4. Moving parameterization

In this Chapter, we present adaptive spline model reconstruction schemes with moving parameterization, which consist of a suitable combination of parameter locations and the control points optimization with adaptive refinement routines, using THB-splines. In particular, we propose adaptive alternating and joint optimization methods to optimize the parameter locations and the control points of the (hierarchical) spline geometric model.

We start in Section 1 by analyzing existing alternating fitting schemes for tensor-product B-splines. These alternating methods optimize the parameters separately from the control point computation by iteratively inferring the footpoints of the point cloud on the current surface and subsequently updating the control points. The method involved in this second step, e. g., Point Distance Minimization (PDM), Tangent Distance Minimization (TDM) [161], Hybrid Distance Minimization (HDM) [9], or QI, gives the name to the whole fitting procedure.

In Section 2, we extend the alternating fitting scheme to the adaptive spline constructions and propose novel adaptive fitting schemes with THB-splines based on different error metrics. We compare the behaviour of different optimization settings for the critical task of distance minimization by also relating the effectiveness of the correction step to the quality of the initial parameterization.

Finally, in Section 3 we extend to adaptive spline scenarios the Joint Point Distance Minimization (J-PDM) method, based on the simultaneous joint optimization of control points and parameters, hence avoiding the solution of a linear system of equations and the computation of foot-point projection at every refinement iteration.

We apply the proposed approaches to the reconstruction of real-world data sets, also consisting of aircraft engine components from scanned point data, see Section 4. Our study reveals that using a moving parameterization instead of a fixed one, when suitably combined with the adaptive spline loop, can significantly improve the fitting results while also reducing the number of degrees of freedom required to achieve a certain accuracy. More specifically, it can lead to earlier termination of the adaptive process, thus providing more compact models with less refinement levels and outperforming state-

Sofia Imperatore, Eindhoven University of Technology, the Netherlands, s.imperatore@tue.nl, 0009-0003-9116-9978

Referee List (DOI 10.36253/fup_referee_list)
FUP Best Practice in Scholarly Publishing (DOI 10.36253/fup_best_practice)

Book References DOI 10.36253/979-12-215-1002-7.references

of-the-art hierarchical spline model reconstruction schemes. This Chapter is mostly based on [61] and [59].

4.1. Alternating fitting schemes: A-PDM, A-TDM, A-HDM, A-QI

In this Section, we consider the surface fitting problem (60) addressed at the beginning of Chapter 3, in case of B-spline constructions. Given a set of scattered points as in (1), so that $\mathcal{P} = \mathcal{P}_I \dot{\cup} \mathcal{P}_B$, i. e. disjoint union between interior and boundary points, find a spline model $\boldsymbol{s} : \Omega \subset \mathbb{R}^D \to \mathbb{R}^N$,

$$\boldsymbol{s}(\boldsymbol{x}) = \sum_{j=0}^{n} \boldsymbol{c}_j \beta_j(\boldsymbol{x}), \text{ for } \boldsymbol{x} \in \Omega$$

with B-spline basis $\{\beta_0, \ldots, \beta_n\}$, coefficients $\boldsymbol{c}_j \in \mathbb{R}^N$ for $j = 0, \ldots, n$, collected in the matrix C, and point parametric values $\mathcal{U}$ (3), collected in the matrix U, so that it satisfies

$$\underset{U,\ C}{\arg\min} \frac{1}{2} \sum_{i=1}^{m} \left\| \sum_{j=0}^{n} \boldsymbol{c}_j \beta_j(\boldsymbol{u}_i) - \boldsymbol{p}_i \right\|_2^2 + \lambda J(\mathcal{C}), \tag{60}$$

where J is a fairing term, which we choose as in (38).

Due to the complexity of this non-linear problem, standard methods usually decouple it into two smaller sub-problems, treated separately, i. e. the computation of the parameters $\mathcal{U}$ and the computation of the control points. Moreover, a *fixed* parameterization $\mathcal{U}$ is initially constructed and is often kept fixed throughout the (adaptive) fitting algorithm.

The idea that the parameterization must also be adjusted to the fitted geometry was introduced in [75], in the context of B-spline curve approximation. The so-called intrinsic parametrization, or Parameter Correction (PC) method, is (iteratively) computed starting from a certain initial approximation. At each iteration, the fitted points $\mathcal{P}$ are projected onto the current geometry, and the projected footpoints take the place of the previous parameter values. The control points of the B-spline geometry are updated by re-fitting the geometry with the updated parameter values. Note that each re-fit can then significantly improve the reconstructed geometric model. The core of the method relies on efficient footpoint projection, which is reduced to a non-linear minimization problem. The iterative approach of [75] for finding the footpoint (i. e. the closest point on the geometry) has been revisited in [142], where a Newton-like method is proposed for the problem. As in all non-linear minimization schemes, the quality of the initial points, the nature of the objective function, as well as the different stepping strategies play a crucial role in the computation.

For a fixed spline space, tackling the problem of foot-points and control points separately, brought to the development of alternating fitting methods, based on the iterative execution of the two following steps:

1. footpoint projection and parameter update;
2. control points computation.

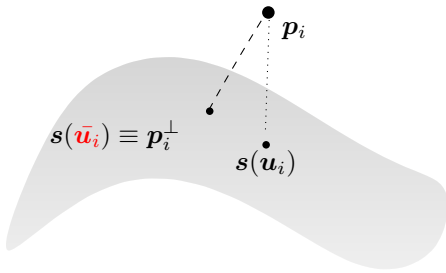


Figura 58. Foot-point projection.

Foot-point projection and parameter update

This method consists of locating the points on the geometric model that are the closest to the data points in terms of Euclidean distance. Given a point cloud $\mathcal{P}$, its parameterization $\mathcal{U}$, and a surface $\boldsymbol{s} : \Omega \subset \mathbb{R}^2 \to \mathbb{R}^3$, the surface closest point problem consists in solving the following minimization problem,

$$\min_{\boldsymbol{u}_i \in \Omega} \frac{1}{2} \left\| \boldsymbol{s}\left(\boldsymbol{u}_i\right) - \boldsymbol{p}_i \right\|_2^2, \quad \text{for each } i = 1, \ldots, m. \tag{61}$$

This two-dimensional nonlinear problem can be explicitly formulated as

$$\left(\boldsymbol{s}\left(\boldsymbol{u}_i\right) - \boldsymbol{p}_i\right)^T \nabla \boldsymbol{s}\left(\boldsymbol{u}_i\right) = 0, \text{ for each } i = 1, \ldots, m, \tag{62}$$

where $\nabla \boldsymbol{s}$ indicates the gradient of the surface $\boldsymbol{s}$, and solved by employing a suitable optimizer. In view of (62), 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 *footpoint* 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 (62), for $i = 1, \ldots, m$. Figure 58 illustrates the projection of a point $\boldsymbol{p}_i$ and its footpoint 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 footpoint 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.

Remark 22. *Since we assume the point cloud $\mathcal{P}$ to be a disjoint union between interior and boundary points, i. e. $\mathcal{P} = \mathcal{P}_I \dot{\cup} \mathcal{P}_B$, we decouple the problem in (61) accordingly. In particular, if $\boldsymbol{p}_i \in \mathcal{P}_I$, i. e. for interior points, we solve the problem in (61). On the other hand, if if $\boldsymbol{p}_i \in \mathcal{P}_B$, i. e. for boundary points, we solve*

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

where $\partial\Omega$ indicates the boundary of the parametric domain $\Omega \subset \mathbb{R}^D$. Moreover, if the boundary of the domain is composed of more than 1 curve, we further decouple the optimization problem by constraining each boundary point to be projected only on the boundary curve it belongs to.

Control points computation

Depending on the strategy employed to compute the control points, different methods have been developed.

A-PDM

Solving at each iteration

$$\min_{C} \frac{1}{2} \sum_{i=1}^{m} \left\| \sum_{j=0}^{n} \boldsymbol{c}_j \beta_j(\boldsymbol{u}_i) - \boldsymbol{p}_i \right\|_2^2 + \lambda J(\mathcal{C}), \tag{63}$$

with respect to the coefficients $C \in \mathbb{R}^{(n+1)\times N}$, leads to the formulation of the Alternating Point Distance Minimization (A-PDM) method. This method is wildly used because of its simplicity, see e. g., [127, 129, 75, 142, 74, 54]. As observed in [7], from an optimization view-point, A-PDM exhibits linear convergence.

Remark 23. *Note that the coefficients update of A-PDM, correspond to the weighted LS problem discussed in Section 1 (with $\lambda = 0$) see (37), with constant weights equal to* 1.

Consequently, the linear system of normal equations to solve at each step of A-PDM is

$$\left(B^\top B + \lambda G\right) \boldsymbol{c} = B^\top \boldsymbol{p}, \tag{64}$$

where B is the collocation matrix as in (26), G is the matrix representing the contribution of the functional J, i. e.

$$G_{i,j} = \int_\Omega \left(\frac{\partial^2 \beta_i}{\partial x \partial x} \frac{\partial^2 \beta_j}{\partial x \partial x} + 2 \frac{\partial^2 \beta_i}{\partial x \partial y} \frac{\partial^2 \beta_j}{\partial x \partial y} + \frac{\partial^2 \beta_i}{\partial y \partial y} \frac{\partial^2 \beta_j}{\partial y \partial y} \right) \mathrm{d}x\mathrm{d}y, \tag{65}$$

for $i, j = 0, \ldots, n$, $\boldsymbol{p} \in \mathbb{R}^{m\times N}$ is the matrix containing the points of $\mathcal{P}$, and $\boldsymbol{c} \in \mathbb{R}^{(n+1)\times N}$ is the matrix containing the unknown coefficients.

We now introduce a different arrangement of the linear system of equations in (64), which will be useful for the following discussions. Let $\widetilde{B} \in \mathbb{R}^{Nm\times N(n+1)}$ be the matrix, where the main-diagonal blocks B consists of the collocation matrix (26),

$$\widetilde{B} = \begin{pmatrix} B & 0 & \ldots & 0 \\ 0 & B & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & B \end{pmatrix}. \tag{66}$$

Similarly, let $\widetilde{G} \in \mathbb{R}^{N(n+1)\times N(n+1)}$ be of the form

$$\widetilde{G} = \begin{pmatrix} G & 0 & \dots & 0 \\ 0 & G & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & G \end{pmatrix}, \tag{67}$$

where the main-diagonal blocks correspond to the matrix G as in (65), representing the functional J. Finally, let $\widetilde{\boldsymbol{p}} \in \mathbb{R}^{mN}$ be the vector containing the data points ordered as follows,

$$\widetilde{\boldsymbol{p}} = \left(p_1^{(1)}, \dots, p_m^{(1)}, \dots, p_1^{(N)}, \dots, p_m^{(N)}\right)^\top \in \mathbb{R}^{mN}. \tag{68}$$

Hence, the problem in (64) can be rewritten as

$$\left(\widetilde{A}_{\text{pdm}} + \lambda \widetilde{G}\right) \boldsymbol{c} = \boldsymbol{b}_{\text{pdm}}, \tag{69}$$

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

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

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

$$\boldsymbol{c} = \left(c_0^{(1)}, \dots, c_n^{(1)}, \dots, c_0^{(N)}, \dots, c_n^{(N)}\right)^\top. \tag{70}$$

A-TDM

The A-TDM method [8, 161, 106] is characterized by the following update rule for the control points,

$$\min_C \frac{1}{2} \sum_{i=1}^{m} \left[\left(\sum_{j=0}^{n} \boldsymbol{c}_j \beta_j(\boldsymbol{u}_i) - \boldsymbol{p}_i \right)^T \cdot \boldsymbol{n}_i \right]^2 + \lambda \boldsymbol{J}(\mathcal{C}), \tag{71}$$

where

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

is the unit normal vector at the surface point $\boldsymbol{s}(\boldsymbol{u}_i)$, for each $i = 1, \dots, m$. For $N = 2, 3$, and for each $i = 1, \dots, m$ the term

$$\left[\left(\sum_{j=0}^{n} \boldsymbol{c}_j \beta_j(\boldsymbol{u}_i) - \boldsymbol{p}_i \right)^T \cdot \boldsymbol{n}_i \right]^2 = \left[(\boldsymbol{s}(\boldsymbol{u}_i) - \boldsymbol{p}_i) \cdot \boldsymbol{n}_i \right]^2,$$

defines the distance between the data point $\boldsymbol{p}_i$ and the tangent line (if $D = 1$) or plane (if $D = 2$) at $\boldsymbol{s}(\boldsymbol{u}_i)$.

In order to recover the linear system of equation to be solved at each step of the A-TDM method, for each $k = 1, \ldots, N$, let $N_k \in \mathbb{R}^{m \times m}$ be the diagonal matrix containing the normal components at each point along the k-th physical direction, i. e.

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

and let $\widetilde{N} \in \mathbb{R}^{Nm \times m}$ the block matrix, so that its blocks correspond to N_k, i. e.

$$\widetilde{N} = \begin{pmatrix} N_1 \\ N_2 \\ \vdots \\ N_N \end{pmatrix} \in \mathbb{R}^{Nm \times m}.$$

The problem in (71) 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 $\boldsymbol{c}$ as defined in (66), (67), (68) and (70), respectively. Therefore, the normal equations with respect to the unknowns $\boldsymbol{c}$ are

$$\left(\widetilde{A}_{\text{tdm}} + \lambda \widetilde{G} \right) \boldsymbol{c} = \boldsymbol{b}_{\text{tdm}}, \tag{72}$$

where $\widetilde{A}_{\text{tdm}} \in \mathbb{R}^{N(n+1) \times N(n+1)}$ and $\boldsymbol{b}_{\text{tdm}} \in \mathbb{R}^{N(n+1)}$ 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 is well known that A-TDM does not show a stable performance near high curvature regions. Moreover, it has been proven to be a Gauss-Newton minimization without a step-size control, see [161], and a regularization term needs to be added in order to improve the stability of the method. In particular, it is common to apply the Levenberg–Marquardt regularization, see [161, 170], by modifying (72) as

$$\left(\widetilde{A}_{\text{tdm}} + \gamma I + \lambda \widetilde{G} \right) \boldsymbol{c} = \boldsymbol{b}_{\text{tdm}}, \tag{73}$$

to add the term γI, where $I \in \mathbb{R}^{N(n+1) \times N(n+1)}$ is the identity matrix and $\gamma \in \mathbb{R}, \gamma \geq 0$. This A-TDM variant is usually addressed in the literature as Alternating Tangent Distance Minimization Levenberg-Marquardt (A-TDMLM).

A-HDM

By combining the A-PDM and A-TDM methods, the Alternating Hybrid Distance Minimization (A-HDM) method can be defined, see, e. g., [113, 9]. In this case, the distance metric at a sample $\boldsymbol{s}(\boldsymbol{u}_i)$ for each $i = 1, \ldots, m$ takes into account both PDM and TDM metrics, i. e.

$$\gamma_i \left\| \boldsymbol{s}(\boldsymbol{u}_i) - \boldsymbol{p}_i \right\|_2^2 + \delta_i \left[\left(\boldsymbol{s}(\boldsymbol{u}_i) - \boldsymbol{p}_i \right)^\top \cdot \boldsymbol{n}_i \right]^2, \tag{74}$$

with $\gamma_i, \delta_i \geq 0$ and $\gamma_i + \delta_i = 1$. The objective function for the control point update to minimize is then

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

Remark 24. *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 fall back to the A-TDM method.*

The authors in [113] 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 (74) results to be a weighted combination of the A-PDM and A-TDM error terms, depending on the point-wise distance

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

In order to exploit the curvature information from the point cloud $\mathcal{P}$, the authors in [9], 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 points $\boldsymbol{p}_i \in \mathcal{P}$, for $i = 1, \ldots, m$. Hence, (74) becomes a weighted combination of the A-PDM and A-TDM error terms, with weights depending also on the surface discrete curvature. In the numerical investigation presented in this Thesis, we also consider a more simple choice of the blending weights as constant non-negative values, i. e.

$$\gamma_i = \gamma \geq 0, \quad \text{and} \quad \delta_i = \delta = (1 - \gamma), \text{ for } i = 1, \ldots, m,$$

with $\gamma + \delta = 1$.

Independently from 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,

$$\left(\widetilde{A}_{\text{pdm}} + \widetilde{A}_{\text{tdm}} + \lambda \widehat{G}\right) \boldsymbol{c} = \left(\boldsymbol{b}_{\text{pdm}} + \boldsymbol{b}_{\text{tdm}}\right), \tag{75}$$

where the matrices and right-hand-sides defined in (64) and (72) include the blending weight matrices

$$\begin{aligned} &\Gamma = \text{diag}\{\gamma_1, \dots, \gamma_m\} \in \mathbb{R}^{m \times m}, \\ &\widetilde{\Gamma} = \begin{pmatrix} \Gamma & 0 & \dots & 0 \\ 0 & \Gamma & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \Gamma \end{pmatrix} \in \mathbb{R}^{Nm \times Nm}, \\ &D = \text{diag}\{\delta_1, \dots, \delta_m\} \in \mathbb{R}^{m \times m}, \end{aligned} \tag{76}$$

so that

$$\widetilde{A}_{\text{pdm}} = \widetilde{B}^\top \widetilde{\Gamma} \widetilde{B}, \text{ and } \widetilde{A}_{\text{tdm}} = \widetilde{B}^\top \widetilde{N} D \widetilde{N}^\top \widetilde{B},$$

and

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

Remark 25. *When dealing with open surfaces, to reconstruct point clouds $\mathcal{P} = \mathcal{P}_I \dot{\cup} \mathcal{P}_B$, instability might arise from the normals computed at the boundary points $\mathcal{P}_B$. To reduce this unstable behaviour, 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, $\boldsymbol{c} = \boldsymbol{c}_{\text{int}} \dot{\cup} \boldsymbol{c}_{\text{bdr}}$, where $\boldsymbol{c}_{\text{int}}$ is the solution of (75) built on the interior parameters $\mathcal{U}_I$ and points $\mathcal{P}_I$, and similarly, $\boldsymbol{c}_{\text{bdr}}$ is the solution of (64) built on the boundary parameters $\mathcal{U}_B$ and points $\mathcal{P}_B$.*

Remark 26. *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 (73), in this case the regularization term is played by the PDM matrix $\tilde{A}_{\text{pdm}}$ as in (75). Moreover, from all the numerical results of [9], 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 results more numerically stable than the A-TDMLM, hence better suited for numerical computations, in particular for complex, e. g., adaptive, constructions. For this reason, we prefer 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 numbers, being numerically more stable also of A-HDM.*

For completeness, we recall that the A-HDM method is a variant of the Alternating Standard Distance Minimization (A-SDM) method, based on the *local* quadratic approximants of the squared distance function of a surface to a point. A-SDM has been developed in [130, 132] for active contour models to fit a surface to a target shape, applied to subdivision fitting schemes in [24], and subsequently adjusted in [161] for the problem of B-spline curve fitting to points clouds and in [105] for the problem of surface fitting and registration. Further discussion on the local quadratic approximants for the squared distance function can be found in [131].

A-QI

All the presented methods are 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 method to be used for their definition. In particular, we consider an additional alternating method, i. e. the Alternating Quasi-Interpolation (A-QI), where the second step of the alternating scheme, see Section 1, consists in the solution of a QI problem, as the one described in Section 2.

Comparison of different weight choice for A-HDM

In this Section we compare the behaviour of the A-PDM and A-HDM methods for different choices of blending weights. More precisely, for A-HDM, we consider the blending weights in [113], [9], together with a suitable constant choice. We consider a gridded point cloud, shown in the middle of the bottom line of Figure 33, consisting of 3025 data sampled from a ship hull geometry. Subsequently, we parameterize the data with CEN and CHL parameterization methods [128]. We then compute a first initial approximation by performing a penalized LS fit, i. e. PDM, with tensor-product B-splines of bi-degree $\boldsymbol{d} = (2, 2)$, a 5×2 tensor-product mesh, and penalization weight $\lambda = 1\mathrm{e} - 7$. We then consider this initial approximation as starting point for the A-PDM and A-HDM schemes.

As concerns the A-HDM method with constant weights, for the shiphull point cloud parameterized with CEN, we perform a fine-tuning search on the values of γ, with $\delta = (1 - \gamma)$. More precisely, we execute the A-HDM method for $\gamma = 1\mathrm{e} - 3, 1\mathrm{e} - 2, 1\mathrm{e} - 1, 2.5\mathrm{e} - 1, 5\mathrm{e} - 1, 7.5\mathrm{e} - 1$ and show the results in Figure 59 in terms of MSE evolution over the number of alternating iterations (10 on the left and 30 on the right). From this fine-tuning, we choose to set $\gamma = 1\mathrm{e} - 2$ (the violet line in the plots) as constant choice in view of the higher approximation accuracy achieved in this case.

Figure 60 shows the comparison of different weight choices for the A-HDM with A-PDM, performed with 10 (left) and 30 (right) iterations, for the approximation of the point cloud with initial CEN parameterization. We can observe that the A-HDM method, for any choice of blending weights, improves the approximation accuracy in terms of MSE with respect to the A-PDM method by about 1 order of magnitude. Note also that for the first 10 iterations, the curvature-based blending weights (teal line in the plots) show the worst approximation behaviour, among the other weight

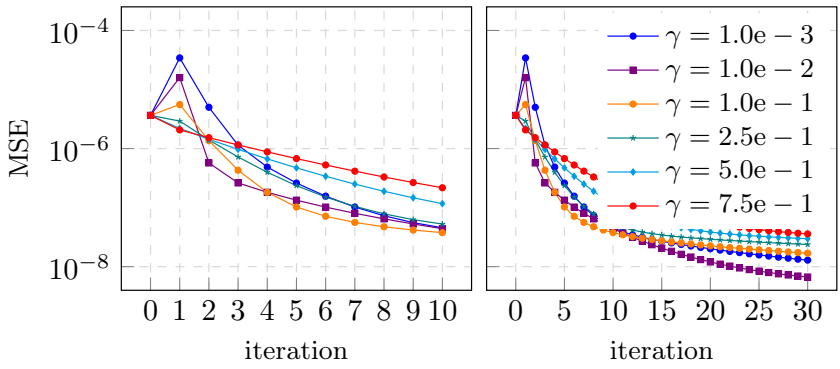


Figura 59. Shiphull point cloud with 3025 gridded points initially parametrized with standard CEN method. Alternating spline approximation for bi-degree $\boldsymbol{d} = (2, 2)$, on a 5×2 tensor-product mesh. Comparison of MSE for different choice of constant weights in the A-HDM method for 10 (left) and 30 (right) iterations.

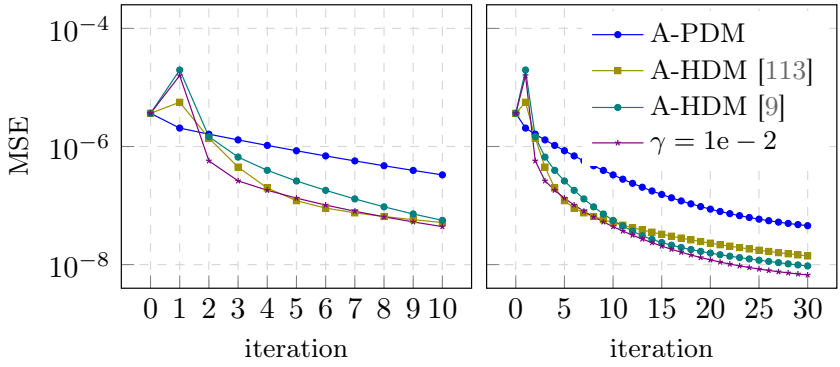


Figura 60. Alternating B-spline fitting of the shiphull point cloud with 3025 gridded data points for bi-degree $\boldsymbol{d} = (2, 2)$, on a 5×2 tensor-product mesh, with initial CEN parameterization. The comparison of MSE between A-PDM, A-HDM with constant choice of the weights, A-HDM with the weights of [113] and A-HDM with the weights of [9] is shown for 10 (left) and 30 (right) iterations.

choices. Nevertheless, if we let the algorithm perform a sufficient number of iterations, e. g., 30, then they show better behaviour with respect to the error-driven weights of [113] (olive line in the plots) and achieve the same accuracy as the constant fine-tuned weights (violet line in the plots).

Analogous results can be observed in Figure 61 for the approximation of the ship hull point cloud initially parameterized with CHL. In particular, the curvature error-based methods show better performance if we let the alternating fitting algorithm run for a sufficient number of iterations.

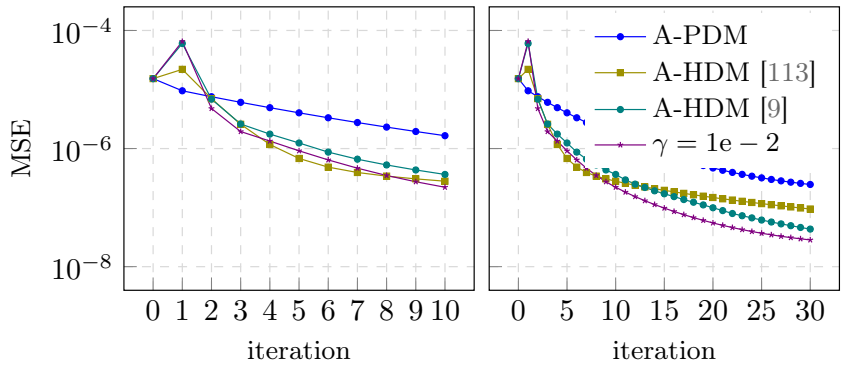


Figura 61. Alternating B-spline fitting of the shiphull point cloud with 3025 gridded data points for bi-degree $\boldsymbol{d} = (2, 2)$, on a 5×2 tensor-product mesh, with initial CHL parameterization. The comparison of MSE between A-PDM, A-HDM with constant choice of the weights, A-HDM with the weights of [113] and A-HDM with the weights of [9] is shown for 10 (left) and 30 (right) iterations.

4.2. Adaptive alternating fitting schemes

In this Section, we extend the presented methods to adaptive THB-spline fitting frameworks, motivated by the idea that, as the refinement proceeds in an adaptive setting, not only the geometric hierarchical model but also the parameter values of each data point should be optimized. In fact, when adaptively fitting a parametric geometric model, there is no evidence that the initial (possible) optimality of the parameterization is maintained when the approximation space is iteratively refined. Note, however, that the optimality of the initial parameterization is fundamental for the success of the proposed method, as discussed in Section 12.

In the adaptive THB-spline fitting scenario, once an initial parameterization and mesh configuration are chosen, the adaptive fitting technique consists of four fundamental steps described in (4) that are successively repeated: 1. SOLVE: fitting the THB-spline approximation on the current (hierarchical) mesh; 2. ESTIMATE: error estimation; 3. MARK: mesh marking strategy; 4. REFINE: mesh refinement strategy to suitably identify the new hierarchical mesh to be used in the next iteration, which starts again from 1.

To address the parameter update within the adaptive fitting loop, we propose exploiting the alternating fitting methods covered in Section 1 in the first step of the adaptive scheme, i. e. SOLVE. This choice enables the definition of adaptive THB-spline fitting with moving parameters.

At the beginning of any adaptive iteration, the SOLVE routine is performed as follows. Compute the control points according to the chosen method, e. g., PDM, TDM, HDM, or QI; perform a certain number of PC steps, i. e.

1. solve (62) for $\boldsymbol{s}$ belonging to the current hierarchical space, hence projecting the points of $\mathcal{P}$ on the THB-spline surface $\boldsymbol{s}$, and considering their foot-point as the new parameters $\mathcal{U}$;

2. update the geometric THB-spline model by fitting again the surface $\boldsymbol{s}$ to the points $\mathcal{P}$ with the corrected parameters.

After the conclusion of the PC steps, the new parameters and updated THB-spline geometry are considered for the next iteration of the adaptive loop. Note that the a certain number of PC-like is applied globally to the entire approximant. In particular, in the A-QI scheme, for each local problem the parameters $\mathcal{U}$ are kept fixed.

As for standard adaptive fitting strategies, the described loop is performed until the maximum point-wise error is within an input tolerance ϵ or a maximum number of hierarchical levels L is reached. Note that PC steps should naturally be embedded in any adaptive scheme, i. e. not limited to fitting problems, since even if the starting parameterization benefits from optimality within the initial space, there is no evidence that optimality is maintained when moving to wider successively refined approximation spaces.

The pseudocode for the adaptive A-HDM fitting scheme is reported in Algorithm 7. The algorithm for the adaptive A-PDM and A-TDM methods can be recovered by choosing δ_i and γ_i for $i = 1, \dots, m$ on line 3 as in Remark 24. Finally, the algorithm for the adaptive A-QI can be inferred by suitably adding the PC routine (lines 4–6 of Algorithm 7) after line 3 of Algorithm 4.

Interaction of foot-point projection with adaptive spline configurations

Finding successfully the foot-point projection of a point on a surface is a challenging open problem; see, e. g., [86] and references therein. As far as parameter correction is concerned, the foot-point projection in [75] is performed iteratively by linearizing the problem in (62). Subsequently, in [142], the iterative procedure has been replaced by the application of a Newton-like approach. Due to the locality of Newton-like gradient methods, to properly correct the parameters, it is of fundamental importance to start with a good initial parameterization guess as well as with a reasonably good geometric model.

Figure 62 shows the consequences of incorporating the PC routine at a too-early stage of an adaptive fitting scheme. More specifically, we consider the point cloud of Section 3, consisting of 9281 scattered data acquired from an industrial *tensile* part, which is approximately $2.5\text{e}-2$ m long. We fit this data with the adaptive LS scheme developed in [85], equipped with the PC routine. Note that combining the PC with the adaptive LS fitting scheme results in the adaptive extension of the A-PDM method.

An initial tensor-product B-spline approximation of bi-degree $\boldsymbol{d} = (2, 2)$ is built on a *coarse* 4×4 tensor-product mesh and illustrated in Figure 62 (a). This initial coarse mesh is then adaptively refined using a criterion of error threshold with $\epsilon = 5\text{e}-5$ m, leading to a final THB-spline model with

Algorithm 7: Adaptive alternating fitting scheme.

Input: Point cloud and initial parameterization $\{\boldsymbol{u}_i, \boldsymbol{p}_i\}_{i=1}^m$, an initial tensor-product B-spline space V^0, the error tolerance $\epsilon > 0$, a maximum number of hierarchical levels L, and a maximum number of PC steps $c_{\max}$.

Compute the initial tensor-product B-spline approximation $\boldsymbol{s} \in V^0$ and the point-wise errors $e_i = \|\boldsymbol{s}(\boldsymbol{u}_i) - \boldsymbol{p}_i\|_2$, for each $i = 1, \dots, m$ and set loop $= 0$

while $\max_i e_i > \epsilon$ *and* loop $<$ L **do**

  Solve the least squares problem

$$\boldsymbol{s} = \arg\min_{\boldsymbol{v} \in V} \frac{1}{2} \sum_{i=1}^{m} \delta_i \left\| \boldsymbol{s}\left(\boldsymbol{u}_i\right) - \boldsymbol{p}_i \right\|_2^2 + \gamma_i \left[\left(\boldsymbol{s}\left(\boldsymbol{u}_i\right) - \boldsymbol{p}_i \right)^\top \cdot \boldsymbol{n}_i \right].$$

  for step $= 1, \dots, c_{\max}$ **do**

    Compute the foot-point projections

$$\min_{\boldsymbol{u}_i \in \Omega} \frac{1}{2} \left\| \boldsymbol{s}\left(\boldsymbol{u}_i\right) - \boldsymbol{p}_i \right\|_2^2, \text{ for each } i = 1, \dots, m$$

    and update the parameters and control points of the spline geometry.

  end

  Compute the errors $e_i = \|\boldsymbol{s}(\boldsymbol{u}_i) - \boldsymbol{p}_i\|_2$, for $i = 1, \dots, m$.

  Mark the domain elements where $e_i > \epsilon$.

  Refine the marked cells and the two surrounding rings of cells.

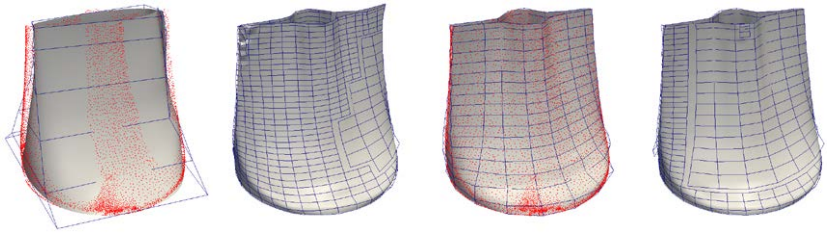
  Update the hierarchical mesh $\mathcal{M}$ and hierarchical space V.

  Set loop $=$ loop $+ 1$.

end

Output: $\boldsymbol{s} \in V$, the THB-spline approximant.

1136 DOFs that registers a MAX error of $8.33\text{e}-5$ m. The THB-spline approximation obtained by introducing 1 PC step at each iteration of the adaptive loop is shown in Figure 62 (b) and is characterized by 969 DOFs, MAX error of $3.059\text{e}-4$ m. Consequently, we may note that using the inaccurate approximation shown in Figure 62 (a) to compute the foot-point projections and correct the parameters leads to a final fitting result that is not suitable for further processing due to the self-intersections visible in Figure 62 (b). We then consider as an initial spline configuration the tensor-product B-spline approximation built on a tensor-product mesh with two additional dyadic refinements with respect to Figure 62 (a), i. e. with a 16×16 mesh, as shown in Figure 62 (c). These settings lead to stable foot-point projection due to the quality of the initial approximation, and the LS scheme with moving parameters results in a model with 607 DOFs and MAX error $8.467\text{e}-5$ m, shown in Figure 62 (d).



(a) coarse initial B-spline model (b) THB-spline model (969 DOFs) (c) refined initial B-spline model (d) THB-spline model (607 DOFs)

Figura 62. Adaptive A-PDM with THB-splines: effect of the initial tensor-product B-spline approximation. The THB-spline model (b) constructed from the initial configuration (a) defined on a 4×4 tensor-product mesh is affected by self-intersections due to the too coarse initial mesh. The THB-spline model (d) constructed from the initial configuration (c) defined on a 16×16 tensor-product mesh is successfully computed.

The choice of the optimizer and its settings are fundamental for the successful computation of the foot-point projections. Here, we consider the LS adaptive fitting method with moving parameters and compare the results with respect to the use of different optimizers and optimization settings. In particular, we start with the configuration of Figure 62 (c), i. e. bi-degree $\boldsymbol{d} = (2, 2)$ and a 16×16 tensor-product mesh, and apply 1 PC step at each adaptive iteration. For better comparison, we also fix the total number of adaptive refinement steps. The results are shown in Table 1. As a reference, we consider the commercial library Parasolid [148]. We can see that taking a too-big minimum step s in the Gradient Descent (GD) method leads to a final result that can be *worse* than not moving the parameters at all, with respect to the number of DOFs of the final models and its accuracy in terms of MAX and MSE. However, if carefully fine-tuned, the results obtained by Parasolid in terms of DOFs, MAX, and MSE can be reached for $s = 1\mathrm{e} - 12$. In the following numerical examples, we employed the Hybrid Limited memory Broyden-Fletcher-Goldfarb-Shanno (HLBFGS) method [107] with minimum step length $s = 1\mathrm{e} - 9$, since it has always shown to provide extremely similar results to Parasolid in all the considered examples. The results obtained employing HLBFGS for the above-mentioned configuration are also shown in Table 1.

Numerical examples for adaptive alternating methods

In this Section, we present a selection of numerical examples to assess the benefits of the moving parameterization approach in combination with the adaptive PDM fitting scheme. In particular, we numerically prove that including the PC routine on a proper initial guess and choosing suitable settings for the optimizer leads to outperforming the state-of-the-art THB-spline fitting schemes, see Section 12. In addition, we combine the presented adaptive A-PDM fitting methods with the parameterization methods presented in Chapter 3 to further improve the accuracy of the

Tabella 1. Influence of optimizers used for the foot-point computation loop starting from configuration of Figure 62 (c).

method	pts $< \epsilon$	MAX (m)	MSE (m^2)	DOFs
no PC	92.77%	$1.97e-4$	$7.30e-10$	751
GD, $s = 1e-9$	91.91%	$2.92e-4$	$1.06e-9$	748
GD, $s = 1e-10$	96.88%	$1.62e-4$	$4.43e-10$	690
GD, $s = 1e-11$	99.29%	$8.47e-5$	$2.41e-10$	607
GD, $s = 1e-12$	99.31%	$8.47e-5$	$2.37e-10$	607
HLBFGS	99.31%	$8.47e-5$	$2.37e-10$	607
Parasolid	99.31%	$8.47e-5$	$2.37e-10$	607

final reconstructed geometry, see Section 12 and Section 12. Besides the strength of moving the parameters accordingly to the designed space, we show that finding an optimal parameterization for the initial settings plays a fundamental role, see Section 12. In Section 12, we combine the moving parameterization technique with the parameterization method in Section 3 and the (reweighted) fitting scheme proposed in Section 7. Finally, in Section 12 we conduct a comparison between the adaptive A-PDM and A-HDM.

Adaptive A-PDM

In the following example, we analyze the effects of enriching the adaptive LS fitting scheme with the moving parameterization method. In particular, we approximate a point cloud of 9636 scattered data in $\mathbb{R}^3$, illustrated in Figure 63 (c), obtained by sampling a ship hull B-spline geometry according to the uniform random distribution.

In this example, we also store the exact sampling parametric values in $[0,1]^2$, in order to exploit the properties of the method and reduce the influence of the initial parametrization. On the contrary, in all the following numerical results, we assume the parametric values to be unknown, and we will consequently compute them, showing in addition how a good initial parameterization is fundamental.

By performing the adaptive LS fitting algorithm proposed in [85] with bi-degree $\boldsymbol{d} = (2,2)$, the final reconstructed geometry has 7173 DOFs, i. e. 74% of the points, and the THB-spline model accuracy is characterized by a MSE equal to $5.54e-5$. By enriching the previous method with the moving parameter schemes, i. e. performing the adaptive A-PDM method, the final geometry is characterized by 4277 degrees of freedom, i. e. 44% of the points, and a MSE of $4.25e-6$. Thus, introducing the PC scheme inside the adaptive loop results in a considerably smaller and more accurate THB-spline model, with about 40% of DOFs less. The final THB-spline models and the relative hierarchical meshes are illustrated in Figure 64.

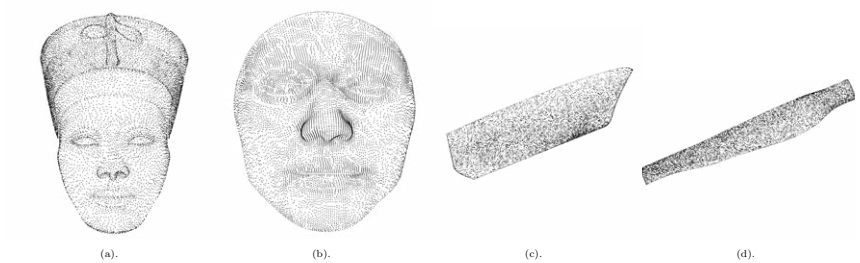


Figura 63. Nefertiti (a), face (b), ship hull (c) and wind turbine (d) scattered point clouds characterized by 8818 (a), 9283 (b), 9636 (c-d) points.

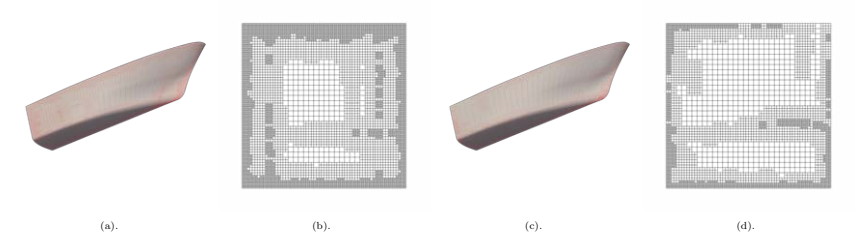


Figura 64. The hierarchical THB-spline models (a-c) and the corresponding hierarchical meshes (b-d) for the ship hull point cloud of dimension 9636 obtained with adaptive LS (a-b) and adaptive A-PDM (c-d) in Section 12.

Experiment

In this experiment, we exploit the performance of the PARCNN parameterization model, introduced in Section 1, with the adaptive A-PDM fitting scheme. We consider the gridded point cloud consisting of 3025 items, shown in Figure 33 (bottom line, center) and we compute its initial parameterization with the PARCNN model. Subsequently, we perform the A-PDM fitting scheme. The resulting hierarchical mesh and the THB-spline approximation are shown in Figure 65. In particular, the final geometry has 1349 DOFs and an accuracy characterized by a MSE equals to $2.02e-11$. Note that the degrees of freedom are properly distributed along the parametric domain Ω, by following the sharp feature and high curvature regions of the geometry, as shown in Figure 65.

Adaptive A-PDM with BIDGCN

In this example, we investigate the generalization capabilities of the BIDGCN parameterization, proposed in Section 3, to spline configurations of different bi-degrees for the reconstruction of different point clouds.

We start by performing the adaptive A-PDM fitting scheme with THB-splines of a human face model of 9283 points, shown in Figure 63 (b).

In particular, we reconstruct the THB-spline models of the input point cloud by performing the adaptive A-PDM fitting scheme both for bidegree

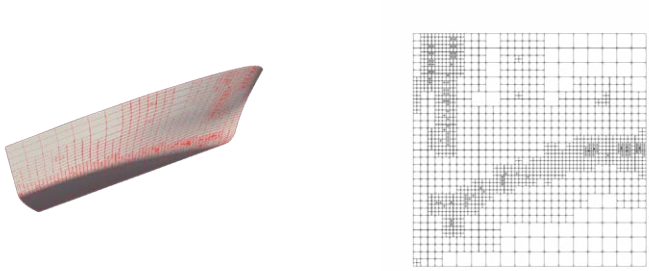


Figura 65. Reconstructed geometric spline model (left) and hierarchical mesh (right) of the ship hull point cloud of dimension 3025 via the adaptive scheme with THB-splines in Section 12.

$(3,3)$ and $(4,4)$. In the first case, we start from a tensor-product space with bi-degree $\boldsymbol{d} = (3,3)$ and perfom 8 adaptive iterations, each of them characterized by 1 step of PC. The smoothig weight λ is set to 1e−6 and the refinement threshold is $\epsilon = 8\text{e}{-4}$. The final reconstructed geometry has 4 uniformly refined and 4 locally refined hierarchical levels. Moreover, it has 2687 degrees of freedom (29% of the number of points) that approximate the input point cloud registering a MSE of 1.45e−8, a MAX error of 1.03e−3, and the DHD from the point cloud to the surface is 3.43e−2. In the second case, the algorithm settings are the same of the previous configuration, except for the bi-degree, which is not set to $\boldsymbol{d} = (4,4)$, and the refinement threshold, which is lowered to 6.5e−4. The final reconstructed model is a THB-spline geometry with 4 uniformly refined and 4 locally refined hierarchical levels, and 2793 degrees of freedom (30% of the number of points) that approximates the input point cloud with a MSE of 1.44e−8, a maximum error of 1.07e−3, and the DHD from the point cloud to the surface is 3.23e−2. The reconstructed THB-spline models, their scaled error distributions on the point cloud, the computed parametric values, and the corresponding hierarchical meshes are illustrated in Figure 66.

We conclude this example by considering two additional point clouds of 9636 items collected from a ship hull and a wind turbine, shown in Figure 63 (c) and (d). In particular, we approximate the two scattered data sets by performing 8 iterations of the adaptive A-PDM fitting algorithm, starting from a tensor-product polynomial space with bi-degree $\boldsymbol{d} = (5,2)$. The point clouds, the reconstructed THB-spline model, the point wise error distribution on the scattered data and the hierarchical meshes are displayed in Figure 67 for both the ship hull (top) and the wind turbine (bottom).

As concerns the ship hull data, we set the smoothig weight $\lambda = 1\text{e}{-6}$ and the refinement threshold is $\epsilon = 1\text{e}{-5}$. The final THB-spline model has 5 uniformly refined and 3 locally refined hierarchical levels. More specifically, it has 4092 degrees of freedom (i.e. 42% of the number of points), a MSE equal to 1.30e−10, a MAX error equal to 2.27e−4, and registers a DHD from the points to the surface equals to 8.45e−2.

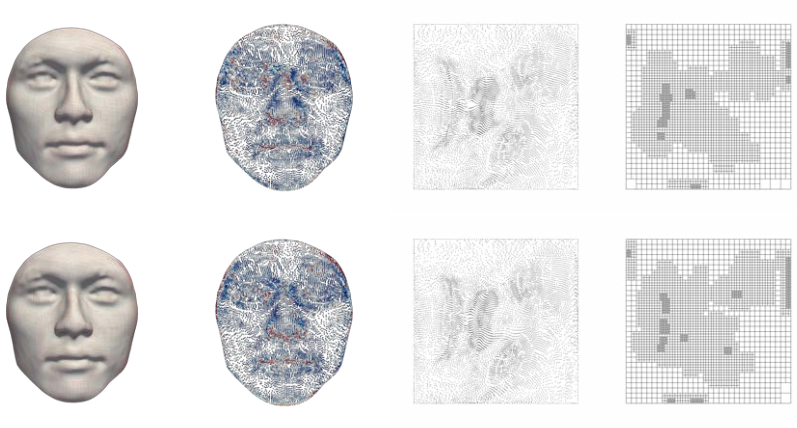


Figura 66. Hierarchical spline model reconstruction of the face point cloud shown in Figure 63 (b) for bidegree $\boldsymbol{d} = (3, 3)$ (top) and $\boldsymbol{d} = (4, 4)$ (bottom) using the BIDGCN parameterization. The reconstructed THB-spline models are shown together with the scaled error distributions on the point cloud, the computed parametric values, and the corresponding hierarchical meshes (from left to right) for Section 12.

Finally, for the wind turbine point cloud we set the smoothig weight as $\lambda = 1\text{e}{-6}$ and the refinement threshold as $\epsilon = 5\text{e}{-5}$. The final THB-spline model has 5 uniformly refined and 3 locally refined hierarchical levels, 2449 degrees of freedom, equivalent in magnitude to 25% of the number of input data, a MSE of 2.13e−10, a MAX error of 4.57e−4, and DHD from the points to the surface of 1.66e−1.

Comparison between adaptive A-PDM with LPSP and BIDGCN

In this example, we process the point cloud representing a Nefertiti face model, of size 8818, shown in Figure 63 (a) to perform a comparison between the standard parameterization methods described in Section 2 and the learning BIDGCN parameterization method proposed in Section 3. As concerns the standard parameterization methods, a suitable choice of the radius r needs to be selected. To produce fair comparisons and avoid unreasonable parameter value distributions, as already illustrated in Figure 56, we execute a heuristic search for $r = \frac{3}{\sqrt{m}}, 0.05, 0.075, 0.1, \ldots, 0.25, 0.275, 0.3$, by computing the polynomial approximation of bi-degree $\boldsymbol{d} = (2, 2)$, and select the radius r giving the best MSE. By analyzing the value of the error with respect to r, we decide to fix the radius as defined in (59), i. e. $r = \frac{3}{\sqrt{8818}}$, since it leads to the best approximation result. We then parameterize the input scattered point cloud with the standard LPSP method as well as the learning BIDGCN method and subsequently reconstruct hierarchical spline model by performing the adaptive A-PDM fitting scheme for bi-degree $\boldsymbol{d} = (2, 2)$. Note that among all the listed values for the radius r of the LPSP method that we tested on this point cloud, the choice $r = \frac{3}{\sqrt{m}}$ gives

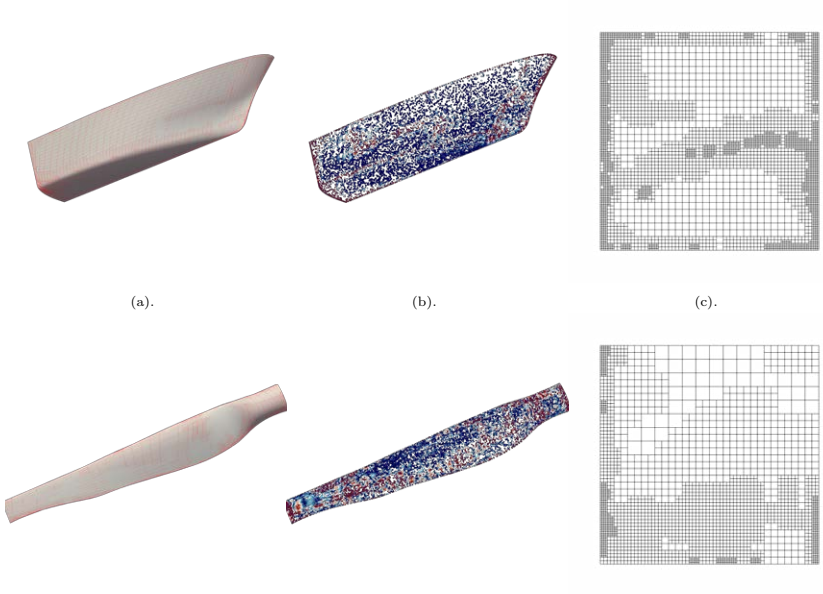


Figura 67. The hierarchical spline model reconstruction, the scaled error distribution and the hierarchical mesh for a ship hull (top) and a wind turbine (bottom) using the BIDGCN parameterization with bi-degree $\boldsymbol{d} = (5, 2)$ in Section 12.

the best results in terms of biquadratic polynomial approximation. We start from a tensor-product space with bi-degree $\boldsymbol{d} = (2, 2)$ and perform 8 adaptive iterations, each of them characterized by 1 step of PC. To properly compare the results, the refinement tolerances are chosen such that the final THB-spline models have (almost) the same number of degrees of freedom. Both BIDGCN and LPSP are characterized by 4 uniformly refined and 4 locally refined hierarchical levels. Moreover, when the LPSP method is considered, we obtain a THB-spline model with 5672 degrees of freedom, which is 64% of the total number of scattered input items, characterized by a MSE of 5.50e−6, a MAX error of 3.34e−2, and DHD from the points to the surface of 1.22. If we parameterize the Nefertiti data with the BIDGCN method, the THB-spline approximation has 5404 degrees of freedom, i.e. 61% of the number of points, and a corresponding MSE of 1.75e−6, MAX error of 1.41e−2, and DHD from the data points to the surface of 3.46e−1. The THB-spline approximations, scaled error distributions on the point cloud, parameter values, and hierarchical meshes obtained with LPSP and BIDGCN parameterizations are shown on the top and bottom of Figure 68, respectively. We observe that BIDGCN leads to better results in terms of final model accuracy, since in average a smaller error is registered and the sharp features are better reproduced. In particular, close to the mouth of the model, the shape preserving parametric values tend to be clustered, a behaviour that clearly affects the final quality of the adaptive spline

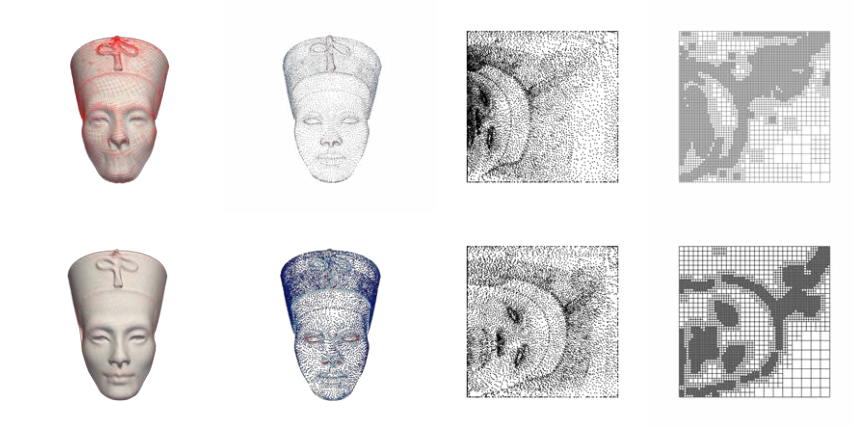


Figura 68. Hierarchical spline model reconstruction of the Nefertiti point cloud shown on left of Figure 63 for LPSP parameterization with 5492 degrees of freedom (top), as well al for the BIDGCN parameterization with 5288 degrees of freedom (bottom) with bi-degree $\boldsymbol{d} = (2, 2)$. The reconstructed THB-spline models are shown together with the scaled error distributions on the point cloud, the computed parametric values, and the corresponding hierarchical meshes (from left to right).

approximation.

Adaptive A-PDM with type II markers and BIDGCN

In this example, we perform the reweighted adaptive spline least squares fitting scheme proposed in Section 7, enriched by the moving parameterization technique, to approximate a scattered point cloud of 9000 points representing a Nefertiti bust, illustrated in Figure 69 (top-left). Firstly, we parameterize the input data with the BIDGCN method proposed in Section 3. In addition, to tackle the presence of corrupted data, we exploit the use of the weights associated to the data. More precisely, together with the points, we are given a set of markers of the second type K_{II}, highlighted in black in Figure 69 (bottom-left) which we use to perform the reweighted adaptive scheme. Due to the complexity of the problem, we decide not to update the markers K_{II} but to keep them fixed over all the iterative scheme, by setting tol_{II} sufficiently small. We then compare the results obtained by the proposed rWLS with the ordinary adaptive least squares scheme LS, where all the weights set to one. Note that both schemes are enriched by the moving parametrization technique, included within the refinement procedure. Such a comparison is illustrated in Figure 69, where we show on the top the hierarchical approximation obtained by LS and on the bottom the one obtained by rWLS. In particular, the region corresponding to the corrupted data is better approximated by the rWLS method, which results free of self-intersections and it is smoother than the one obtained with LS. Finally, the MAX error registered by LS is $3.28\text{e}-2$, whereas the MAX of rWLS is $1.57\text{e}-2$.

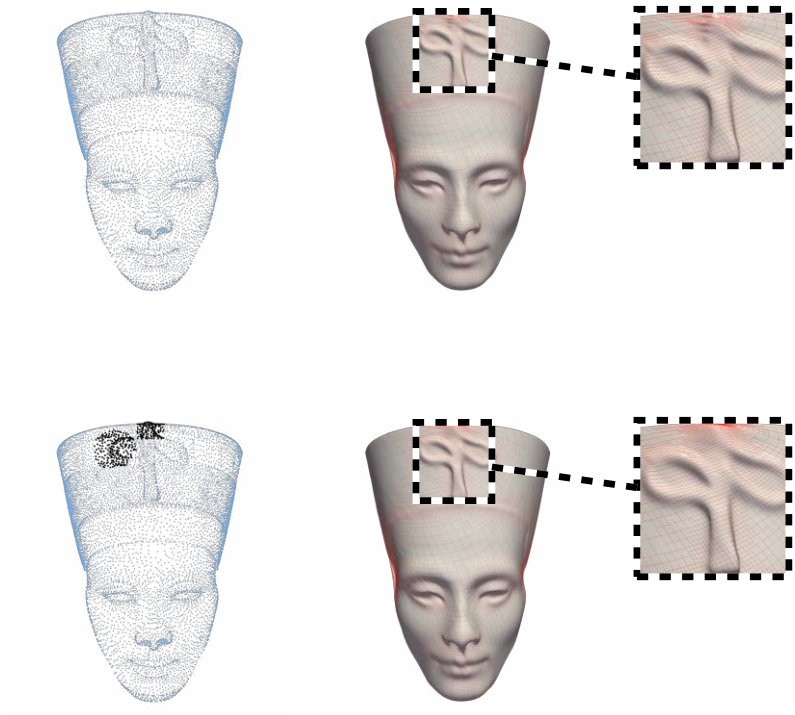


Figura 69. Adaptive surface fitting experiment described in Section 12 using adaptive LS (top) and rWLS (bottom) for data with markers of type II. The data point clouds and the marked points are also shown.

Comparison of A-PDM, A-HDM

In this last example, we analyze the behaviour of the adaptive A-PDM and adaptive A-HDM with two different choices of blending weights, the weights proposed in [9], and the constant choice. We adaptively approximate the ship hull point cloud already considered in Section 1, consisting of 3025 gridded data points and initial CEN parameterization.

As initial settings, we choose the polynomial bi-degree $\boldsymbol{d} = (2, 2)$, the initial 5×2 tensor-product mesh, smoothing weight $\lambda = 1\mathrm{e} - 7$, and we set the refinement threshold as $\epsilon = 1\mathrm{e} - 4$. Moreover, based on the weights investigations conducted in Section 1, we select the constant weight choice $\gamma = 1\mathrm{e} - 2$. We run the adaptive alternating fitting methods for 1 and 5 PC steps performed at each adaptive loop. The results in terms of MSE are shown in Figure 70.

We can observe that the A-HDM method with constant weight choice (orange line) shows better approximation performance with respect to A-PDM, whereas this is not the case for the A-HDM method with curvature-based weights as in [9], for 1 PC step. This is in line with the results

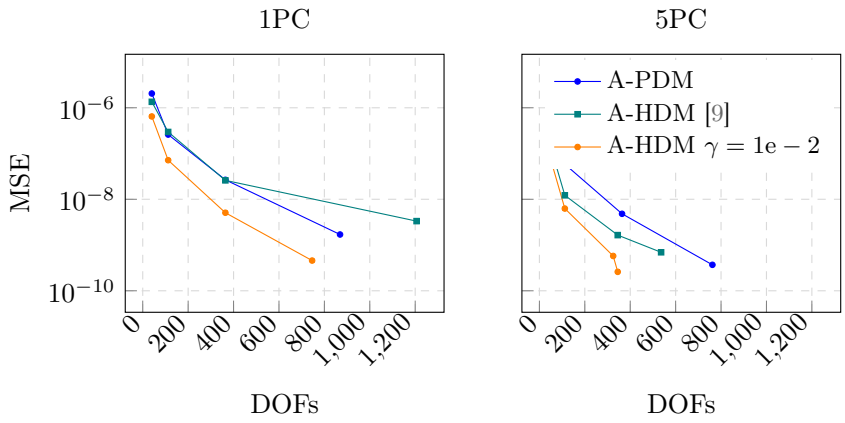


Figura 70. Adaptive spline approximation for the ship hull point cloud consisting of 3025 gridded data points parametrized with CEN method. Comparison of MSE between adaptive A-PDM (blue), A-HDM with the weights as in [9] (teal) and A-HDM with constant choice of the weights, for $\gamma = 1e-2$ (orange), for 1PC (left) and 5PC (right) steps within the adaptive loop.

presented in Section 1, specifically in Figure 60. After 1 PC step, the A-HDM methods tend not to be in a convergence regime, hence initiating the refinement on a bad approximation leads to a worse final result; see also the considerations in Section 12.

We finally run all the adaptive alternating algorithms with 5 PC steps. In this case, also adaptive A-HDM with curvature-based weights shows good approximation performance, better than A-PDM. Note that the accuracy gain is kept for all the weight choices during the adaptive refinement. In particular, the constant choice of the weights results in the best choice for this example. The final geometries obtained with 5 PC steps are in Figure 71, together with their respective number of DOFs and MSE values. More specifically, the least number of DOFs and the lowest MSE are registered by A-HDM with constant weights. When using curvature-based weights, A-HDM requires around 35% more DOFs to attain the same level of accuracy, whereas A-PDM requires about 55% more DOFs to achieve the same level of precision, yet registering a higher MSE.

4.3. Adaptive fitting with joint-optimization

Moving parameterization methods consider 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 the previous Sections is linear with respect to the control points 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

A-PDM
DOFs = 762
MSE = 3.70e − 10

A-HDM [9]
DOFs = 536
MSE = 6.94e − 10

A-HDM $\gamma = 1\text{e} - 2$
DOFs = 345
MSE = 2.59e − 10

Figura 71. Adaptive spline approximation for the ship hull point cloud consisting of 3025 gridded data points and initial CEN parameterization in Section 10. From left to right: final THB-spline model with adaptive A-PDM, A-HDM and the weights choice in [9], and A-HDM and constant weights $\gamma = 1\text{e} - 2$, with 5PC steps within each adaptive loop. The DOFs and MSE are alro reported.

Joint Optimization approach to solve simultaneously the parameterization and fitting problems for a fixed spline space, i. e. to address the problem in (60). Subsequently, we extend this surface fitting optimization method to the adaptive spline framework. More precisely, we address the parameterization problem within the first step (SOLVE) of the adaptive loop by *simultaneously* computing the optimal parameter sites $\mathcal{U}$ for $\mathcal{P}$ and control points for $\boldsymbol{s}$. Therefore, with this method, we avoid solving a linear system of equations and performing PC routines at every adaptive iteration.

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 B-spline basis functions, see Chapter 1 and the references therein. In particular, the explicit computation of the Jacobian matrix enables the use of numerical optimization techniques. Since the explicit calculation of the Hessian matrix can potentially be a costly procedure, we consider an optimization framework based on the Limited memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) optimizer [104, 21], which exploit the Broyden-Fletcher-Goldfarb-Shanno (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 LBFGS-based methods can be particularly appreciated to solve problems in which the Hessian matrices are large, unstructured, dense, or even unavailable.

Limited memory BFGS optimization

In the following, we provide some background knowledge on the LBFGS method, used in our fitting method to minimize (60). 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 LBFGS method is a quasi-Newton algorithm for solving large, i. e. $n \gg 50$ [21], non-linear optimization problems, namely

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}), \tag{77}$$

where $\boldsymbol{x}$ is the set of unknown variables. The LBFGS 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 LBFGS method is identical to the BFGS method. For $k > \nu$, H_k is obtained by applying m 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

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

and thereafter,

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

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

$$\begin{aligned}
\boldsymbol{d}_k &= -H_k \boldsymbol{g}_k, \\
\boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \gamma_k \boldsymbol{d}_k,
\end{aligned}$$

where γ_k is a scalar variable controlling the step-size, which satisfies the Wolfe conditions [165, 166], i. e.

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

with $0 < c_1 < c_2 < 1$. Finally, the LBFGS 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 27. *The LBFGS algorithm uses a sequence of m gradient vectors to approximate the inverse of the Hessian matrix. Thus, a large value of m 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 $\nu = 20$ [123, 170].*

Adaptive spline fitting with LBFGS

We employ the LBFGS algorithm to address the problem of spline surface fitting. In particular, given a point cloud $\mathcal{P}$ as in (1), 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 spline space $V = \operatorname{span}\{\beta_0, \dots, \beta_n\}$ of dimension $n+1$, our goal consists in finding a spline model $\boldsymbol{s} \in V$, which satisfied (60). 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 HLBFGS optimizer until convergence.

Note that there are no requirements concerning the initial guesses in steps 1 and 2. For instance, any method discussed in Chapter 3 can be suitably applied to compute the initial parameterization, as any fitting method presented in Chapter 2 or in the first part of the present Chapter to compute the initial spline configuration. Nevertheless, both the chosen parameterization and spline geometry need to be *good enough* to guarantee fast convergence to a more accurate solution.

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

$$\boldsymbol{c} \in \mathbb{R}^{(n+1)\times N} \quad \text{and} \quad \boldsymbol{u} \in \mathbb{R}^{m\times D},$$

respectively. Moreover, we consider the collocation matrix $B = B(\mathcal{U})$ as in (26), and we obtain the matrix form of the objective function (60), i. e.

$$f(\boldsymbol{c}, \boldsymbol{u}) = \|B\boldsymbol{c} - \boldsymbol{p}\|_F^2 + \lambda J(\boldsymbol{c}) = \operatorname{trace}\left\{ [B\boldsymbol{c} - \boldsymbol{p}] [B\boldsymbol{c} - \boldsymbol{p}]^\top \right\} + \lambda J(\boldsymbol{c}), \tag{78}$$

where $\|\cdot\|_F$ represents the Frobenius norm, $\boldsymbol{p} \in \mathbb{R}^{m\times N}$ is a matrix containing the data points $\mathcal{P}$, and $J(\boldsymbol{c})$ represents the contribution of the smoothing term. In order to exploit the LBFGS optimizer, we need to provide the gradient of the objective function (78), 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,

$$\nabla f(\boldsymbol{c}, \boldsymbol{u}) = \left(\frac{\partial f}{\partial c_0^{(1)}}, \dots, \frac{\partial f}{\partial c_n^{(1)}}, \dots, \frac{\partial f}{\partial c_0^{(N)}}, \dots, \frac{\partial f}{\partial c_n^{(N)}}, \frac{\partial f}{\partial u_1^{(1)}}, \dots, \frac{\partial f}{\partial u_m^{(1)}}, \dots, \frac{\partial f}{\partial u_1^{(D)}}, \dots, \frac{\partial f}{\partial u_m^{(D)}} \right), \tag{79}$$

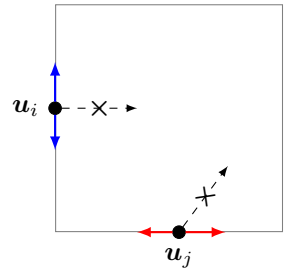


Figura 72. Two parametric sites $\boldsymbol{u}_i$ and $\boldsymbol{u}_j$ respectively constrained to the west and the south boundary edge of the rectangular domain Ω. The only possible directions are highlighted by blue arrows for $\boldsymbol{u}_i$ and red arrows for $\boldsymbol{u}_j$. In addition, not acceptable directions are represented by dashed arrows, marked by a cross symbol.

hence, by letting G as in (65),

$$\begin{aligned} \nabla_{\boldsymbol{c}} f(\boldsymbol{c}, \boldsymbol{u}) &= \left(B^\top B + \lambda G\right) \boldsymbol{c} - B^\top \boldsymbol{p} \\ \nabla_{u^{(k)}} f(\boldsymbol{c}, \boldsymbol{u}) &= \frac{1}{2} \nabla_{u^{(k)}} \operatorname{trace}\left((B\boldsymbol{c} - \boldsymbol{p}) (B\boldsymbol{c} - \boldsymbol{p})^\top \right) \\ &= \operatorname{diag}\left((B\boldsymbol{c} - \boldsymbol{p}) (\partial_{u^{(k)}} B\boldsymbol{c})^\top \right), \quad k = 1, \dots, D. \end{aligned} \tag{80}$$

Similarly to Remark 22, we are again dealing with the disjoint union of interior and boundary points and, consequently, with interior and boundary parameters, i. e. $\mathcal{U} = \mathcal{U}_I \dot{\cup} \mathcal{U}_B$. In particular, in this case, we constrained 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, hence also from (79), the parametric directions $u_j^{(k)}$, for $j = n+1, \dots, m$ and some $k = 1, \dots, D$, pointing towards the interior of Ω. Figure 72 illustrates the boundary constraints in the case of $D = 2$ and $\Omega = [0, 1]^2$, with 4 boundary edges. More precisely, the parametric site $\boldsymbol{u}_i = (u_i^{(1)}, u_i^{(2)})$ belongs to the west edge of the domain; hence, the only component of $\boldsymbol{u}_i$ that plays a role in the optimization problem (60) is $u_i^{(2)}$, whereas $u_i^{(1)}$ is kept fixed to 0. Similarly, $\boldsymbol{u}_j$ belongs to the south edge of the domain; hence, we optimize $u_j^{(1)}$ while we set $u_j^{(2)} = 0$.

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. For completeness, we recall that a B-spline curve fitting method based on LBFGS has been developed in [170].

In this Section, 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 (19), belonging to a hierarchical spline space V, at the beginning of each adaptive routine (4), 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 LBFGS optimizer defines the new control points $\boldsymbol{c}_j \in \mathbb{R}^N$, for $j \in \mathcal{A}_{\boldsymbol{k}}^{\ell}$ and $\ell = 0, \dots, L-1$, 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. To summarize, the optimal geometry $\boldsymbol{s}$ is evaluated on the new parametric sites $\mathcal{U}$ and the point-wise error estimator is computed, namely

$$\|\boldsymbol{s}(\boldsymbol{u}_i) - \boldsymbol{p}_i\|_2 , \text{ for each } i = 1, \dots, m.$$

We identify the cells of the current hierarchical level ℓ, which contain the parameters $\boldsymbol{u}_i$ so that $\|\boldsymbol{s}(\boldsymbol{u}_i) - \boldsymbol{p}_i\|_2 > \epsilon$, for a certain input error threshold $\epsilon > 0$, and mark them for refinement, together with two surrounding rings of cells in the hierarchical mesh. Finally, the marked cells are dyadically split in order to refine the hierarchical space.

Remark 28. *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 HLBFGS optimizer on the enlarged hierarchical spline space, we need to project the solution from the previous iteration to the new spline space.*

Finally, we suggest as automatic approach to generating the initial tensor-product B-spline fitting geometry to solve the (weighted) LS problem for a given input parameterization, see Section 1. The joint optimization routine is summarized in Algorithm 8.

Numerical results

In this Section, we present a comparison between the adaptive fitting schemes with moving parameterization presented in this Chapter, i. e. the adaptive A-PDM, A-HDM methods of Section 2, and the adaptive J-PDM of Section 3. In Section 10 we approximate a ship hull point cloud of 3025 gridded data, for which we investigate different fine-tuning strategies for adaptive J-PDM. In Section 10, we approximate a real-world data set of 9000 points representing a Nefertiti bust. From both examples, the J-PDM method results in the best approximation accuracy from a qualitative and quantitative perspective, being on the other hand the most computationally expensive.

The algorithms related to the adaptive fitting schemes with THB-splines have been implemented in `C++` with the open-source G+Smo library [78, 112]. The developed code for the proposed algorithms has been integrated and will be available in the next releases. As concerns the implementation of the LBFGS optimization method, we exploit the HLBFGS `C++` library [107], which provides the LBFGS method, the preconditioned LBFGS method [77], and the preconditioned conjugate gradient method [134, 133].

Algorithm 8: Adaptive joint optimization fitting scheme.

Input: Point cloud and parameters $\{\boldsymbol{u}_i, \boldsymbol{p}_i\}_{i=1}^m$, an initial tensor-product B-spline space V^0, the error tolerance $\epsilon > 0$, a maximum number of hierarchical levels L.

Compute the initial tensor-product LS B-spline approximation $\boldsymbol{s} \in V^0$ and the point-wise errors $e_i = \|\boldsymbol{s}(\boldsymbol{u}_i) - \boldsymbol{p}_i\|_2$, for each $i = 1, \dots, m$ and set loop $= 0$

while $\max_i e_i > \epsilon$ *and* loop $<$ L **do**

  Solve the non linear least squares problem

$$\boldsymbol{s} = \underset{\substack{\boldsymbol{c}_j,\ j \in \mathcal{A}^\ell, \ell = 0, \dots, \text{loop} \\ \boldsymbol{u}_i \in \Omega, i = 1, \dots, m}}{\arg\min} \frac{1}{2} \sum_{i=1}^{m} \|\boldsymbol{s}(\boldsymbol{u}_i) - \boldsymbol{p}_i\|_2^2 + \lambda J(\boldsymbol{c}).$$

  Compute the errors $e_i = \|\boldsymbol{s}(\boldsymbol{u}_i) - \boldsymbol{p}_i\|_2$, for $i = 1, \dots, m$.

  Mark the domain elements where $e_i > \epsilon$.

  Refine the marked cells and two surrounding rings of cells.

  Update the hierarchical mesh $\mathcal{M}$ and hierarchical space V.

  Project the solution $\boldsymbol{s}$ onto the refine space V.

  Set loop $=$ loop $+ 1$.

end

Output: $\boldsymbol{s} \in V$, the THB-spline approximant.

Comparison on the ship hull point cloud

In this example, we revisit the adaptive approximation analyzed in Section 12. We start with the same initial configuration, i. e. we approximate a ship hull point cloud consisting of 3025 gridded data points and initial CEN parameterization, starting with a tensor-product B-spline of bi-degree $\boldsymbol{d} = (2, 2)$ on a 5×2 mesh. Differently from Section 12, we lower the refinement threshold to $\epsilon = 1\text{e} - 5$ to induce more refinement levels.

As concerns the adaptive A-PDM we perform 10 steps of PC within each adaptive loop. As concerns adaptive J-PDM, we exploit two configurations for its hyper-parameters, namely the tolerance on the norm of the gradient ν and the maximum number of iterations $K_{\max}$.

(a) In this first configuration we set $\eta = 1\text{e} - 6$ and $K_{\max} = 500$.

(b) In this second configuration, we set $\eta_0 = 1\text{e} - 5$ and $K^0_{\max} = 2^\ell \cdot 200$ and adaptively change them with the number of hierarchical level, i. e.

$$\eta^\ell = \frac{\eta_0}{2^\ell}, \text{ and } K^\ell_{\max} = 2^\ell \cdot K^0_{\max}, \tag{81}$$

for $\ell = 0, \dots, L - 1$.

The configuration in (b) arises from the idea that spline space V poses a constraint to the solution of (60). In particular, if the approximation space

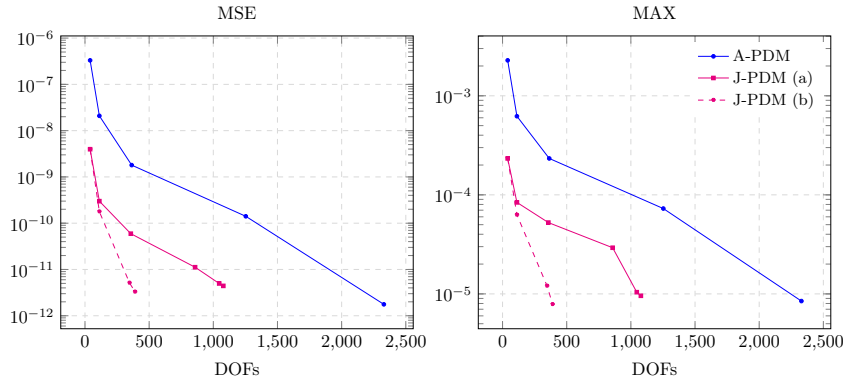


Figura 73. Adaptive spline approximation for the ship hull point cloud consisting of 3025 gridded data points and initial CEN parameterization in Section 10. Comparison of MSE (left) and MAX (right) between adaptive A-PDM (blue) and the J-PDM approach with constant hyper parameters (solid magenta) and adaptive hyper parameters (dashed magenta).

V is too coarse, it is not reasonable to spend computational resources, by setting ν too low or $K_{\max}$ too high, to find an optimal solution in V, which will not satisfy the refinement threshold ϵ and will be anyway recomputed after refinement. On the other hand, when the space V enlarges, then it is more plausible that the solution satisfying the input tolerance ϵ belongs to it; hence, it is valuable to search for an optimum within this space. Higher accuracy is achieved by lowering the threshold ν; consequently, we also increase the LBFGS maximum number of iterations $K_{\max}$ in order not to stop the optimization routine too early. Note that a too low choice for η^0 and $K^0_{\max}$ might result in a geometry with too poor quality at the end of the first loop of the adaptive scheme, which would also affect the refinement and the whole adaptive approximation results, similarly to the analysis in Section 12.

The results are shown in Figure 73 in terms of MSE (left) and MAX (right) for the adaptive A-PDM (solid blue line), J-PDM for the configuration (a) (solid magenta line), and J-PDM for the configuration (b) (dashed magenta line). We can observe that both J-PDM configurations show considerably better approximation accuracy with respect to A-PDM in terms of MSE and MAX errors. More specifically, we can note that the convergence rate of the adaptive J-PDM with the hyper-parameter configuration in (b) overcomes all the other schemes, satisfying the input tolerance ϵ with less than 400 DOFs, whereas in all the other configurations more than 1000 DOFs are required, up to 2329 DOFs for adaptive A-PDM.

The reconstructed geometries are shown in Figure 74, from left to right for A-PDM, J-PDM (a), and J-PDM (b), together with their DOFs, MSE and MAX values.

A-PDM
DOFs = 2329
MSE = 1.76e − 12
MAX = 8.48e − 6

J-PDM (a)
DOFs = 1078
MSE = 4.21e − 12
MAX = 9.56e − 6

J-PDM (b)
DOFs = 634
MSE = 3.29e − 12
MAX = 7.84e − 6

Figura 74. Adaptive spline approximation for the ship hull point cloud consisting of 3025 gridded data points and initial CEN parameterization in Section 10. From left to right: final THB-spline model with adaptive A-PDM, J-PDM (a) and J-PDM (b). The DOFs, MSE and MAX values are also reported.

Comparison on the Nefertiti real-wrold point cloud

In this last example, we compare the adaptive fitting methods with moving parameterization on a real-world dataset, corresponding to 9000 scattered points representing a Nefertiti bust already considered in Section 12, and we compute the initial parameterization with the BIDGCN method, presented in Chapter 3. We start the adaptive loop from a polynomial space of bi-degree $\boldsymbol{d} = (2, 2)$ and set the refinement threshold to $\epsilon = 2.5\text{e}-3$. The input point cloud is characterized by high-curvature regions; hence, the point projection involved in the adaptive alternating methods is very challenging. Consequently, performing too many PC steps affects the quality of the final reconstructed geometry. For this reason, we perform only 1 PC step within each adaptive loop of the alternating methods. By the analysis conducted in Section 1 and Section 12, adaptive A-HDM method with curvature-/error-based blending weights does not show a stable behaviour after only 1 step of PC, hence proceeding with the refinement ruins the accuracy of the final reconstructed hierarchical model. For this reason, we consider A-HDM with constant blending weights, with $\gamma = 1/3$, chosen after an appropriate fine-tuning. As concerns adaptive J-PDM, we set $\eta = 1\text{e} - 5$ and $K_{\max} = 100$ and we keep them constant.

The results in terms of MSE and MAX are illustrated in Figure 75, whereas Figure 76 shows the qualitative results. More precisely, from left to right, the moved parameters, the hierarchical mesh, the hierarchical reconstructed geometry, and the scaled point-wise error distributions for the three considered fitting schemes, adaptive A-PDM (top), A-HDM (middle), and J-PDM (bottom).

We can notice that A-PDM and A-HDM show very similar approximation behaviour in terms of MSE. This can be motivated by the choice of the initial parameterization: BIDGCN computes a parameterization that minimizes the PDM error term, see Section 3, and the optimality is maintained also for the adaptive A-PDM scheme, characterized by the minimization

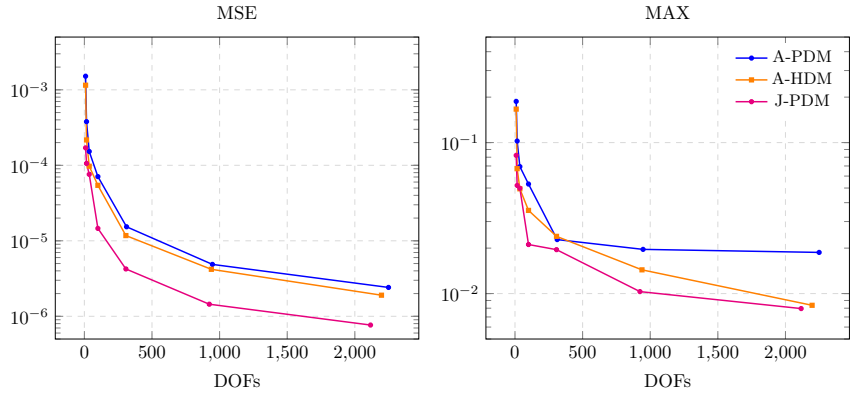


Figura 75. Adaptive spline approximation for the Nefertiti point cloud consisting of 9000 scattered data points with initial BIDGCN parameterization. Comparison of MSE (left) and MAX (right) between adaptive A-PDM (blue), A-HDM (orange) and J-PDM (magenta).

of the same error term. On the other hand, the MAX error stagnates at $\sim 2.00\text{e}-2$ for adaptive A-PDM, whereas it keeps decreasing for adaptive A-HDM until $8.35\text{e}-3$, achieved with 2249 and 2197 DOFs, respectively. Finally, adaptive J-PDM exhibits a better approximation capacity both for MSE and MAX if compared with the adaptive alternating methods. As concerns the MAX, adaptive A-HDM and J-PDM converge to very similar values. More precisely, J-PDM registers a MAX error of $7.96\text{e}-3$, obtained with 2116 DOFs. However, the quality of the reconstructed geometry is higher for J-PDM, as can be observed in Figure 76, especially in the area around the eyes.

4.4. Industrial applications with adaptive A-PDM and A-QI

In this Section, we illustrate the performance of the proposed fitting methods within an industrial environment. More precisely, we consider the reverse engineering problem of reconstructing highly accurate CAD models of aircraft engine components. This reverse engineering process consists of three *major* steps, i. e.

1. data acquisition;
2. point processing;
3. surface fitting;
4. geometric model post-processing.

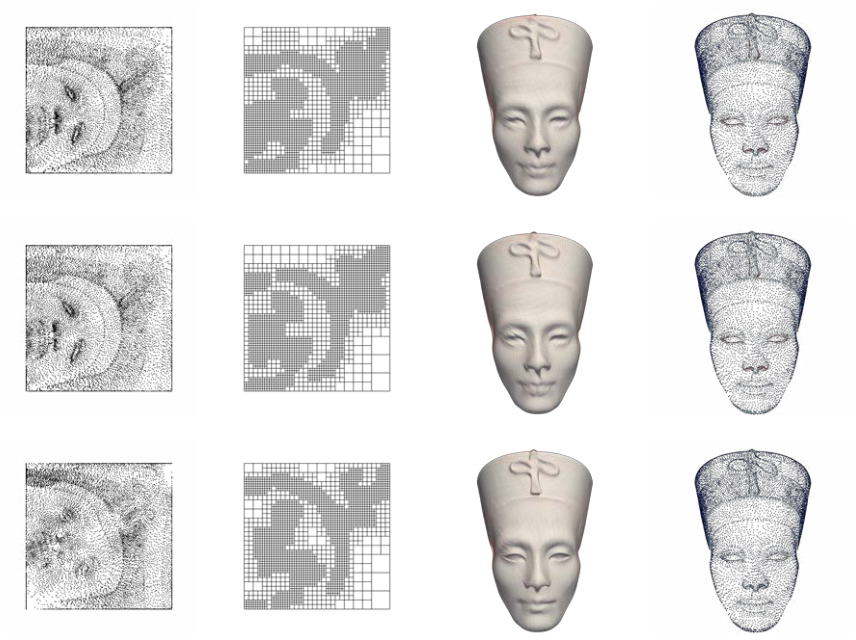


Figura 76. Adaptive spline approximation for the Nefertiti point cloud consisting of 9000 scattered data points with initial BIDGCN parameterization. Comparison of adaptive A-PDM (top line), A-HDM (central line), and J-PDM (bottom line). From left to right: the final (moved) parameters, the final hierarchical mesh, the final approximating THB-spline geometry ad the scaled point wise error.

More specifically, step 1 consists of the collection of the data from a physical part through scanning, e. g., white light scanners, laser scanners, charge-coupled device (CCD) sensors, or coordinate-measuring machines (CMM). Step 2 concerns more *specific* procedures, such as the construction of a suitable triangulation, trimming, feature extractions, and data segmentation, among others. In particular, these routines are needed to recover intrinsic features of the geometric part, e. g., holes, slots, or curves. In step 3, dynamic fitting algorithms are used for surface generation. That is the case of our examples, where the considered data, due to the acquisition process, are scattered and affected by noise, yet the reconstructed geometric models are required to be compact and smooth while simultaneously capturing key geometric features of the engine parts. Finally, step 4 consists of the post-processing of the reconstructed surface models to shape and assemble the final CAD model. For instance, trimming routines can be performed to introduce holes, see e. g., [116], whereas multi-patch representations can be exploited to merge several components of the CAD model, see e. g., [159]. Subsequently, the virtual models obtained are used for redesign operations, further manufacturing, or numerical simulations.

Throughout this section, we use two fitting methods, namely the global adaptive PDM approximation with THB-splines and the QI scheme using

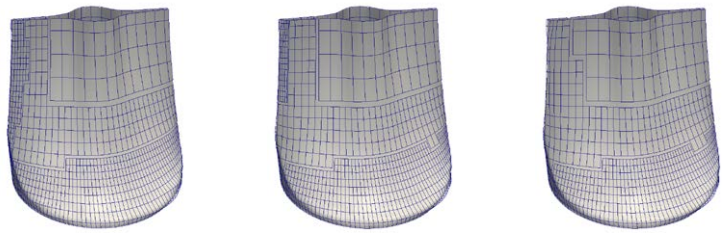


Figura 77. Adaptive THB-spline QI with moving parameters (adaptive A-QI) in Section 4. From left to right: models computed with 0, 1, and 2 PC steps.

local spline approximation, both described in Chapter 2, to assess the benefits of the moving parameterization approach. Hence, we compare adaptive PDM and QI with adaptive A-PDM and A-QI on scanned data provided by MTU Aero Engines.

The algorithms related to the adaptive fitting schemes with THB-splines have been implemented in `C++` with the open-source G+Smo library [78, 112]. The developed code for the proposed algorithms has been integrated and will be available in the next releases.

Tensile with moving parameterization

In this example, we revisit the second configuration of Section 3, where the point cloud of a tensile part is initially approximated by a bi-quadratic spline on a tensor-product basis defined on a 4×16 mesh, with tolerance $\epsilon = 5\text{e} - 5\text{m}$. We compare the performance of the QI scheme without PC and with one or more rounds of PC after the adaptive refinement step (adaptive A-QI). The LBFGS optimizer with minimum step-size $s = 1\text{e} - 9$ is used for the foot-point projection. In particular, the resulting geometries when applying 0, 1 or 2 rounds of PC after each refinement step are shown in Figure 77. We can observe that all the fitting surfaces are of good quality. Nevertheless, in this case suitably embedding the parameter correction routine in the adaptive loop led to gaining the same precision in terms of MAX and MSE errors, while using fewer degrees of freedom to reconstruct the final geometric model, as summarized in the first three lines of Table 2. In particular, applying 3 steps of PC after each hierarchical mesh refinement produces an approximation with 15.20% fewer DOFs with respect to the one achieved with the standard adaptive QI scheme, while simultaneously reducing the MAX error by 9.89% and the MSE by 17.20%. Note that additional rounds (from 4 to 6 in Table 2) of parameter correction, yield diminishing improvements, since the DOFs are still reduced, but there is no gain in terms of MAX and MSE. This suggests that from 1 to 3 PC steps are in general a proper choice when including the parameter correction within A-QI.

Tabella 2. Analysis in terms of points within the prescribed tolerance (%pts $< \epsilon$), MAX or MSE, and final number of DOFs for the adaptive THB-spline QI scheme in Section 4. The percentages of the DOFs reduction with respect to QI without PC are also reported.

method	%pts $< \epsilon$	MAX (m)	MSE (m^2)	DOFs	
QI	99.36	$8.11e-5$	$1.42e-10$	1960	
A-QI, 1 PC	99.33	$8.17e-5$	$1.41e-10$	1834	6.43%
A-QI, 2 PC	99.40	$7.97e-5$	$1.25e-10$	1718	12.35%
A-QI, 3 PC	99.46	$7.31e-5$	$1.17e-10$	1662	15.20%
A-QI, 4 PC	99.15	$8.66e-5$	$1.22e-10$	1659	15.36%
A-QI, 5 PC	99.02	$8.52e-5$	$1.23e-10$	1621	17.30%
A-QI, 6 PC	99.11	$9.19e-5$	$1.15e-10$	1621	17.30%

Blade with moving parameterization

In this example, we revisit Section 3. However, in view of the considerations of Section 12, the initial tensor-product basis of bi-degree $\boldsymbol{d} = (3, 3)$ is now more refined, i. e. we start with a 8×8 tensor product mesh. The results are still comparable, as the lowest level basis is entirely refined in the configuration considered in Section 3. We then fit the set of 27191 measured data points from a blade geometry that has a length of about $5e-2m$. We compare adaptive A-PDM and A-QI, by always considering as smoothing coefficient $\lambda = 1e-8$. In both cases, we stop when at least 95% of data points are within the tolerance $\epsilon = 2e-5m$.

Figure 78 shows the final THB-spline models obtained with adaptive A-PDM (top) and A-QI (bottom) when 0 (PDM, QI), 1, or 2 PC steps are considered in the adaptive THB-spline fitting schemes. In particular, we can observe that the A-QI geometries are characterized by a smaller amount of oscillations along the sharp features. To have more insights into this phenomenon, we present the reflection lines on the different THB-spline models in Figure 79. By analyzing the top and bottom row of this figure from left to right, we see that the PC improves the surface quality by reducing the oscillations near the top left corner. Moreover, by comparing the A-PDM (top) and A-QI (bottom) results in each column, we can see that the oscillations just under the feature in the top right area of the blade are reduced in the THB-spline models obtained with A-QI scheme.

The quantitative analysis for this example is detailed in Table 3. In particular, both for A-PDM and A-QI schemes, we report the final percentage of points within the tolerance, the MAX and MSE errors, as well as the number of DOFs, when 0 (PDM, QI), 1, or 2 parameter corrections are considered. Note that the stopping criterion (95% pts $< 2e-5m$) is met in all cases, with a significant reduction ($\sim 70\%$) of DOFs both for A-PDM and A-QI schemes when 1 or 2 PC steps are applied. This is due to the fact that the prescribed precision is achieved two iterations earlier, preventing the introduction of two additional hierarchical levels. This can be explicitly seen in Figure 80 for A-PDM (left) and A-QI (right), where the number of points below the prescribed tolerance versus the amount of DOFs at each adaptive iteration is reported. In this example, either one or two PC steps,

Tabella 3. Analysis in terms of points within the tolerance (%pts $< \epsilon$), MAX or MSE, and number of DOFs for the adaptive THB-spline A-PDM and A-QI schemes in Example 4. The percentages of the DOFs reduction with respect to A-PDM and A-QI without PC are also reported.

method	%pts $< \epsilon$	MAX (m)	MSE (m^2)	DOFs	
PDM	99.99	$2.12e-5$	$6.32e-12$	8164	
A-PDM, 1 PC	99.19	$4.04e-5$	$2.48e-11$	2314	71.66%
A-PDM, 2 PC	99.65	$2.90e-5$	$2.37e-11$	2212	72.91%
QI	99.98	$2.74e-5$	$1.00e-11$	8278	
A-QI, 1 PC	97.42	$8.86e-5$	$4.71e-11$	2554	69.15%
A-QI, 2 PC	95.97	$5.71e-5$	$5.93e-11$	2539	69.33%

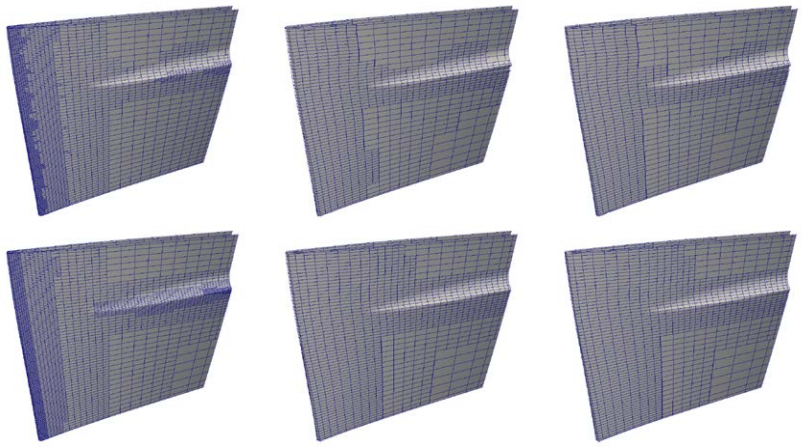


Figura 78. THB-spline models with control nets in Example 4. Top row, from left to right: PDM, A-PDM with 1, and 2 PC steps; bottom row: from left to right: QI, A-QI with 1, and 2 PC steps.

after each refinement procedure, are a good choice, since the results are very similar, iteration by iteration, both for A-PDM and A-QI.

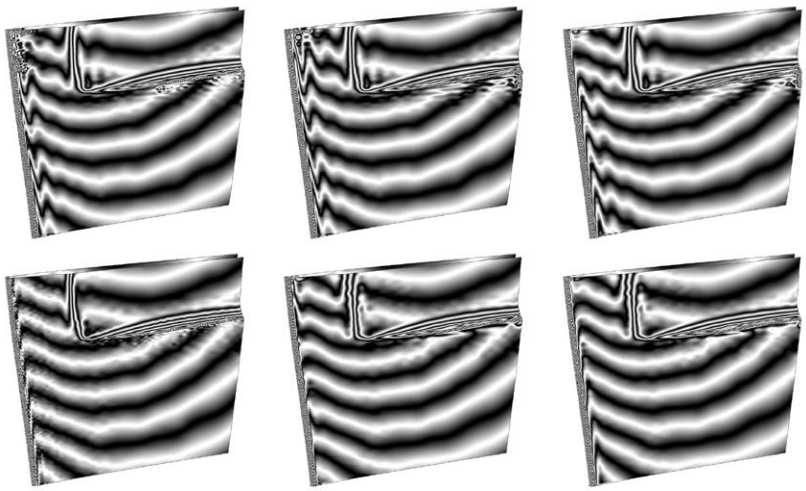


Figura 79. THB-spline models with reflection lines in Example 4. Top row, from left to right: PDM, A-PDM with 1, and 2 PC steps; bottom row: from left to right: QI, A-QI with 1, and 2 PC steps.

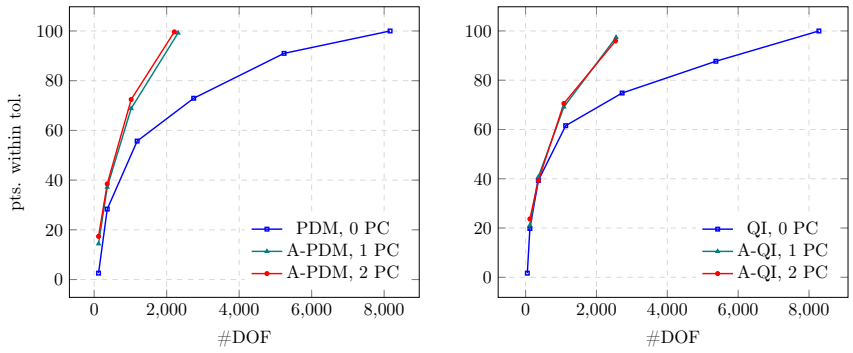


Figura 80. Percentage of points within the prescribed tolerance during the THB-spline PDM and A-PDM schemes in Example 4 (left) and during the THB-spline QI and A-QI schemes in Example 4 (right).