fitting schemes, and its approximation power is further enhanced by working with suitably graded meshes.

7. Closure

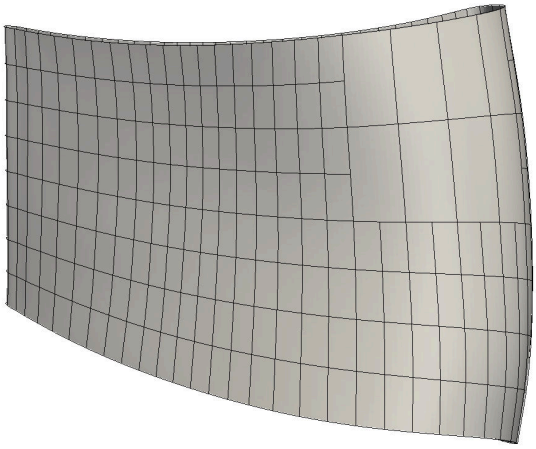
We proposed a novel paradigm for adaptive fitting methods, where both the (scattered) data parameterization and the spline local refinement are addressed within the same fitting scheme. In particular, the key interplay between parameterization and spline adaptivity is proposed in the context



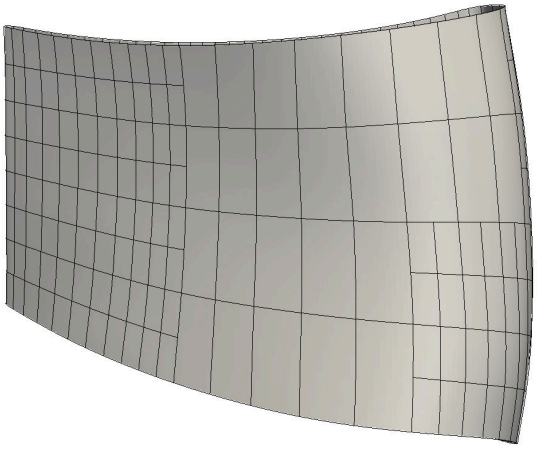
(a) A-PDM, 5 PC, DOF = 566



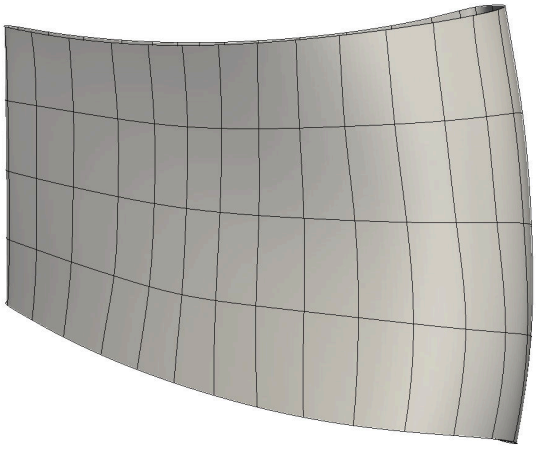
(b) A-HDM, 5 PC, DOF = 245



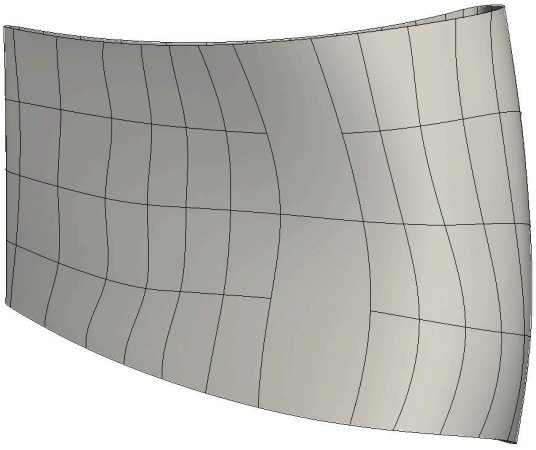
(c) J-PDM, 100 it, DOF = 695



(d) J-PDM, 200 it, DOF = 599



(e) J-PDM, 500 it, DOF = 245



(f) J-PDM, 1000 it, DOF = 230

Figure 15: Airfoil geometry reconstructions from Section 6.3 with A-PDM and 5 PC (top left), A-HDM with the error-based weights [38] and 5 PC (top right), and J-PDM with 100 (middle left), 200 (middle right), 500 (bottom left) and 1000 (bottom right) L-BFGS iterations. The dimension of each geometric model (DOF) is also reported.

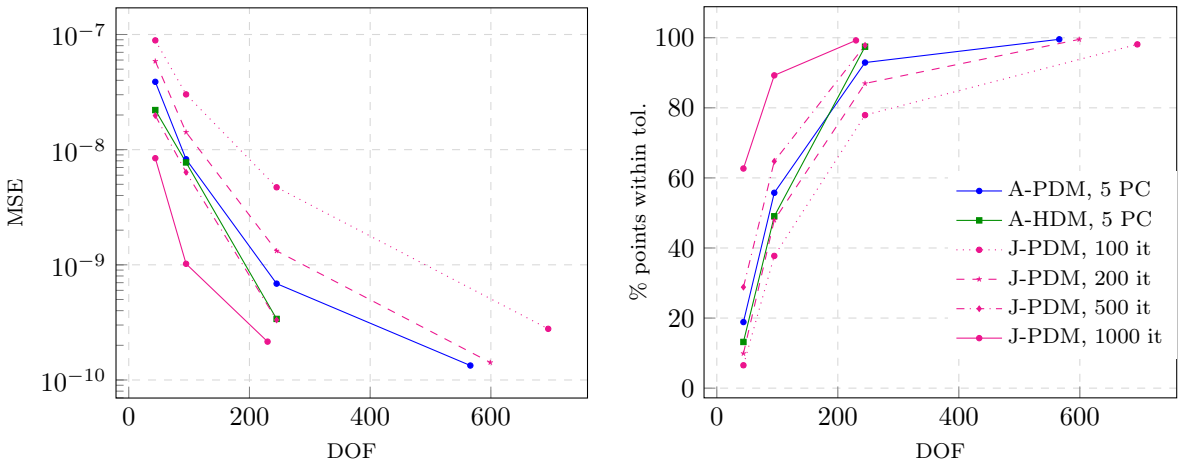


Figure 16: Airfoil geometry reconstruction from Section 6.3. Trends of the MSE (left) and the percentage of points below the tolerance (right) with respect to the number of DOF, for A-PDM (solid blue), A-HDM with the error-based weights [38] (solid green), both with 5 PC steps, and J-PDM (magenta) with 100 (dotted), 200 (dashed), 500 (dash-dotted) and 1000 (solid) L-BFGS iterations.

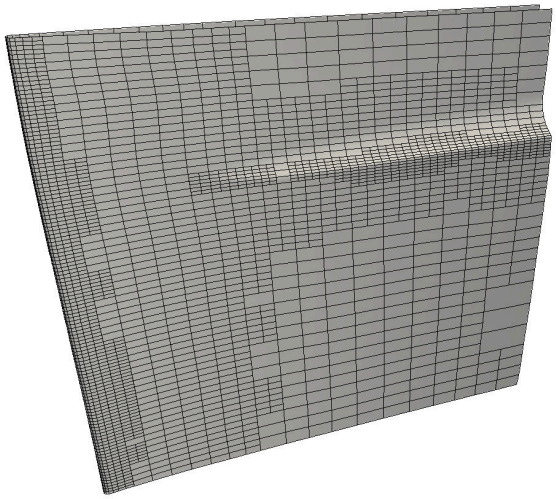
of a novel adaptive surface fitting methods with THB-splines based on alternating tangent and hybrid distance minimization. While A-TDM suffers from numerical instability, we showed that A-HDM with THB-splines strongly improves the results recently obtained in [18], where only A-PDM was considered. Nevertheless, adding a few parameter correction steps at each iteration increases the computational costs. A joint method, based on limited memory BFGS optimization to update parameter values and control points simultaneously, is then proposed to avoid the need for iteratively computing foot-point projections. The performance of J-PDM can outperform A-HDM if sufficiently many iterations are considered. Note however that increasing the number of L-BFGS iterations also increases the computational costs of the overall procedure. A suitable trade-off between the desired accuracy and the computational resources needs to be identified for challenging applications of industrial complexity, such as the ones considered in this paper. Finally, in the last example, we also showed that, when suitably graded THB-spline refinement is considered, a further reduction in the number of degrees of freedom needed to obtain a certain accuracy can be achieved both for the alternating (A-PDM, A-HDM) and the joint (J-PDM) schemes. An interesting future research direction could be the study of more general joint optimization schemes such as a J-TDM approach, possibly also including different kinds of constraints in the adaptive THB-spline fitting scheme.

CRediT author statement

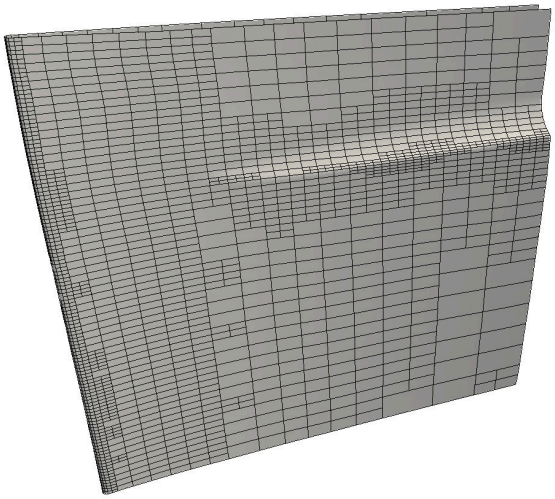
Carlotta Giannelli: Conceptualization, Methodology, Investigation, Resources, Writing - Original Draft, Writing - Review & Editing. **Sofia Imperatore:** Conceptualization, Investigation, Methodology, Resources, Software, Validation, Visualization, Writing - original draft, Writing - Review & Editing. **Angelos Matzanflaris:**. Conceptualization, Methodology, Software, Investigation, Resources, Writing - Original Draft, Writing - Review & Editing. **Dominik Mokriš:** Conceptualization, Investigation, Methodology, Resources, Software, Validation, Visualization, Writing - original draft, Writing - Review & Editing.

Acknowledgments

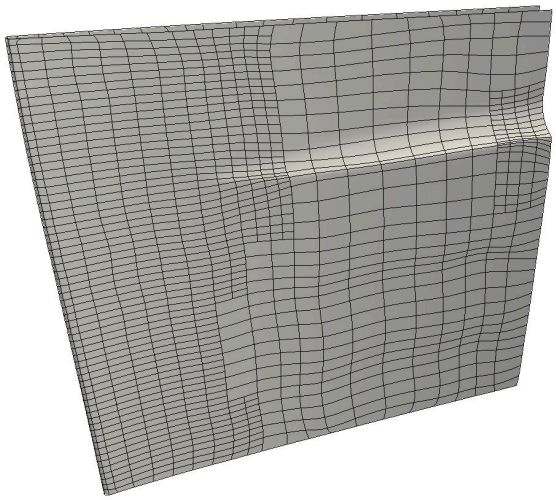
CG, SI, AM acknowledge the PHC GALILEE project 47786N. AM acknowledges the H2020 Marie Skłodowska-Curie grant GRAPES No. 860843. CG acknowledges the contribution of the National Recovery and Resilience Plan, Mission 4 Component 2 – Investment 1.4 – CN_00000013 "CENTRO NAZIONALE HPC, BIG DATA E QUANTUM COMPUTING", spoke 6. The partial support of the Italian Ministry of University and Research (MUR) through the PRIN projects COSMIC (No. 2022A79M75) and NOTES (No. P2022NC97R), funded by the European Union - Next Generation EU, is also acknowledged. CG and SI are members of the INdAM research group GNCS, whose support is gratefully acknowledged.



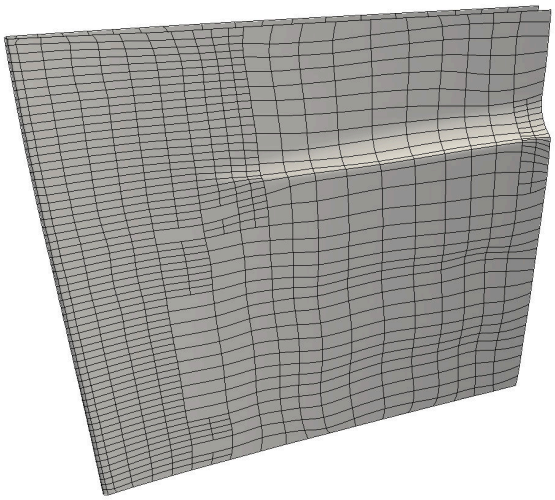
(a) A-PDM, 5 PC, DOF = 3841



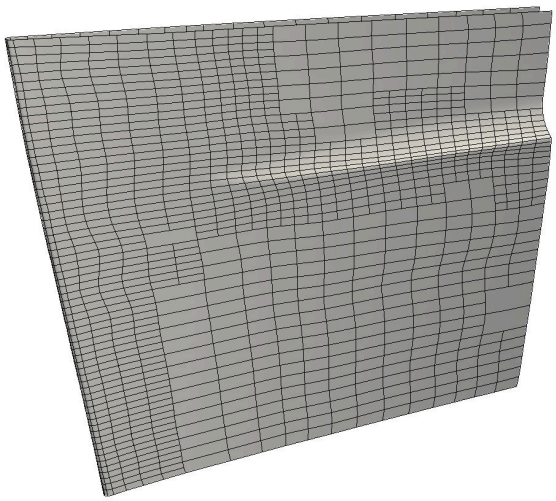
(b) A-PDM, 5 PC, adm., DOF = 2785



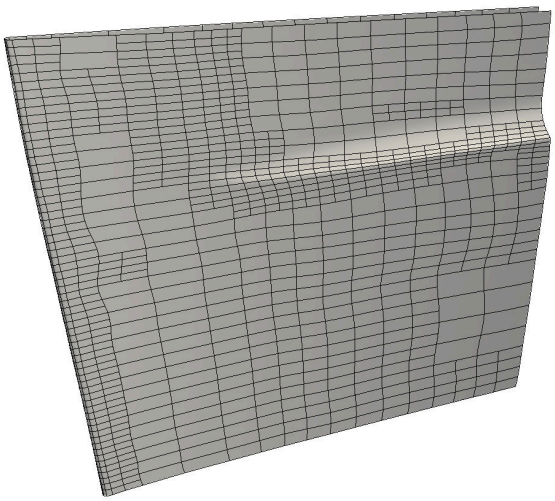
(c) A-HDM, 5 PC, DOF = 2254



(d) A-HDM, 5 PC, adm., DOF = 1841



(e) J-PDM, 1000 it, DOF = 2137



(f) J-PDM, 1000 it, adm., DOF = 1586

Figure 17: Blade geometry reconstructions from Section 6.4. Left column: standard results; right column: admissible meshes (adm.). First row: A-PDM with 5 PC steps; second row: A-HDM with error-based weights [38] and 5 PC steps; third row: J-PDM with 1000 L-BFGS iterations. The dimension of each geometric model (DOF) is also reported.

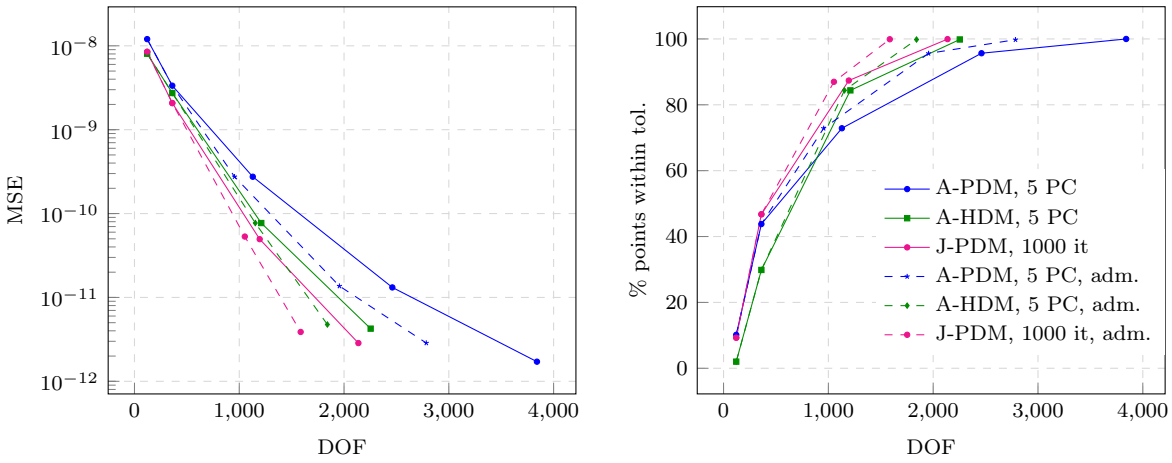


Figure 18: Blade geometry reconstructions from Section 6.4. Trends of the MSE (left) and the percentage of points below the tolerance (right) with respect to the number of DOF and different adaptive fitting schemes and standard (solid lines) and admissible (dashed lines) refinement strategies. A-PDM (blue) with 5 PC steps combined with standard (solid blue) and admissible (dashed blue) refinement routine; A-HDM (green) with error-based weights [38] and 5 PC steps combined with standard (solid green) and admissible (dashed green) refinement routine; J-PDM (magenta) with 1000 L-BFGS iterations combined with standard (solid magenta) and admissible (dashed magenta) refinement routine.

References

[1] Siemens Software: Parasolid. https://plm.sw.siemens.com/en-US/plm-components/parasolid/.

[2] Å. Björck. *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics, 1996.

[3] A. Blake and M. Isard. *Active Contours: The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion*. Springer London, 1998.

[4] P. Bo, R. Ling, and W. Wang. A revisit to fitting parametric surfaces to point clouds. *Computers & Graphics*, 36(5):534–540, 2012. Shape Modeling International (SMI) Conference 2012.

[5] C. Bracco, C. Giannelli, D. Großmann, S. Imperatore, D. Mokriš, and A. Sestini. THB-spline approximations for turbine blade design with local B-spline approximations. In *Mathematical and Computational Methods for Modelling, Approximation and Simulation*, pages 63–82. Springer, 2022.

[6] C. Bracco, C. Giannelli, D. Großmann, and A. Sestini. Adaptive fitting with THB-splines: Error analysis and industrial applications. *Computer Aided Geometric Design*, 62:239–252, 2018.

[7] C. Bracco, C. Giannelli, and R. Vázquez. Refinement Algorithms for Adaptive Isogeometric Methods with Hierarchical Splines. *Axioms*, 7(3), 2018.

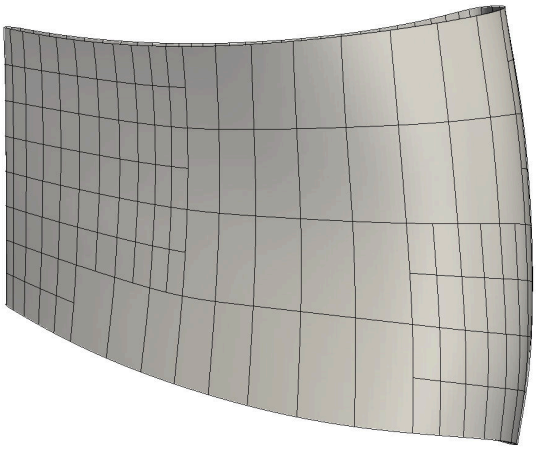
[8] C. G. Broyden. The convergence of a class of double-rank minimization algorithms 1. *Journal of the Institute of Mathematics and its Applications*, 6:76–90, 1970.

[9] A. Buffa and C. Giannelli. Adaptive isogeometric methods with hierarchical splines: Error estimator and convergence. *Mathematical Models and Methods in Applied Sciences*, 26(01):1–25, 2016.

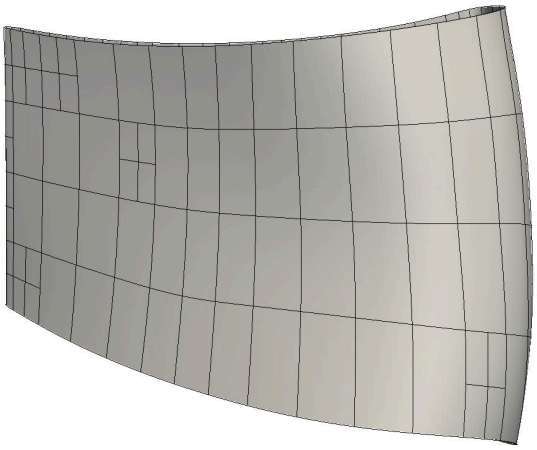
[10] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[11] P. Craven and G. Wahba. Smoothing noisy data with spline functions. *Numerische Mathematik*, 31(4):377–403, 1978.

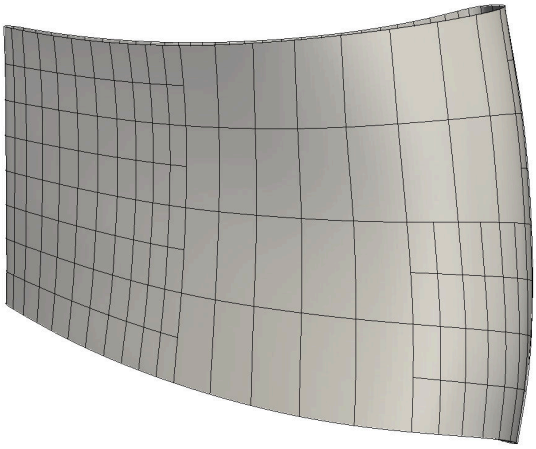
[12] C. de Boor. *A Practical Guide to Splines*. Springer New York, NY, 1978.



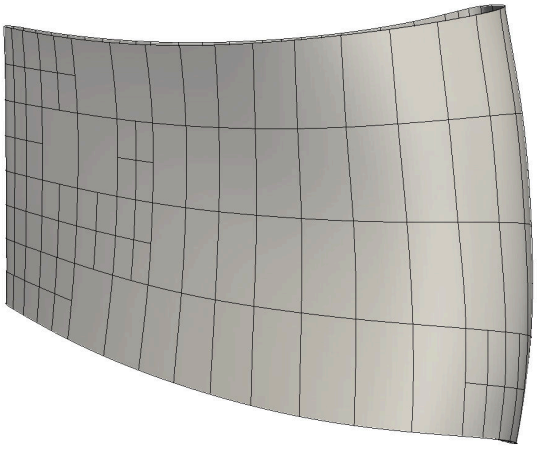
(a) A-PDM, 5PC, DOF = 566



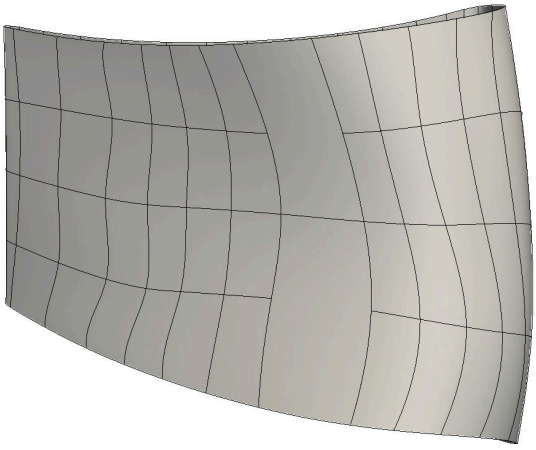
(b) A-PDM adm., 5 PC, DOF = 364



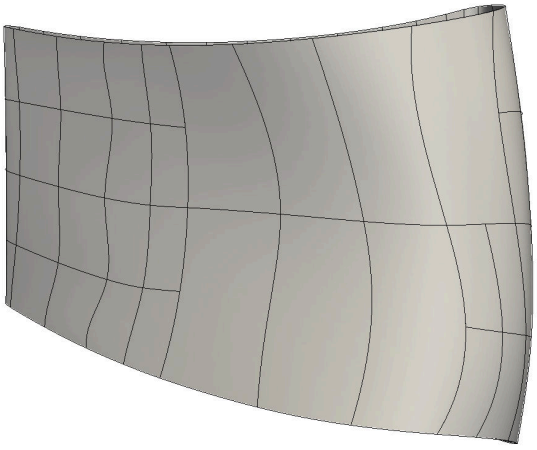
(c) J-PDM, 200 it, DOF = 599



(d) J-PDM, 200 it, adm., DOF = 369



(e) J-PDM, 1000 it, DOF = 230



(f) J-PDM, 1000 it, adm., DOF = 197

Figure 19: Airfoil geometry reconstructions from Section 6.4. Left column: standard results; right column: admissible meshes (adm.). First row: A-PDM with 5 PC steps; second row: J-PDM with 200 L-BFGS iterations; third row: J-PDM with 1000 L-BFGS iterations. The dimension of each geometric model (DOF) is also reported.

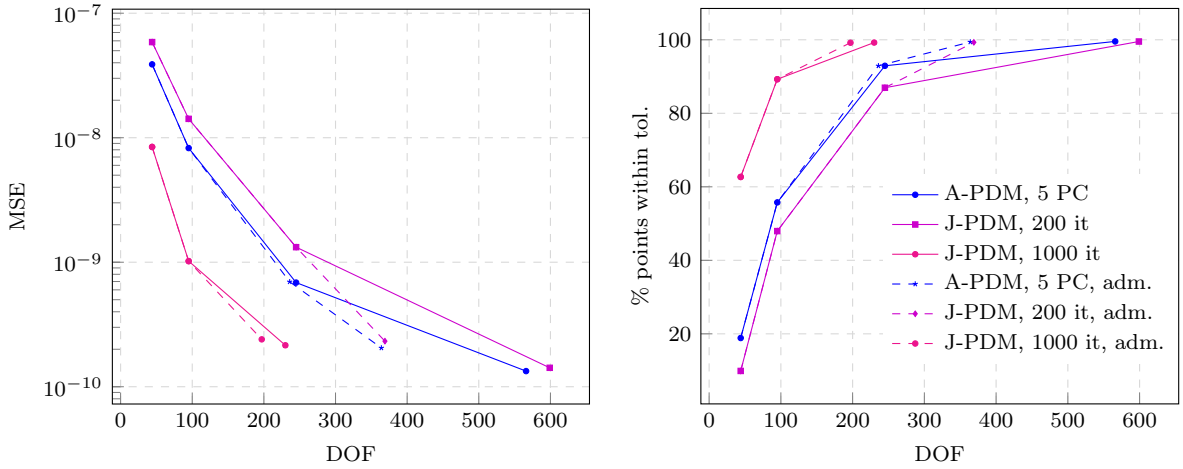


Figure 20: Airfoil geometry reconstructions from Section 6.4. Trends of the MSE (left) and the percentage of points below the tolerance (right) with respect to the number of DOF and different adaptive fitting schemes and standard (solid lines) and admissible (dashed lines) refinement strategies. A-PDM with 5 PC steps combined with standard (solid blue) and admissible (dashed blue) refinement routine; J-PDM with 200 L-BFGS iterations combined with standard (solid violet) and admissible (dashed violet) refinement routine.; J-PDM with 1000 L-BFGS iterations combined with standard (solid magenta) and admissible (dashed magenta) refinement routine.

[13] J. Deng, F. Chen, X. Li, C. Hu, W. Tong, Z. Yang, and Y. Feng. Polynomial splines over hierarchical T-meshes. *Graphical Models*, 70(4):76–86, 2008.

[14] T. Dokken, T. Lyche, and K. F. Pettersen. Polynomial splines over locally refined box-partitions. *Computer Aided Geometric Design*, 30(3):331–356, 2013.

[15] G. Farin. *Curves and surfaces for CAGD: a practical guide*. Morgan Kaufmann, 2002.

[16] M. S. Floater and M. Reimers. Meshless parameterization and surface reconstruction. *Computer Aided Geometric Design*, 18(2):77–92, 2001.

[17] D. R. Forsey and R. H. Bartels. Hierarchical B-Spline Refinement. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '88, page 205–212, New York, NY, USA, 1988. Association for Computing Machinery.

[18] C. Giannelli, S. Imperatore, A. Mantzaflaris, and D. Mokriš. Leveraging Moving Parameterization and Adaptive THB-Splines for CAD Surface Reconstruction of Aircraft Engine Components. In F. Banterle, G. Caggianese, N. Capece, U. Erra, K. Lupinetti, and G. Manfredi, editors, *Smart Tools and Applications in Graphics - Eurographics Italian Chapter Conference*. The Eurographics Association, 2023.

[19] C. Giannelli, S. Imperatore, A. Mantzaflaris, and F. Scholz. BIDGCN: boundary informed dynamic graph convolutional network for adaptive spline fitting of scattered data. *Neural Computing and Applications*, pages 1–24, 2024.

[20] C. Giannelli, B. Jüttler, S. K. Kleiss, A. Mantzaflaris, B. Simeon, and J. Špeh. THB-splines: An effective mathematical technology for adaptive refinement in geometric design and isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 299:337–365, 2016.

[21] C. Giannelli, B. Jüttler, and H. Speleers. THB-splines: The truncated basis for hierarchical splines. *Computer Aided Geometric Design*, 29(7):485–498, 2012. Geometric Modeling and Processing 2012.

[22] 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, Innovations in Applied Mathematics*. Vanderbilt University Press, Nashville, TN, 1997.

[23] G. Greiner, J. Loos, and W. Wesselink. Data dependent thin plate energy and its use in interactive surface modeling. *Computer Graphics Forum*, 15(3):175–185, 1996.

[24] C. Gu. Cross-validating non-gaussian data. *Journal of Computational and Graphical Statistics*, 1(2):169–179, 1992.

[25] J. Hoschek. Intrinsic parametrization for approximation. *Computer Aided Geometric Design*, 5(1):27–31, 1988.

[26] A. Jahanshahloo and A. Ebrahimi. Reconstruction of 3D shapes with B-spline surface using diagonal approximation BFGS methods. *Multimedia Tools and Applications*, 81:38091–38111, 2022.

[27] L. Jiang, R. H. Byrd, E. Eskow, and R. Schnabel. A preconditioned L-BFGS Algorithm with Application to Molecular Energy Minimization. Technical report, Colorado University at Boulder Department of Computer Science, 2004.

[28] G. Kermarrec, V. Skytt, and T. Dokken. *Optimal surface fitting of point clouds using local refinement: Application to GIS data*. SpringerBriefs in Earth System Sciences. Springer Nature, Cham, 2023.

[29] G. Kiss, C. Giannelli, U. Zore, B. Jüttler, D. Großmann, and J. Barner. Adaptive CAD model (re-) construction with THB-splines. *Graphical models*, 76(5):273–288, 2014.

[30] 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, 1997.

[31] D. Lenz, R. Yeh, V. Mahadevan, I. Grindeanu, and T. Peterka. Customizable adaptive regularization techniques for B-spline modeling. *Journal of Computational Science*, 71, 2023.

[32] K. Levenberg. A method for the solution of certain non–linear problems in least squares. *Quarterly of Applied Mathematics*, 2:164–168, 1944.

[33] X. Li and M. A. Scott. Analysis-suitable T-splines: Characterization, refineability, and approximation. *Mathematical Models and Methods in Applied Sciences*, 24(06):1141–1164, 2014.

[34] Y. Liu. HLBFGS: a hybrid L-BFGS routine. https://xueyuhanlang.github.io/software/HLBFGS/.

[35] Y. Liu, H. Pottmann, and W. Wang. Constrained 3D shape reconstruction using a combination of surface fitting and registration. *Computer-Aided Design*, 38(6):572–583, 2006.

[36] Y. Liu and W. Wang. A Revisit to Least Squares Orthogonal Distance Fitting of Parametric Curves and Surfaces. In F. Chen and B. Jüttler, editors, *Advances in Geometric Modeling and Processing*, pages 384–397. Springer Berlin Heidelberg, 2008.

[37] A. Mantzaflaris. An overview of geometry plus simulation modules. In D. Slamanig, E. Tsigaridas, and Z. Zafeirakopoulos, editors, *Mathematical Aspects of Computer and Information Sciences*, pages 453–456. Springer, 2020.

[38] M. Marinov and L. Kobbelt. Optimization methods for scattered data approximation with subdivision surfaces. *Graphical Models*, 67(5):452–473, 2005.

[39] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[40] S. Merchel, B. Jüttler, and D. Mokriš. Adaptive and local regularization for data fitting by tensor-product spline surfaces. *Advances in Computational Mathematics*, 49(4):58, 2023.

[41] H. P. Moreton and C. H. Séquin. Functional optimization for fair surface design. *SIGGRAPH Comput. Graph.*, 26(2):167–176, 1992.

[42] J. Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782, 1980.

[43] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer New York, 1999.

[44] L. Piegl and W. Tiller. *The NURBS Book*. Number 2 in Monographs in Visual Communication. Springer Berlin, Heidelberg, 1997.

[45] H. Pottmann and S. Leopoldseder. A concept for parametric surface fitting which avoids the parametrization problem. *Computer Aided Geometric Design*, 20(6):343–362, 2003.

[46] H. Pottmann, S. Leopoldseder, and M. Hofer. Approximation with active B-spline curves and surfaces. In *10th Pacific Conference on Computer Graphics and Applications, 2002. Proceedings.*, pages 8–25, 2002.

[47] R. Pytlak. *Conjugate gradient algorithms in nonconvex optimization*, volume 89 of *Nonconvex Optimization and Its Applications*. Springer Berlin, Heidelberg, 2008.

[48] R. Pytlak and T. Tarnawski. Preconditioned conjugate gradient algorithms for nonconvex problems. In *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*, volume 3, pages 3191–3196, 2004.

[49] E. Saux and M. Daniel. An improved Hoschek intrinsic parametrization. *Computer Aided Geometric Design*, 20(8):513–521, 2003. In memory of Professor J. Hoschek.

[50] T. W. Sederberg, D. L. Cardon, G. T. Finnigan, N. S. North, J. Zheng, and T. Lyche. T-spline simplification and local refinement. *ACM Transaction on Graphics*, 23(3):276–283, 2004.

[51] T. W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-splines and T-NURCCs. *ACM Transaction on Graphics*, 22(3):477–484, 2003.

[52] T. Speer, M. Kuppe, and J. Hoschek. Global reparametrization for curve approximation. *Computer Aided Geometric Design*, 15:869–877, 1998.

[53] G. Wahba. *Spline models for observational data*. CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM, 1990.

[54] W. Wang, H. Pottmann, and Y. Liu. Fitting B-Spline Curves to Point Clouds by Curvature-Based Squared Distance Minimization. *ACM Trans. Graph.*, 25(2):214–238, 2006.

[55] X. Wei, Y. Zhang, L. Liu, and T. J. Hughes. Truncated T-splines: Fundamentals and methods. *Computer Methods in Applied Mechanics and Engineering*, 316:349–372, 2017.

[56] P. Wolfe. Convergence Conditions for Ascent Methods. *SIAM Review*, 11(2):226–235, 1969.

[57] P. Wolfe. Convergence Conditions for Ascent Methods. II: Some Corrections. *SIAM Review*, 13(2):185–188, 1971.

[58] W. Zheng, P. Bo, Y. Liu, and W. Wang. Fast B-spline curve fitting by L-BFGS. *Computer Aided Geometric Design*, 29(7):448–462, 2012.