Mathematics and Computers in Simulation 228 (2025) 103–123

Contents lists available at ScienceDirect

Mathematics and Computers in Simulation

journal homepage: www.elsevier.com/locate/matcom

Original articles

Local spline refinement driven by fault jump estimates for scattered data approximation

Cesare Bracco, Carlotta Giannelli, Francesco Patrizi *, Alessandra Sestini

Department of Mathematics and Computer Science "Ulisse Dini", University of Florence, Viale Giovanni Battista Morgagni 67/a, 50134 Florence, Italy

ARTICLE INFO

Keywords:
Fault detection
Gradient fault detection
Fault jump estimate
Scattered data
Adaptive surface reconstruction
LR B-splines
Quasi-interpolation

ABSTRACT

We present new fault jump estimates to guide local refinement in surface approximation schemes with adaptive spline constructions. The proposed approach is based on the idea that, since discontinuities in the data should naturally correspond to sharp variations in the reconstructed surface, the location and size of jumps detected in the input point cloud should drive the mesh refinement algorithm. To exploit the possibility of inserting local meshlines in one or the other coordinate direction, as suggested by the jump estimates, we propose a quasi-interpolation (QI) scheme based on locally refined B-splines (LR B-splines). Particular attention is devoted to the construction of the local operator of the LR B-spline QI scheme, which properly adapts the spline approximation according to the nature and density of the scattered data configuration. A selection of numerical examples outlines the performance of the method on synthetic and real datasets characterized by different geographical features.

1. Introduction

Computer aided design methods offer accurate and flexible shape representations using trimmed tensor product B-spline surfaces. However, due to the inherent rigidity of tensor product B-spline constructions, achieving localized refinement requires the identification of a suitable non-standard B-spline-like basis across various unstructured mesh configurations. Several constructions are available in the literature, including T-splines [1], (truncated) hierarchical B-splines (THB-splines) [2], and locally refined B-splines (LR B-spline) [3]. In practice, starting with an initial tensor product grid, adaptive spline refinement is usually performed by simply considering a dyadic splitting of the selected mesh elements. Few attempts were proposed so far to construct locally refinable extensions of tensor product constructions which enable anisotropic local knot insertion in one or the other direction. In principle, both T-splines and LR-splines allow the insertion of local knot segments in arbitrary regions and directions of the parametric domain. However, since these adaptive spline models do not necessarily guarantee the linear independence of the considered blending functions [1,4,5], the focus of the research communities so far was mainly devoted to the identification of admissible mesh configurations for a proper basis construction. In particular, different refinement algorithms that generate LR meshes with the needed properties for LR B-spline spaces were recently proposed [6–8]. In the hierarchical spline framework instead, the linear independence of the basis construction is straightforward [9] but the required nestedness of the tensor product spline spaces, considered at different refinement levels, precludes anisotropic refinement schemes along arbitrary directions. In order to overcome these limitations, partially nested hierarchical B-spline refinement was proposed as Patchwork B-splines (PB-splines) [10,11] by necessarily penalizing the inherent simplicity of hierarchical spline constructions.

* Corresponding author.
E-mail addresses: cesare.bracco@unifi.it (C. Bracco), carlotta.giannelli@unifi.it (C. Giannelli), francesco.patrizi@unifi.it (F. Patrizi), alessandra.sestini@unifi.it (A. Sestini).

https://doi.org/10.1016/j.matcom.2024.08.031
Received 18 November 2023; Received in revised form 21 June 2024; Accepted 25 August 2024
Available online 30 August 2024

Discontinuity detection algorithms have been widely considered in different application settings to properly identify salient features in the input data and drive surface reconstruction processes. For example, fault indicators based on radial basis function interpolation were introduced in [12,13]. Detection and recovery of discontinuity curves from scattered data were also previously addressed in [14,15] relying on an intermediate gridded data approximation. Ordinary and gradient fault indicators based on numerical differentiation formulas were proposed in [16,17] to work directly on unstructured input data without the need of additional data processing. More recently, the use of the so-called null rules was proposed for the design of fault detection algorithms [18], by also considering its application to (adaptive) surface reconstruction with THB-splines. Such reconstruction employs a Quasi-Interpolation (QI) methodology, for a comprehensive overview, refer to the recent book [19]. A QI is composed of two ingredients essentially: a basis for a finite dimensional functional space such as, e.g., B-splines or radial functions, and linear functionals, of low computational costs, providing coefficients for the basis elements. A popular choice for the latter are the discrete Lagrange type functionals, which are based on a finite number of function evaluations. Discrete QIs have also been extensively used, for instance, in formulating numerical quadrature rules, see for example [20–22]. In this work as well we consider discrete Lagrange type quasi-interpolation for developing an adaptive scattered data approximation method based on Locally Refined (LR) B-splines. LR spaces have been already exploited for similar purposes in [23–27], in particular for surface reconstruction from terrain and bathymetry datasets. However, in these references, the approximation is carried out through the so-called multilevel B-spline methods rather than quasi-interpolation. Furthermore, the adaptivity of their schemes rely on error evaluation, which requires the computation of the approximate surface at each refinement iteration. The repeated assembling and evaluation of such intermediate approximating surfaces constitute the most expensive steps and may result highly time-consuming for large point clouds. Conversely, and similar to the approach in [18] for THB splines, our method separates adaptive mesh generation from reconstruction, eliminating the need for intermediate constructions and evaluations during the refining process. Additionally, thanks to the new jump estimates, we achieve a more comprehensive understanding of both ordinary and gradient faults. This additional information is effectively employed to enhance the compression capability of the scheme without compromising the fidelity of the produced reconstruction. Moreover, LR B-splines allow for anisotropic refinements, as local meshlines can be inserted in only one coordinate direction as opposed to THB splines. This property is exploited to further reduce degrees of freedom while sustaining the same accuracy at straight fault portions. Particular attention is also devoted to the construction of the local operator of the LR B-spline QI scheme in such a way that it properly adapts to the nature and density of the scattered data configuration. A selection of numerical examples outlines the performance of the method on synthetic and real datasets characterized by different geographical features.

The structure of the paper is as follows. Section 2 briefly recalls the definition and key properties of LR B-splines in the bivariate setting and the fault detection approach for scattered data introduced in [16,17] and here adopted. The new jump estimates are presented in Section 3, while Section 4 introduces the algorithm here developed to produce the final adaptive LR quasi-interpolating spline for scattered data approximation. The numerical examples are then presented in Section 5. Finally, Section 6 concludes the paper.

2. Preliminaries

Let $f : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$ be a function on a closed finite domain Ω, and $X \subset \Omega$ be a set of scattered data-sites, with associated function values $f(X) := \{f(\boldsymbol{x}) : \boldsymbol{x} \in X\}$. The goal of the paper is to define an adaptive spline quasi-interpolant which approximates the function f by using its values at the points in X. The main idea is that, since discontinuities of the function or of its gradient correspond to the main sharp variations of the function, the refinement of the spline space used in the approximation can be driven by the location and size of such jumps. As spline space, we choose the one spanned by LR B-splines, which, allowing local anisotropic refinement, provides the flexibility we need. In this section we recall the basics of LR B-splines and the technique we employ to find the location of the discontinuities.

2.1. Locally refined B-splines

Let $\mathcal{M}$ be an open tensor mesh on Ω with respect to a bidegree $\mathbf{p} = (p_1, p_2)$, that is, with boundary meshlines of full multiplicities, that is, with corresponding knot value repeated $p_k + 1$ times with $k = 1$ for vertical meshlines and $k = 2$ for horizontal meshlines, respectively, and with simple internal meshlines, i.e., of multiplicity one (with no repetitions of their knot values). Consider a sub-mesh in $\mathcal{M}$, denoted by $\mathcal{M}_B$, composed of $p_1 + 2$ vertical meshlines and $p_2 + 2$ horizontal meshlines, respectively, counting the meshline multiplicities. Such meshlines can be parametrized as $\{x_i\} \times [y_1, y_{p_2+2}]$ and $[x_1, x_{p_1+2}] \times \{y_j\}$ with $\mathbf{x} := (x_i)_{i=1}^{p_1+2}$ and $\mathbf{y} := (y_j)_{j=1}^{p_2+2}$ such that $x_i \le x_{i+1}$ and $y_j \le y_{j+1}$ for all $i = 1, \dots, p_1 + 1$ and $j = 1, \dots, p_2 + 1$. $\mathbf{x}$ and $\mathbf{y}$ are local knot vectors in the two directions from which we can defined a tensor product B-spline $B := B[\mathbf{x}, \mathbf{y}]$. Thus, there is a one-to-one correspondence between the sub-meshes, such as $\mathcal{M}_B$, that one can identify on $\mathcal{M}$ and the tensor product B-splines that can possibly be defined on $\mathcal{M}$. Among all of them, there is the special class of minimal support B-splines.

Definition 2.1. We say that B has minimal support on $\mathcal{M}$ if no line in $\mathcal{M} \setminus \mathcal{M}_B$ traverses the interior of the support of B, $\mathrm{int}(\mathrm{supp}\, B)$, entirely.

Therefore, what is commonly called (tensor product) B-spline set on a tensor mesh $\mathcal{M}$ is the class of minimal support B-splines on $\mathcal{M}$. Given now an open tensor mesh $\mathcal{M}$ with associated B-spline set $\mathcal{B}$, assume to insert a new meshline γ in $\mathcal{M}$, with endpoints on $\mathcal{M}$ and long enough to traverse entirely the support of one B-spline $B \in \mathcal{B}$ but not necessarily traversing the entire domain Ω (in

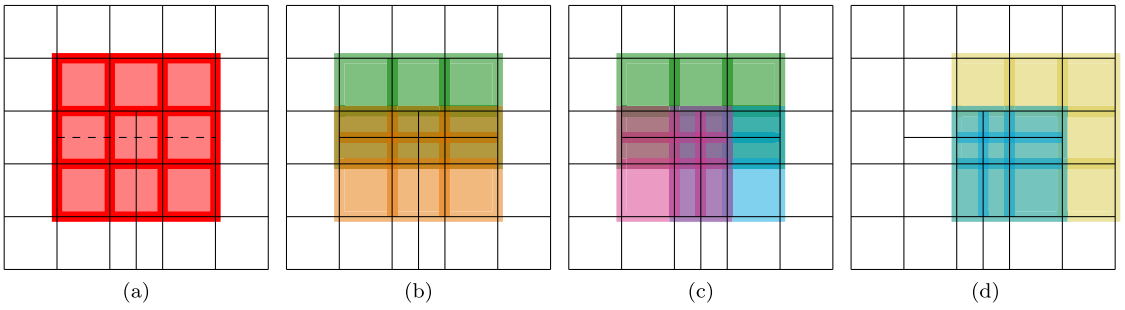


Fig. 1. Replacement of an LR B-spline due to the refinement of the LR mesh. Consider the LR mesh $\mathcal{M}$ reported in figure (a) and the bidegree $\mathbf{p} = (2, 2)$. Let $B[\mathbf{x}, \mathbf{y}]$ be the LR B-spline on $\mathcal{M}$ whose support and tensor mesh $\mathcal{M}_B$ are highlighted in figure (a). Let us insert a horizontal meshline γ (dashed in figure (a)). Such γ is traversing $\operatorname{supp} B$ and $B[\mathbf{x}, \mathbf{y}]$ is replaced by the B-splines $B[\mathbf{x}, \mathbf{y}_1]$ and $B[\mathbf{x}, \mathbf{y}_2]$ involved in the knot insertion. In figure (b) we see the supports and tensor meshes of the latter on the new LR mesh. In particular we see that $B[\mathbf{x}, \mathbf{y}_1]$ (the bottom B-spline in figure (b)) has not minimal support on the LR mesh as there is a vertical meshline traversing its support which is not in its associated tensor mesh. Thus $B[\mathbf{x}, \mathbf{y}_1]$ is replaced as well, via knot insertion, by two other B-splines $B[\mathbf{x}_1, \mathbf{y}_1], B[\mathbf{x}_2, \mathbf{y}_1]$. Therefore, in the end, we move from $B[\mathbf{x}, \mathbf{y}]$, which was defined on $\mathcal{M}$, to $B[\mathbf{x}_1, \mathbf{y}_1], B[\mathbf{x}_2, \mathbf{y}_1], B[\mathbf{x}, \mathbf{y}_2]$ on the new LR mesh $\mathcal{M}' := \mathcal{M} \cup \{\gamma\}$. The supports and tensor meshes of the latter are represented in figure (c). In figure (d) we see that $\mathcal{M}'$ has not the $\mathrm{N_2S}$ property as $B[\mathbf{x}_2, \mathbf{y}_1]$ is nested into another LR B-spline on the mesh and both are defined on simple knots, i.e., their associated tensor meshes have all distinct meshlines, of multiplicity one.

this case we would say that γ is a local meshline). Let $\mathcal{M}' := \mathcal{M} \cup \{\gamma\}$ be the new mesh. By construction, there is at least one B-spline $B = B[\mathbf{x}, \mathbf{y}] \in \mathcal{B}$ that has not minimal support on $\mathcal{M}'$. Suppose γ is a vertical meshline, $\gamma = \{\hat{x}\} \times [a, b]$ with $a \leq y_1$ and $b \geq y_{p_2+2}$. Then $\hat{x}$ is not in $\mathbf{x}$. By knot insertion of $\hat{x}$ in $\mathbf{x}$ we can express $B[\mathbf{x}, \mathbf{y}]$ in terms of two B-splines $B[\mathbf{x}_1, \mathbf{y}]$ and $B[\mathbf{x}_2, \mathbf{y}]$ with minimal support on $\mathcal{M}'$. This procedure is the key idea of the Locally Refined (LR) B-splines and meshes, which are defined as follows.

Definition 2.2. An LR mesh $\mathcal{M}'$ on Ω is either a tensor mesh or it is recursively obtained by inserting a meshline γ in a previous LR mesh $\mathcal{M}$, traversing the support of at least one of the LR B-splines on $\mathcal{M}$. The LR B-spline set on $\mathcal{M}'$ is either directly the B-spline set on $\mathcal{M}'$ if $\mathcal{M}'$ is a tensor mesh, or it is formed by replacing LR B-splines on $\mathcal{M}$ that do not have minimal support on $\mathcal{M}'$ with the minimal support B-splines on $\mathcal{M}'$ obtained from the knot insertion procedure.

It is important to underline that, despite a given LR mesh could be constructed with different meshline insertion orderings, the LR B-spline set is well-defined, that is, independent of which order is considered, as shown in [3, Theorem 3.4]. Figs. 1 (a)–(c) show how an LR B-spline is replaced when the underlying LR mesh is refined.

Furthermore, although being defined on locally refined meshes, the LR B-splines preserve most of the properties enjoyed by B-splines on tensor meshes, such as, non-negativity, locality of the supports, being piecewise polynomials and the so-called two-scale relations with only non-negative coefficients [28], that is, the possibility to be expressed by LR B-splines on finer LR meshes using non-negative coefficients (which are provided by the knot insertion procedure). Moreover, they form a partition of unity when suitable scaled with positive weights [3, Section 7].

However, while B-splines on tensor meshes are always locally linearly independent, the almost total absence of constraints in the refinement process of LR meshes may create linear dependence relations in the LR B-spline sets [5]. Nevertheless, LR B-splines can be as well locally linearly independent. This property has been indeed characterized as follows in [6, Theorem 4].

Theorem 2.3. *The following statements are equivalent:*

1. *The LR B-splines on the LR mesh $\mathcal{M}$ are locally linearly independent.*
2. *The LR B-splines on $\mathcal{M}$ form a partition of unity, without the use of scaling weights.*
3. *Each box β of $\mathcal{M}$ is contained in exactly $(p_1 + 1)(p_2 + 1)$ LR B-spline supports.*
4. *There exists no pair of LR B-splines B_1, B_2 on $\mathcal{M}$ defined on local knot vectors made of simple knots with $B_1 \neq B_2$ and $\operatorname{supp} B_1 \subseteq \operatorname{supp} B_2$.*

A box β of $\mathcal{M}$ verifying condition 3. is said non-overloaded and a mesh $\mathcal{M}$ for which condition 4. is satisfied is said to have the Non-Nested-Support ($\mathrm{N_2S}$) property. In Fig. 1(d) we see an example of LR mesh without the $\mathrm{N_2S}$ property.

Remark 2.4. The $\mathrm{N_2S}$ property has slightly different (more complicated) definitions in [6,8], where it was introduced. This is because we are considering a simplified framework where boundary meshlines have full multiplicity and the internal meshlines are simple in the LR meshes.

Since the introduction of LR B-splines, in order to reduce the approximation error, several refinement strategies have been introduced to automatically refine the LR mesh where it is needed. The best known and used are called minimum span, full span, structured mesh, all introduced in [29], hierarchical LR mesh [6], effective grading [8] and Non-Nested-Support-Structured ($\mathrm{N_2S_2}$) mesh [7]. In this paper we focus on the last as it is the only strategy among them which has the $\mathrm{N_2S}$ property and allows for

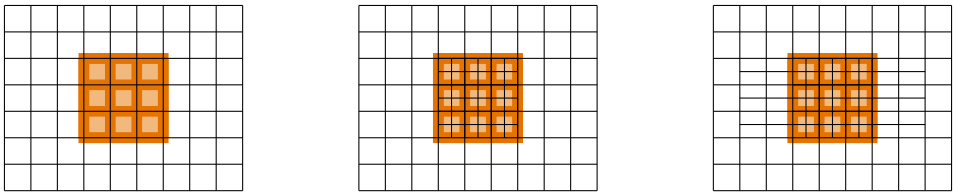


Fig. 2. N_2S_2 refinement of a marked B-spline. Assume bidegree $\mathbf{p} = (2, 2)$. In figure (a) we have picked a B-spline on a tensor mesh to be refined. Its support and associated tensor mesh are highlighted with colours. The first step of each iteration of the N_2S_2 refinement is to halve all the non-empty knot intervals, in both directions for isotropic refinement, only in one of the two for anisotropic refinement. This step is pictured in figure (b), in the isotropic case. The next step re-instate the N_2S property on the new mesh by suitably extending the vertical or the horizontal meshlines on the new mesh, in order to prevent nesting of the LR B-spline supports. In figure (c) we show the result of this second step, when extending the horizontal meshlines. Figure (c) is also the final mesh produced by the N_2S_2 refinement procedure.

anisotropic refinements. The local linear independence guarantees the polynomial reproduction [7, Proposition 4.4], a feature that we shall exploit in the surface reconstruction scheme presented in Section 4.

The N_2S_2 mesh strategy is a function-based strategy, i.e., the input is a set of LR B-splines that need to be refined rather than a set of mesh boxes, and it is composed of two steps. In the first step, we halve all the non-empty knot intervals of the selected LR B-splines, which simply results in a dyadic refinement of all the boxes in the tensor meshes associated to the marked LR B-splines. In case we want to refine (some of) the LR B-splines anisotropically, we would halve only the knot intervals in one of the two directions. This first step may create nesting in the LR mesh, that is, we may have a pair of LR B-splines defined on simple knots whose supports are one inside the other. The second step re-instate the N_2S property. If B_1 is nested in B_2 we extend all the meshlines of $\mathcal{M}_{B_1}$ in one of the two directions, to cross entirely $\operatorname{supp} B_2$. This eliminates the nesting of B_1 as B_2 is refined into other LR B-splines. However, solving the nesting of B_1 may trigger nesting in other LR B-splines. Anyways, it has been show [7, Theorem 3.5] that this nesting propagation ends in a finite number of iterations. Fig. 2 outlines the steps of the N_2S_2 refinement of a B-spline.

In Section 4, we shall see how to generate an LR mesh according to the jump estimates described in Section 3. We then present a discrete quasi-interpolation method formulated in the related LR spline space. More precisely, given the scattered data-sites X and the corresponding set of values $f(X)$, we shall define a spline, denoted as $\mathfrak{Q}f$, in the space spanned by the LR B-spline set $\mathcal{L}$ defined on the created LR mesh, which is a good approximation of the data in some sense, e.g., with respect to the Root-Mean-Square Error (RMSE). $\mathfrak{Q}f$ has therefore expression

$$\mathfrak{Q}f(x, y) := \sum_{B\in\mathcal{L}} q_B B(x, y), \tag{1}$$

for some set of coefficients $\{q_B\}_{B\in\mathcal{L}}$. As the LR mesh will be finer in proximity of large variations of the scattered data (jumps), there will be more basis functions where it is needed to capture the behaviour of the surface, resembling the outcome that one would get refining the mesh based on an error indicator.

Each coefficient q_B is computed by solving a local polynomial least squares problem with the addition of a smoothing correction term, within the support of B. As we shall prove in Proposition 4.6 at the end of Proposition 4.6, for degrees $\mathbf{p} \geq (1, 1)$, the resulting QI shall have linear precision under the N_2S property of the mesh. We underline that such reproduction of linear polynomials is essential for an accurate representation of flat areas in the reconstructed surface, as it prevents the occurrence of artefacts, such as bumps. On the other hand, the smoothing term considered in our local scheme hinders the achievement of an higher order polynomial reproduction and consequently it reduces the convergence order of the method. However, it ensures the existence of the QI and moreover provides an overall more gentle and reasonable behaviour of the surface reducing also the risk of overfitting. For real world applications, these aspects should be favoured compared to the convergence rate, as often data amount and density cannot be changed so that no convergence can be achieved at all.

2.2. Fault detection

Let $\mathcal{F}^O$ and $\mathcal{F}^G$ be unknown ordinary and gradient, respectively, fault curves in Ω. Setting $\mathcal{F} := \mathcal{F}^O \cup \mathcal{F}^G$, let $f : \Omega \to \mathbb{R}$ be a function on Ω that is smooth (at least C^2) in $\Omega \setminus \mathcal{F}$ and with finite jump discontinuities across the curves of $\mathcal{F}^O$. Similarly, let $\nabla f : \Omega \setminus \mathcal{F} \to \mathbb{R}$ be the gradient of f and assume that it has finite jump discontinuities across the curves of $\mathcal{F}^G$. In this section we summarize the available strategies for the direct detection of the ordinary and gradient fault curves in Ω, that is, $\mathcal{F}^O$ and $\mathcal{F}^G$, from the scattered evaluation of f. According to [17], this is done by defining two indicators based on Minimal Numerical Differentiation Formulas (MNDFs).

MNDFs, introduced in [30], generalize finite difference formulas: they approximate the value of any linear differential operator applied to multivariate functions by using their values at some scattered data points, as follows. Let D be a bivariate linear differential operator of order k, i.e.,

$$Df(\boldsymbol{x}) := \sum_{\alpha\in\mathbb{N}^2, |\alpha|\leq k} c_\alpha(\boldsymbol{x})\partial^\alpha f(\boldsymbol{x}), \quad \text{with} \quad \partial^\alpha f := \frac{\partial^\alpha f}{\partial x_1^{\alpha_1}\partial x_2^{\alpha_2}}. \tag{2}$$

Given $\boldsymbol{x} \in \Omega$ and a proximity set of points $X_{\boldsymbol{x}} := \{\boldsymbol{x}_i, i = 1, \dots, N_{\boldsymbol{x}}\} \subset \Omega$ and the corresponding values $f(X_{\boldsymbol{x}})$, a numerical differentiation formula is an approximation of $Df(\boldsymbol{x})$ of the form

$$\widehat{D}f(\boldsymbol{x}) := \sum_{i=1}^{N_{\boldsymbol{x}}} w_i f(\boldsymbol{x}_i), \tag{3}$$

where the weights w_i depend on $\boldsymbol{x}$ and are chosen such that the formula is exact for any function in a certain finite dimensional space. A natural choice is requiring that (3) is exact whenever f belongs to the space Π_q of bivariate polynomials of order less than or equal to q (i.e., of total degree $q-1$). When $q > k$ and f is sufficiently smooth, it can be proved that the error of the formula goes to zero as the diameter of $X_{\boldsymbol{x}}$ goes to zero [30, Proposition 4]. Of course we need $N_{\boldsymbol{x}} \geq q(q+1)/2$ in order to be able to find a formula of exactness order q. However, if $N_{\boldsymbol{x}} > q(q+1)/2$ the formula is not unique, and then it is also required that the weights minimize the weighted ℓ_2-(semi)norm, that is, the weight vector $\bar{\bar{\mathbf{w}}}$ should verify

$$\|\bar{\bar{\mathbf{w}}}\|_{2,\mu} = \operatorname{argmin}\{\|\mathbf{w}\|_{2,\mu} : Dp(\boldsymbol{x}) = \widehat{D}p(\boldsymbol{x}) \text{ for any } p \in \Pi_q\} \text{ with } \|\mathbf{w}\|_{2,\mu} := \left(\sum_{i=1}^{N_{\boldsymbol{x}}} w_i^2 \|\boldsymbol{x}_i - \boldsymbol{x}\|_2^{2\mu}\right)^{\frac{1}{2}}.$$

We denote by ℓ_2-MNDFs the MNDFs with this constraint on the weights. In the following, for simplicity we will omit the double overline.

For each point $\boldsymbol{x}_i \in X$, let us consider the ℓ_2-MNDF $\widehat{\nabla} f(\boldsymbol{x}_i)$ approximating the gradient $\nabla f(\boldsymbol{x}_i)$ with polynomial exactness of order $q = 2$, and obtained setting $\mu = 3$

$$\widehat{\nabla} f(\boldsymbol{x}_i) := \sum_{j \in J_i} \mathbf{w}_j f(\boldsymbol{x}_j), \tag{4}$$

using the points of the proximity set $X_i := \{\boldsymbol{x}_j : j \in J_i\} \subseteq X$ consisting of the $N_i := 2 \dim \Pi_q$ points of X closest to $\boldsymbol{x}_i$. Note that each $\mathbf{w}_j := (w_{1,j}, w_{2,j})$ is a vector of 2 weights, one for each component of the gradient. We define the ordinary fault indicator at $\boldsymbol{x}_i$ as

$$I_\nabla(\boldsymbol{x}_i, X_i) := \frac{\|\widehat{\nabla} f(\boldsymbol{x}_i)\|_2}{\left\| \sum_{j \in J_i} |\mathbf{w}_j| \, \|\boldsymbol{x}_j - \boldsymbol{x}_i\|_2 \right\|_2}, \tag{5}$$

where $|\mathbf{w}_j| := (|w_{1,j}|, |w_{2,j}|)$. Similarly, the gradient fault indicator is defined as

$$I_\Delta(\boldsymbol{x}_i, X_i) := \frac{|\widehat{\Delta} f(\boldsymbol{x}_i)|}{\sum_{j \in J_i} |w_j| \, \|\boldsymbol{x}_j - \boldsymbol{x}_i\|_2^2}, \tag{6}$$

where

$$\widehat{\Delta} f(\boldsymbol{x}_i) := \sum_{j \in J_i} w_j f(\boldsymbol{x}_j) \tag{7}$$

is the ℓ_2-MNDF approximating $\Delta f(\boldsymbol{x}_i)$ with polynomial exactness order $q = 3$. It can be shown by analysing its asymptotic behaviour that $I_\nabla(\boldsymbol{x}_i, X_i)$ tends to take larger values for $\boldsymbol{x}_i$ close to an ordinary fault and, similarly, $I_\Delta(\boldsymbol{x}_i, X_i)$ tends to take larger values if $\boldsymbol{x}_i$ is close to an ordinary or to a gradient fault [16,17].

Considering these behaviours, the sets

$$F^G(\alpha^G, X) := \{\boldsymbol{x}_i \in X : I_\Delta(\boldsymbol{x}_i, X_i) > \alpha^G\}, \qquad F^O(\alpha^O, X) := \{\boldsymbol{x}_i \in X : I_\nabla(\boldsymbol{x}_i, X_i) > \alpha^O\},$$

shall contain the points close to ordinary or gradient faults and close to only ordinary faults, respectively. The following detection algorithm was introduced in [17], using these two sets and two input parameters, denoted as C^O and C^1.

1. Set $\alpha_1^G = \text{median}(\{I_\Delta(\boldsymbol{x}_i, X_i) : \boldsymbol{x}_i \in X\})$, and compute $F^G(\alpha_1^G, X)$.
2. Set $\alpha_2^G = C^G \cdot \text{median}(\{I_\Delta(\boldsymbol{x}_i, X_i) : \boldsymbol{x}_i \in F^G(\alpha_1^G, X)\})$, and compute $F := F^G(\alpha_2^G, F^G(\alpha_1^G, X))$.
3. Set $\alpha_1^O = \text{median}(\{I_\nabla(\boldsymbol{x}_i, X_i) : \boldsymbol{x}_i \in X\})$, and compute $F^O(\alpha_1^O, X)$.
4. Set $\alpha_2^O = C^O \cdot \text{median}(\{I_\nabla(\boldsymbol{x}_i, X_i) : \boldsymbol{x}_i \in F^O(\alpha_1^O)\})$, and compute $F^O(\alpha_2^O, F^O(\alpha_1^O, X))$.
5. Return the sets:

$$F, \qquad F^O := F \cap F^O(\alpha_2^O, F^O(\alpha_1^O, X)), \qquad F^G := F \setminus F^O, \tag{8}$$

with points close to ordinary or gradient faults, F, points close to ordinary faults, F^O, and points close to gradient fault, F^G.

Concerning the two input parameters, we assume $C^O \geq 1$ and $C^G \geq 1$, since for reasonable sampling strategies the majority of the points in X are not close to faults.

The points of F are contained in strip shaped areas of certain thickness, along the true fault curves in Ω. The exact location of the latter can be betted identified from F if we apply a narrowing technique to each $\boldsymbol{x} \in F$, which defines a corresponding point $\boldsymbol{x}_N \in \Omega$ nearer to the true fault curve located inside the strip shaped area where $\boldsymbol{x}$ lies in. We denote by F_N the resulting narrowed set. As already done in [17], F_N is obtained by using the technique based on the computation of local least squares approximations described in [31]. Essentially, for each point $\boldsymbol{x} \in F$, we collect all the points of X included in a ball of suitable radius centred at $\boldsymbol{x}$,

and compute the local linear least squares approximation for these subset of points. We then define a local system of coordinates where the x axis has the direction, denoted by $\mathbf{d}(\boldsymbol{x}_N)$, of the just obtained linear approximation. In this new system of coordinates, we compute a quadratic least squares approximation and then the projection of $\boldsymbol{x}$ on it, which is $\boldsymbol{x}_N$. Therefore, beside $\boldsymbol{x}_N$, such narrowing process naturally provides also an associated local direction $\mathbf{d}(\boldsymbol{x}_N)$, which can be considered as an approximation of the tangent to the true fault curve at its point closest to $\boldsymbol{x}_N$. Note that F_N is not contained in X and hence we have no access to the values in $f(F_N)$ which will make the computation of the jump harder.

3. Jump estimates and classification

In this section we present a new strategy to obtain directly from the scattered dataset also the information about the jump variation along each detected fault. This is clearly of interest on its own but it will be also exploited in Section 4 to ensure a considerable reduction of the degrees of freedom in the adaptive reconstruction of f. We explain first the procedure to estimate the jump in ordinary faults. Namely, for each true fault curve $\mathcal{F}^O$, we aim to estimate the corresponding jump function $\mathcal{J}^O : \mathcal{F}^O \to \mathbb{R}$, where

$$\mathcal{J}^O(\hat{\boldsymbol{x}}) := |f_+(\hat{\boldsymbol{x}}) - f_-(\hat{\boldsymbol{x}})|\,, \quad \forall\, \hat{\boldsymbol{x}} \in \mathcal{F}^O\,, \tag{9}$$

with $f_\pm(\hat{\boldsymbol{x}}) = \lim_{\boldsymbol{x}\in\Omega^O_\pm, \boldsymbol{x}\to\hat{\boldsymbol{x}}} f(\boldsymbol{x})$, and $\bar{\Omega}^O_+ \cup \bar{\Omega}^O_- = \Omega$, $\Omega^O_+ \cap \Omega^O_- = \varnothing$, $\mathcal{F}^O \subseteq \bar{\Omega}^O_+ \cap \bar{\Omega}^O_-$. Given $R > 0$, we define

$$G_R = G_R(\hat{\boldsymbol{x}}) := \sup\{\|\nabla f(\boldsymbol{x})\|_2 \ :\ \|\boldsymbol{x} - \hat{\boldsymbol{x}}\| \le R \wedge \boldsymbol{x} \notin \mathcal{F}^O\},$$

as first step, we prove the following proposition.

Proposition 3.1. *Let us assume $\hat{\boldsymbol{x}} \in \mathcal{F}^O$ and $\boldsymbol{x} \in X \cap \Omega^O_+$ with the segment $s(\boldsymbol{x}, \hat{\boldsymbol{x}})$ fully contained in Ω^O_+ and let us define $\rho := \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|$. Furthermore, let $R > \rho$ be a positive constant such that there exist $\boldsymbol{x}_1 \in X \cap \Omega^O_+$ and $\boldsymbol{x}_2 \in X \cap \Omega^O_-$ with $\rho \le r_i := \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}\|_2 \le R, i = 1, 2$. Then, if $\mathcal{J}^O(\hat{\boldsymbol{x}}) \ge 4RG_R$ and the segments $s(\boldsymbol{x}_i, \hat{\boldsymbol{x}}), i = 1, 2$ are fully contained in Ω^O_+ and Ω^O_-, respectively, it holds*

$$\frac{|f(\boldsymbol{x}_2) - f(\boldsymbol{x})|}{r_2} \ge \frac{|f(\boldsymbol{x}_1) - f(\boldsymbol{x})|}{r_1}\,. \tag{10}$$

Proof. Setting $\mathbf{v} := (\boldsymbol{x} - \hat{\boldsymbol{x}})/\rho$, we can write

$$f(\boldsymbol{x}) = f_+(\hat{\boldsymbol{x}}) + \rho\,(\nabla f(\boldsymbol{\xi}) \cdot \mathbf{v})\,,$$

for some $\boldsymbol{\xi} \in s(\boldsymbol{x}, \hat{\boldsymbol{x}})$, because of the assumption on $s(\boldsymbol{x}, \hat{\boldsymbol{x}})$. In a similar manner, defining $\mathbf{v}_i := (\boldsymbol{x}_i - \hat{\boldsymbol{x}})/r_i, i = 1, 2$, we have

$$f(\boldsymbol{x}_1) = f_+(\hat{\boldsymbol{x}}) + r_1\left(\nabla f(\boldsymbol{\xi}_1) \cdot \mathbf{v}_1\right), \qquad f(\boldsymbol{x}_2) = f_-(\hat{\boldsymbol{x}}) + r_2\left(\nabla f(\boldsymbol{\xi}_2) \cdot \mathbf{v}_2\right),$$

with $\boldsymbol{\xi}_i \in s(\boldsymbol{x}_i, \hat{\boldsymbol{x}})$. Then, it follows that

$$\begin{aligned} |f(\boldsymbol{x}_1) - f(\boldsymbol{x})| &= |r_1\left(\nabla f(\boldsymbol{\xi}_1) \cdot \mathbf{v}_1\right) - \rho\,(\nabla f(\boldsymbol{\xi}) \cdot \mathbf{v})\,| && \le G_R(r_1 + \rho) \\ |f(\boldsymbol{x}_2) - f(\boldsymbol{x})| &= |f_-(\hat{\boldsymbol{x}}) - f_+(\hat{\boldsymbol{x}}) + r_2\left(\nabla f(\boldsymbol{\xi}_2) \cdot \mathbf{v}_2\right) - \rho\,(\nabla f(\boldsymbol{\xi}) \cdot \mathbf{v})\,| && \ge \mathcal{J}^O(\hat{\boldsymbol{x}}) - G_R(r_2 + \rho). \end{aligned}$$

Thus, we have

$$\frac{|f(\boldsymbol{x}_2) - f(\boldsymbol{x})|}{r_2} \ge \frac{\mathcal{J}^O(\hat{\boldsymbol{x}})}{R} - 2G_R \ge 2G_R \quad \text{and} \quad \frac{|f(\boldsymbol{x}_1) - f(\boldsymbol{x})|}{r_1} \le 2G_R. \quad \square$$

In principle, we do not have access to $\mathcal{F}^O$. We have just available from the detection and narrowing phases, respectively, two discrete sets of points: $F^O \subset X$ and the corresponding set F^O_N, which is closer to $\mathcal{F}^O$. Furthermore, the value of f is not assigned at the points in F^O_N. Relying on the available information and the result of Proposition 3.1, we have formulated Algorithm 1 to assign a value $J^O(\boldsymbol{x}_N)$ to each point $\boldsymbol{x}_N \in F^O_N$. Since $\boldsymbol{x}_N$ is obtained by the narrowing process [31], and therefore represents an approximation of a point belonging to the true fault $\mathcal{F}^O$, $J^O(\boldsymbol{x}_N)$ provides an estimate of the jump $\mathcal{J}^O(\hat{\boldsymbol{x}})$ at the point $\hat{\boldsymbol{x}} \in \mathcal{F}^O$ nearest to $\boldsymbol{x}_N$. The algorithm works as follows. $J^O(\boldsymbol{x}_N)$ is a weighted mean of differences $|f(\boldsymbol{y}) - f(\boldsymbol{x})|$, with $\boldsymbol{x} \in X$ denoting the point that generates $\boldsymbol{x}_N$ after the narrowing procedure and $\boldsymbol{y}$ varying in the set defined as follows. Assume, without loss of generality, that $\boldsymbol{x}$ belongs to Ω^O_+. Then, the points $\boldsymbol{y}$ are the points of X in $X_{\boldsymbol{x}_N} \cap \Omega^O_-$ with $X_{\boldsymbol{x}_N}$ a small annulus centred at $\boldsymbol{x}_N$, that is, those points $\boldsymbol{y}$ in X verifying $\|\boldsymbol{x} - \boldsymbol{x}_N\|_2 =: \rho \le \|\boldsymbol{y} - \boldsymbol{x}_N\|_2 \le R$ for some small R. In order to select points in the annulus only on the other side of the fault, first we compute the quantities $|f(\boldsymbol{y}) - f(\boldsymbol{x})|/\|\boldsymbol{y} - \boldsymbol{x}_N\|_2$ for all the points $\boldsymbol{y}$ of X in the whole annulus $X_{\boldsymbol{x}_N}$ and we sort them in a decreasing ordered vector $\mathbf{q}$. Then, for Proposition 3.1, we know that the higher values in the entries of $\mathbf{q}$ should correspond to points on the other side of the fault with respect to $\boldsymbol{x}$, that is, in Ω^O_-. Hence, we identify the index k for which the component q_{k+1} is significantly lower than the previous q_k. Such index is likely the index of the last point $\boldsymbol{y}$ on the other side of the fault. The others should be on the same side of $\boldsymbol{x}$, i.e., in Ω^O_+. Thus, by doing the average of the weighted difference $|f(\boldsymbol{y}) - f(\boldsymbol{x})|$ only for those first k entries of $\mathbf{q}$, we make sure to consider only $\boldsymbol{y}$ in Ω^O_-. Concerning the weights, these are chosen such that they increase as the distance between $\boldsymbol{y}$ and $\boldsymbol{x}_N$ decreases, as closer points to $\boldsymbol{x}_N$ should contribute more to the approximation of the jump. Namely, the weights chosen are the normalizations of $w_{\boldsymbol{y}} := \|\boldsymbol{y} - \boldsymbol{x}_N\|_2^{-1}$, in order to have all of them in $(0, 1)$ and summing to 1. As final remark, we stress that in principle, we should have considered $f(\boldsymbol{x}_N)$ in the differences defining $J^O(\boldsymbol{x}_N)$. However, we do not have access to this information as $\boldsymbol{x}_N$ does not belong to X. Approximating $f(\boldsymbol{x}_N)$ as an average is also not easy as the

Algorithm 1: Jump estimate, $J = \texttt{jump\_estimate}(X, f(X), \boldsymbol{x}, \boldsymbol{x}_N, R)$

Input:

- $X, f(X)$: scattered dataset of distinct points and associated function values,
- $\boldsymbol{x}$: point of X detected as ordinary fault, i.e., in F^O,
- $\boldsymbol{x}_N$: point of F_N^O corresponding to $\boldsymbol{x}$,
- R: radius for a neighbourhood of $\boldsymbol{x}_N$, with $R > \|\boldsymbol{x} - \boldsymbol{x}_N\|_2 =: \rho$.

Output:

- $J^O(\boldsymbol{x}_N)$: estimate of the jump value $\mathcal{J}^O(\hat{\boldsymbol{x}})$ with $\hat{\boldsymbol{x}}$ being the point of $\mathcal{F}^O$ closest to $\boldsymbol{x}_N$.

Let $X_{\boldsymbol{x}_N}$ be the set of points $\boldsymbol{y} \in X$ in the annulus $\rho \le \|\boldsymbol{y} - \boldsymbol{x}_N\|_2 \le R$;

for $\boldsymbol{y} \in X_{\boldsymbol{x}_N}$

  Compute $q(\boldsymbol{y}) := \frac{|f(\boldsymbol{y}) - f(\boldsymbol{x})|}{\|\boldsymbol{y} - \boldsymbol{x}_N\|_2}$;

Sort the points in $X_{\boldsymbol{x}_N}$ so that the values $q_j := q(\boldsymbol{y}_j)$ are decreasingly ordered;

Compute the index $k := \underset{1 \le j \le |\mathbf{q}|-1}{\operatorname{argmax}}(q_j - q_{j+1})$;

Set $J^O(\boldsymbol{x}_N) := \sum_{j=1}^{k} q_j \Big/ \sum_{j=1}^{k} \|\boldsymbol{y}_j - \boldsymbol{x}_N\|_2^{-1}$;

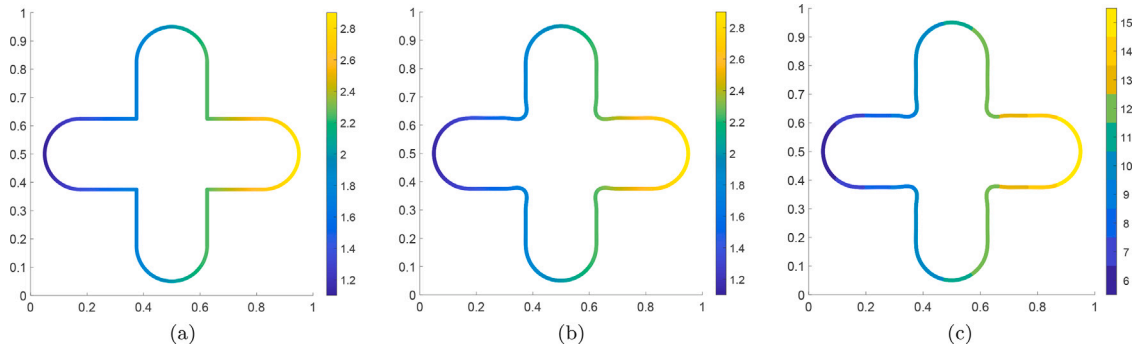


Fig. 3. Example of jump estimate and classification for an ordinary fault. We consider the sampling on one million Halton points of a piecewise surface in $[0, 1]^2$ which is zero outside of a region R, in the shape of a plus symbol, and it is the restriction of the plane $f(x, y) = 2x + 1$ inside R. In figure (a) we report the analytical jump. In figure (b) we show the fault and jump reconstructed. In figure (c) the classification of the jump when allowing $L = 15$ levels of refinements.

surrounding points in X could belong to either sides of the fault. On the other hand, $f(\boldsymbol{x})$ is a good approximation of $f(\boldsymbol{x}_N)$ for ρ sufficiently small. Fig. 3(b) shows the outcome of such Algorithm 1 in a test case which can be compared with the corresponding exact jump distribution presented in Fig. 3(a).

For gradient faults, we are interested in approximating the jump of the normal derivative of f across any fault $\mathcal{F}^G$, that is, the function $\mathcal{J}^G : \mathcal{F}^G \to \mathbb{R}$

$$\mathcal{J}^G(\hat{\boldsymbol{x}}) := |\partial_{\hat{\mathbf{n}}} f_+(\hat{\boldsymbol{x}}) - \partial_{\hat{\mathbf{n}}} f_-(\hat{\boldsymbol{x}})|\,, \quad \forall\, \hat{\boldsymbol{x}} \in \mathcal{F}^G\,, \tag{11}$$

where, extending the notation introduced for ordinary faults, it is $\partial_{\hat{\mathbf{n}}} f_\pm(\hat{\boldsymbol{x}}) = \lim_{\boldsymbol{x} \in \Omega_\mp^G, \boldsymbol{x} \to \hat{\boldsymbol{x}}} \partial_{\hat{\mathbf{n}}} f(\boldsymbol{x})$, with $\hat{\mathbf{n}} = \hat{\mathbf{n}}(\hat{\boldsymbol{x}})$ denoting the unit normal to the fault at $\hat{\boldsymbol{x}}$. If we had a good approximation of $\nabla f(\boldsymbol{x})$ at all the points $\boldsymbol{x}$ of X, we could just reuse Algorithm 1 simply by replacing the values of f with the estimated values of its normal derivative. In fact, since we have a local direction $\mathbf{d}(\boldsymbol{x}_N)$ of the fault at every $\boldsymbol{x}_N \in F_N^G$, we could easily estimate the jump of the normal derivative across the point $\hat{\boldsymbol{x}} \in \mathcal{F}^G$ nearest to $\boldsymbol{x}_N$ with respect to the direction $\mathbf{n}_N = \mathbf{n}_N(\boldsymbol{x}_N) := \mathbf{d}(\boldsymbol{x}_N)^\perp$, which constitutes an approximation of the normal direction $\hat{\mathbf{n}} = \hat{\mathbf{n}}(\hat{\boldsymbol{x}})$. Unfortunately, getting good approximations of the gradient near a gradient fault is a delicate task. Indeed, at any $\boldsymbol{x} \in X$, the gradient estimate given by (4) is not reliable if the adopted proximity set includes points on both sides of the fault. We then preliminary need a new way to approximate safely the gradient at least at some points of X in a neighbourhood of $\boldsymbol{x}_N$, for any $\boldsymbol{x}_N \in F_N^G$. The procedure we propose works as follows.

For every fault point $\boldsymbol{x}_N \in F_N^G$ and $R > 0$, let us define $O(\boldsymbol{x}_N)$ as the offset of the straight line $r_{\boldsymbol{x}_N}(t) : \boldsymbol{x}_N + t\mathbf{d}(\boldsymbol{x}_N)$ of width $2R/3$, i.e., the strip with axis $r_{\boldsymbol{x}_N}$ and size $2R/3$, and let $X_{\boldsymbol{x}_N} := X \cap B_R(\boldsymbol{x}_N) \setminus O(\boldsymbol{x}_N)$, see Fig. 4. We add to $X_{\boldsymbol{x}_N}$ the original detected point $\boldsymbol{x} \in X$ (i.e., before narrowing replacement) if not already contained. In $X_{\boldsymbol{x}_N}$ we have two distinct groups of points separated by the strip $O(\boldsymbol{x}_N)$, which can be easily distinguished with geometric considerations. We choose R such that there are enough points to apply the ℓ_2-MNDF formula to estimate the gradient in each point $\boldsymbol{y}$ of $X_{\boldsymbol{x}_N}$ using only points of the group to which $\boldsymbol{y}$ belongs to. Of course, for the sake of efficiency, we apply this procedure only to those points that have not a gradient value assigned yet. After

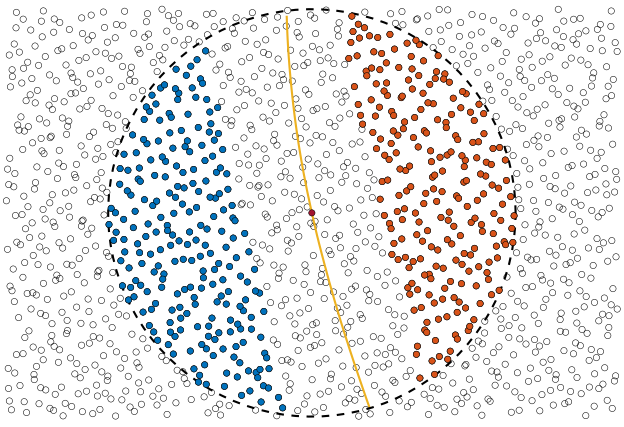


Fig. 4. Setting for the gradient estimate in an offset of the fault curve. All the dots belong to X except the central red dot. The red central marker is the narrowed point $\boldsymbol{x}_N \in F_N^G$ defined from a detected fault point $\boldsymbol{x} \in F^G$. The yellow curve is the restriction of the fault curve in the considered neighbourhood (bounded by the dashed circumference). The two coloured groups of dot markers are the two sets, composing $X_{\boldsymbol{x}_N}$, considered for the gradient estimate on the two sides of the fault curve. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

applying this algorithm to all $\boldsymbol{x}_N \in F_N^G$, a new, smaller, dataset X^G, and corresponding set of estimated gradient values $\widehat{\nabla} f(X^G)$, are produced, with

$$X^G := \bigcup_{\boldsymbol{x}_N \in F_N^G} X_{\boldsymbol{x}_N} \subseteq X.$$

Finally, we compute while running Algorithm 1 the normal derivatives with respect to $\mathbf{n}_N(\boldsymbol{x}_N)$ for every point of X^G in the neighbourhood of $\boldsymbol{x}_N \in F_N^G$. More precisely, provided the collection of local directions $\{\mathbf{d}(\boldsymbol{x}_N)\}_{\boldsymbol{x}_N \in F^G}$, X^G and $\widehat{\nabla} f(X^G)$ as inputs instead of X and $f(X)$, we change the for cycle of Algorithm 1 as follows:

Estimate the normal derivative at $\boldsymbol{x}$ as $\widehat{\partial_{\mathbf{n}_N} f}(\boldsymbol{x}) := \langle \widehat{\nabla} f(\boldsymbol{x}), \mathbf{n}_N(\boldsymbol{x}_N) \rangle$;

for $\boldsymbol{y} \in X_{\boldsymbol{x}_N}$

Estimate the normal derivative at $\boldsymbol{y}$ as $\widehat{\partial_{\mathbf{n}_N} f}(\boldsymbol{y}) := \langle \widehat{\nabla} f(\boldsymbol{y}), \mathbf{n}_N(\boldsymbol{x}_N) \rangle$;

Compute $q(\boldsymbol{y}) := \frac{|\widehat{\partial_{\mathbf{n}_N} f}(\boldsymbol{y}) - \widehat{\partial_{\mathbf{n}_N} f}(\boldsymbol{x})|}{\|\boldsymbol{y} - \boldsymbol{x}_N\|_2}$;

Fig. 5(b) shows the estimate of the jump in the normal derivative along a gradient fault. The analytic jump is reported in Fig. 5(a).

Remark 3.2. The proposed approximation of the gradient relies on the fact that the restriction of the fault curve within the neighbourhood $B_R(\boldsymbol{x}_N)$ is inside the stripe $O(\boldsymbol{x}_N)$. Thereby, the two identified groups of points used for the gradient estimates are one on one side and one on the other side of the fault. This means that R has to be small enough, especially close to points of larger curvature in the true fault, to verify this assumption. On the other hand, a smaller R requires a dense enough point cloud in order to have enough points to apply the ℓ_2-MNDF. Therefore, we assume that the point cloud is sufficiently dense, especially along the gradient faults.

Remark 3.3. As already mentioned, we may need to force the inclusion of $\boldsymbol{x}$ in $X_{\boldsymbol{x}_N}$, if $\boldsymbol{x}$ lies within $O(\boldsymbol{x}_N)$. The side of the line $r_{\boldsymbol{x}_N}$ where $\boldsymbol{x}$ lies in decides which sub-group of points in $X_{\boldsymbol{x}_N}$ we shall use for the ℓ_2-MNDF. In the worst case scenario, if $\boldsymbol{x}$ is sufficiently close to the true fault curve $\mathcal{F}^G$, it would have belonged to the opposite group of points in $X_{\boldsymbol{x}_N}$ if the sides were defined with respect to $\mathcal{F}^G$ rather than $r_{\boldsymbol{x}_N}$. Hence, the gradient estimate for $\boldsymbol{x}$ is not reliable in these cases. However, if we use the set $X_{\boldsymbol{x}_N} \setminus \{\boldsymbol{x}\}$ for the ℓ_2-MNDF, the effect is basically to extend to $\boldsymbol{x}$ the restriction of the function to the opposite side of the gradient fault. In the interest of estimating the jump in the normal derivative using Algorithm 1, we only need a coherent computation of a gradient approximation, that is, using only points on one side of the true fault. Thus, by excluding $\boldsymbol{x}$, the value $\widehat{\nabla} f(\boldsymbol{x})$ becomes an estimate of the gradient expression on the opposite side of the fault with respect to $\boldsymbol{x}$. This is enough for having a reliable jump estimate from Algorithm 1, despite the original wrong grouping of $\boldsymbol{x}$ in $X_{\boldsymbol{x}_N}$.

3.1. Jump classification

In order to consider the computed estimates into an adaptive refinement scheme, the jump values assigned to the fault points have to be clustered into classes. Each class corresponds to a maximal level that should be applied in the refinement process. We highlight that the straightforward strategy of uniformly distributing the jump values is not an option as, in case of constant jumps,

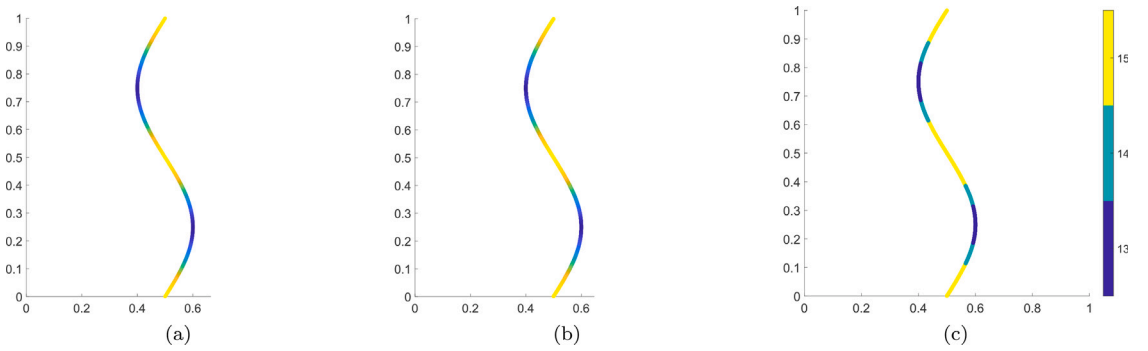


Fig. 5. Example of jump estimate and classification for a gradient fault. We consider the sampling on one million Halton points in $[0,1]^2$ of the function $f(x,y) = |x - \frac{1}{2} - \frac{1}{10}\sin(2\pi y)|$, which is only C^0 along the gradient fault $x = \frac{1}{2} + \frac{1}{10}\sin(2\pi y)$. The analytical jump in the normal derivative across the fault is $g(y) = 2\sqrt{1 + \frac{\pi^2}{25}\cos^2(2\pi y)}$ and it is reported in figure (a). In figure (b) we show the reconstructed fault and jump. In figure (c) the classification of the jump when allowing $L = 15$ levels of refinements. Despite the availability of 15 classes, the detected points are gathered only in the last three by the jump classification algorithm. This is due to the fact that the range of the jump estimate is relatively small with respect to its magnitude and therefore it is reckon as fairly "close to constant" by the classification algorithm.

the estimates would hardly be exactly constant as well and points that should belong to the same class would rather be spread over all the available classes. Furthermore, if the range of values of the jump is not very large compared to the overall magnitude, only the last few classes should be invoked, as the fineness of the mesh should be close to the maximal everywhere along the fault. With these considerations in mind, we propose the procedure schematized in Algorithm 2, which has proved to be reliable in these aspects, see Figs. 3,5 and Figs. 7–11 in the numerical tests.

Algorithm 2: Jump classification, $C^J = \texttt{jump\_classification}(F_N, J, L, \bar{L})$

Input:

- F_N, J: set of narrowed points and related jump estimates,
- $L, \bar{L}$: Maximal and minimal class indices.

Output:

- C^J: set of jump classes of indices $\ell \in \{\bar{L}, \dots, L\}$ related to F_N.

Initialize $\hat{L} := \bar{L}$, $J_m := \min_{\boldsymbol{x}\in F_N} J(\boldsymbol{x}_N)$ and $J_M := \max_{\boldsymbol{x}\in F_N} J(\boldsymbol{x}_N)$;

Define $\hat{J}_{\text{old}} := 0$ and

$$\hat{J}_{\text{new}} := \frac{J_M - J_m}{J_M}\left(\frac{1}{|F_N|}\sum_{\boldsymbol{x}_N\in F_N} J(\boldsymbol{x}_N)\right); \tag{12}$$

while $F_N \neq \emptyset$ *and* $\hat{L} \geq \bar{L}$ **do**

  Update $L \leftarrow \max\left\{L - \left\lfloor \hat{J}_{\text{old}}/\hat{J}_{\text{new}} \right\rfloor, \hat{L}\right\}$;

  Let $D := L - \hat{L}$;

  if $D > 0$ **then**

    Define $v_D := J_M - (J_M - \hat{J}_{\text{new}})/D$;

  else

    Define $v_D := 0$;

  Set $C^J(\boldsymbol{x}_N) = L$ for all $\boldsymbol{x}_N \in F_N$ with $J(\boldsymbol{x}_N) \geq v_D$ and erase $\boldsymbol{x}_N$ from F_N, $F_N \leftarrow F_N \setminus \{\boldsymbol{x}_N\}$;

  if $F_N \neq \emptyset$ **then**

    Update $\hat{J}_{\text{old}} \leftarrow \hat{J}_{\text{new}}$;

    Update minimal and maximal jumps, J_m and J_M, and $\hat{J}_{\text{new}}$ as in Equation (12);

    Update $\hat{L} \leftarrow \left\lfloor \hat{J}_{\text{new}}/J_M L \right\rfloor$;

if *still* $F_N \neq \emptyset$ **then**

  Define $v_0 := 0$ and $v_k := \hat{J}_{\text{old}} + \frac{k-1}{D}(J_M - \hat{J}_{\text{old}})$ for $k = 1, \dots, D$;

  Set $C^J(\boldsymbol{x}_N) := \hat{L} + k - 1$ with $\boldsymbol{x}_N : v_{k-1} \leq J(\boldsymbol{x}_N) < v_k$ for $k = 1, \dots, D$;

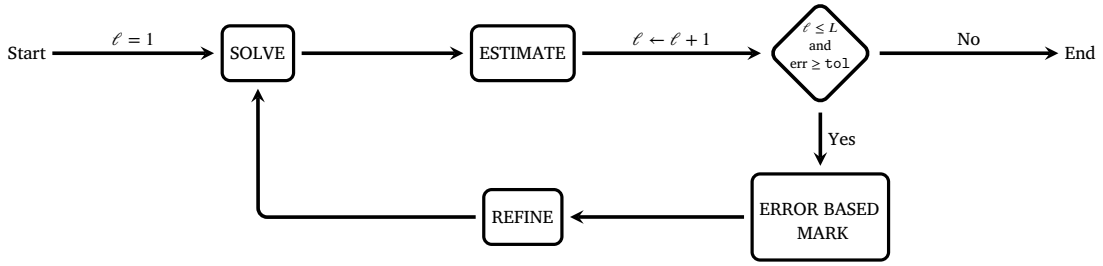
The classification can be summarized as follows. Let F_N be either F_N^O or F_N^G and J be the corresponding set of estimated jump values, i.e., J^O or J^G. We identify the points whose jump estimates should be at the maximal level L. Once they have been clustered, we remove them from F_N. We update by suitably lowering the maximal level L and we determine the points, in the remaining collection, that should be in the new maximal level. We iterate this routine until nothing is left in the narrowed points collection

or when a minimal level $\bar{L}$ has been reached for the classification. More precisely, both the current maximal level L and the points whose jump values are assigned to class L are identified by a quantity $\widehat{J}_{\text{new}}$, defined in (12), that is linear in the relative range size and in average of the estimated jump values in the remaining points of F_N throughout the algorithm. In particular, the smaller the range size and/or average is, the lower will be $\widehat{J}_{\text{new}}$. As reported in Algorithm 2, after each assignment of class and extraction of such just assigned points from F_N, L is updated as $L \leftarrow \max\left\{L - \left\lfloor \widehat{J}_{\text{old}}/\widehat{J}_{\text{new}} \right\rfloor, \widehat{L}\right\}$ where $\widehat{J}_{\text{old}}$ is the $\widehat{J}_{\text{new}}$ that was computed before the reduction of F_N, in the previous class assignment. Whereas, the points whose assigned class will be the current L are those $\boldsymbol{x}_N \in F_N$ whose jump verifies $J(\boldsymbol{x}_N) \geq v_D := J_M - {}^{(J_M - \widehat{J}_{\text{new}})}/_{D}$, with $D := L - \widehat{L}$. $\widehat{L}$ is also a varying quantity that depends on the values of $\widehat{J}_{\text{new}}$ and the maximum J_M and provides, through D, the range of classes that should be considered after each reduction of F_N. At first it is initialized as $\bar{L}$, the minimal class allowed. During the process it changes according to (i) how large the relative jump range is in the remaining collection and (ii) to how much the majority of fault points in the current F_N have a jump value far from the jump maximum. If any of these two factors is small, then basically all of the remaining points should be classified at the new maximal level L. $\widehat{L}$ and its relation to L are defined in such a way to achieve this result.

Remark 3.4. We stress that the classification provided by Algorithm 2 does not necessarily involve the classes from L down to $\bar{L}$. It may very well spread over less classes, depending on L, $\bar{L}$, the range of values of the jump and its magnitude.

4. Quasi-interpolation of scattered data

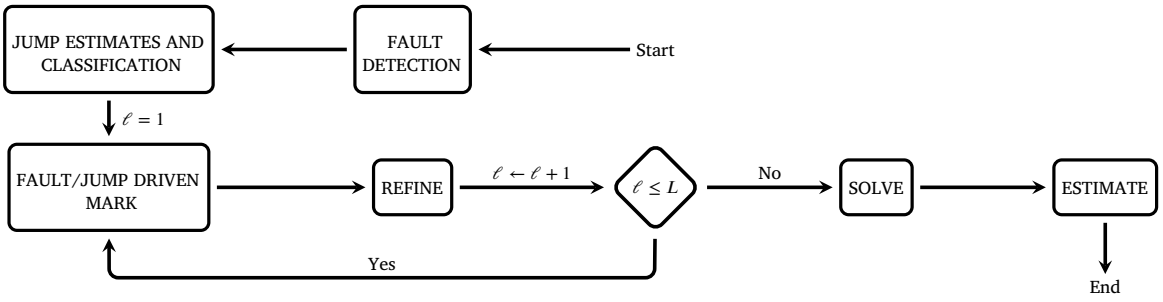
In this section we describe a quasi-interpolation method exploiting the LR B-splines to allow adaptive refinements. The classical adaptivity cycle for numerical approximations is based on the achievement of a given tolerance by the error, computed in some norm. Assuming to allow up to L refinement iterations, the structure of the procedure is the following:



More precisely, for each iteration ℓ, a numerical approximation in the module SOLVE is computed, then the error is estimated and ℓ is updated. As long as ℓ is less than the maximal allowed loop iteration L and the error is still above the input tolerance, one keeps marking those elements (or basis functions) contributing the most to the error. Consequently, these cells (or functions) are refined and then the cycle starts over by computing the new approximation.

However, the SOLVE module as well as the estimation of the error, for large size problems, are severely time-consuming. For this reason, the adaptivity cycle might be terminated prematurely, preventing it to reach the given tolerance for the error or the maximal loop iteration.

On the contrary, in the fault and jump driven strategies, the SOLVE module and error estimation are done only once at the end of the process. In fact, the scheme of the procedure is the following



The tremendous lightening of the approximation construction constitutes the main advantage and motivation for our approach. Note that the fault or jump driven adaptive cycle could also be used in combination with the standard error based cycle. We could in fact skip L intermediate approximations and error evaluations with the former, and then check the quality of the surface constructed. Based on that, we further refine where needed, possibly by lowering the tolerance and provided a sufficiently high density of the point cloud. If required, an higher accuracy could also be attained by raising the maximum level L and/or clustering differently the jump estimates in refinement levels by raising the parameter $\bar{L}$. On the contrary, if all the dataset points are within the tolerance at the end of the procedure one could consider to lower such $\bar{L}$ and L to save more degrees of freedom while still achieving an acceptable approximation.

4.1. Jump estimates guided marking

Once we have divided the jump estimates into classes, as explained in Remark 3.4, we have a set C^J, associated to the points in F_N, whose elements are indices ℓ in the range $\{1, \dots, L\}$, with L the maximum number of refinements allowed on the mesh.

Provided an LR B-spline basis $\mathcal{L}$, obtained after $\bar{\ell}$ refinements of the mesh with the N_2S_2 strategy from an initial tensor mesh, we now describe the marking procedure to refine further such basis, according to the jump classification C^J. Basically, an LR B-spline in $\mathcal{L}$ has to verify two conditions in order to be marked. First, we ask that the majority of the points of F_N, in the support of the LR B-spline being analized, have associated classes larger than $\bar{\ell}$. Then, we check if there are "enough" data points of the global dataset X in its support. This second requirement is needed to assemble a local least squares system and construct the approximation of the point cloud, as it will be explained in Proposition 4.6. It is clear that the quality of the local problem depends on the number of points of X in the support. The minimal number $m \geq 3$ of data considered acceptable for this purpose is provided as input by the user.

More precisely, the marking procedure, schematized in Algorithm 3, works as follows. For each LR B-spline $B \in \mathcal{L}$, let $F_B := F_N \cap \operatorname{supp} B \subseteq F_N$. Note that possibly F_B is the empty set, in this case B will not be marked for refinement. Otherwise, we split F_B into groups $F_B^{\ell_1}, \dots, F_B^{\ell_{n_B}}$ with $F_B^{\ell_k} := \{\boldsymbol{x}_N \in F_B \ : \ C^J(\boldsymbol{x}_N) = \ell_k \text{ for } k = 1, \dots, n_B\}$, with n_B the number of different classes to which the points in F_B belong to. Then, we define the sets of those points in F_B with class ℓ over and under $\bar{\ell}$, respectively, as

$$F_B^O := \bigcup_{\ell_j \geq \bar{\ell}} F_B^{\ell_j}, \qquad F_B^U := \bigcup_{\ell_j < \bar{\ell}} F_B^{\ell_j}. \tag{13}$$

Then, if the cardinality $|F_B^U|$ is strictly greater than $|F_B^O|$, that is, the majority of points in F_B has class strictly less than $\bar{\ell}$, the LR B-spline B is not refined. Otherwise, we look at the subset $X_B := X \cap \operatorname{supp} B \subseteq X$. If X_B contains at least m points, then the LR B-spline B is marked for refinement.

Algorithm 3: Jump estimates driven marking, $\mathcal{L}^M = \texttt{marking}(\mathcal{L}, X, F_N, C^J, m, \bar{\ell})$

Input:

- $\mathcal{L}$: LR B-spline set,
- X: Dataset of scattered data,
- F_N, C^J: set of narrowed points and corresponding set of jump classes,
- m: Minimal number of points of X to have in an LR B-spline support,
- $\bar{\ell}$: Current refinement level.

Output:

- $\mathcal{L}^M$: subset of $\mathcal{L}$ of those LR B-splines marked for refinement.

for $B \in \mathcal{L}$
- Define F_B as the set of points in F_N lying in $\operatorname{supp} B$;
- **if** $F_B = \emptyset$ **then**
  - B is not marked, continue;
- Split the points of F_B into groups, according to jump class: $F_B^{\ell_1}, \dots, F_B^{\ell_{n_B}}$;
- Define the sets of points in F_B with class over and under $\bar{\ell}$, i.e., F_B^O and F_B^U as in (13);
- **if** $|F_B^O| \geq |F_B^U|$ **then**
  - Define X_B as the set of points in X lying in $\operatorname{supp} B$;
  - **if** $|X_B| \geq m$ **then**
    - Mark B for refinement, i.e., add it to $\mathcal{L}^M$;

Remark 4.1. The density of the dataset X establishes the maximal number L of refinement iterations that can be considered. Indeed, as the supports of the LR B-splines become smaller and smaller as we refine, the number of points in them decreases. Hence, if we set an L too large compared to the density of the points, it will not be reached.

Remark 4.2. Although in principle the marking algorithm ensures that we refine only LR B-splines having enough data for the local least squares problem, the N_2S property demands some propagation of the refinement at each step, possibly out of the marked region. Therefore, the recovery of the local linear independence of the basis could lead to an LR B-spline set in which some LR B-splines lack of data for the approximation of the point cloud afterwards. Despite this situation is rarely experienced, as the propagation is still limited in practice, a workaround will be proposed in Proposition 4.6 to deal with this inconvenience.

Once we have created the subset $\mathcal{L}^M \subseteq \mathcal{L}$ of the marked LR B-splines, we have to decide whether or not they should be refined anisotropically, that is, only in one of the two directions, according to the local displacement of the fault points in their supports.

As a pre-processing step, given the set F_N and the associated set of local directions, we make a characteristic vector $\mathbf{a}$ of length $|F_N|$ with values in $\{1,2,3\}$. Given $\boldsymbol{x}_N \in F_N$ and a "small" angle θ, let $\mathbf{d}(\boldsymbol{x}_N)$ be the related local direction. We have that

$$a(\boldsymbol{x}_N) := \begin{cases} 1 & \text{if } \mathbf{d}(\boldsymbol{x}_N) \text{ makes an angle less than } \theta \text{ with } \mathbf{e}_1 := (1,0), \\ 2 & \text{if } \mathbf{d}(\boldsymbol{x}_N) \text{ makes an angle less than } \theta \text{ with } \mathbf{e}_2 := (0,1), \\ 3 & \text{otherwise.} \end{cases} \tag{14}$$

Such vector $\mathbf{a}$ indicates which fault points have a local direction aligned, except for a small angle θ, with one of the two axes and which have not. Such θ is arbitrary. In the numerical tests we performed, including those of Section 5, we have set $\theta = \pi/12$ and we reckon that the fault points classified as axis-aligned are somehow reasonable.

Remark 4.3. The use of the angle θ is necessary for the identification of the axis-aligned fault points. In fact, even when the real fault is a line parallel to one of the axis, the fault detected could be not completely straight and the local directions associated to the fault points will not be all parallel to such line. Hence, it is necessary to compensate such deflections by allowing some room in the angles that the local directions can make with the axes.

As another free parameter, we can decide from which level $L^A \in \{1,\dots,\bar{L},\dots,L\}$ one should perform anisotropic refinements. Here $\bar{L} \in \{1,\dots,L\}$ is the minimal class in the jump classification C^J and so the level from which local refinements is driven by jump intensity. In fact, not necessarily we should introduce anisotropy in the refinement as soon as we start placing local insertions in the mesh where the jump classes indicate. One may require anisotropy only at the very last steps. This choice could be considered in order to capture details in the direction of the fault as well, at least for some further steps after $\bar{L}$.

With vector $\mathbf{a}$ and level L^A at hand, we decide if we should refine an LR B-spline $B \in \mathcal{L}^M$ anisotropically, according to the following procedure. First of all, we should have that the refinement step we are performing is greater than L^A, otherwise the refinement is isotropic. If this is the case, for every marked LR B-spline $B \in \mathcal{L}^M$, we consider again the subset $F_B \subseteq F_N$ as well as $\mathbf{a}_B \subseteq \mathbf{a}$ the restriction of $\mathbf{a}$ to such sub-collection. Let then F_B^k for $k = 1,2,3$ be the set of points in F_B with k as associated value in $\mathbf{a}_B$. We define $\bar{k} := \text{argmax}_{k\in\{1,2,3\}} |F_B^k|$. If $\bar{k}$ is not unique, then it is set as $\bar{k} := 3$. If $\bar{k} = 1$, B will be refined only horizontally, if $\bar{k} = 2$, only vertically and if $\bar{k} = 3$, in both directions, i.e., isotropically.

Remark 4.4. A trivial observation, which however has ultimately a strong impact on the number of degrees of freedom, is that if most of the fault points makes an angle in the range $\alpha \pm \theta$ with one of the axes, we could always apply a rotation by $-\alpha$ to the point cloud, in order then to consider axis-aligned most of them and therefore perform anisotropic refinements along a larger portion of the faults.

4.2. Solving

In this section we explain the SOLVE module, schematized in Algorithm 4, which provides the approximation in the LR spline space, given as input, by means of a quasi-interpolation (QI) technique. The quasi-interpolant here described is a local method, that is, the QI coefficient assigned to each LR B-spline is computed by solving a local problem, i.e., a problem that can be set up by using only the small sub-collection of points of the dataset lying in (or close to) the support of the LR B-spline at hand.

Before starting the description of the method, we point out a couple of remarks. First, note that as opposed to the problem of approximating analytical functions, which can be evaluated at as many points as demanded to assemble the local least squares problems, when facing scattered data approximation, the least squares systems might be not always constructed, due to the possible lack of points in the dataset and/or to the bad displacement of them, e.g., in the case of almost collinear points, which leads to ill-conditioned systems [32]. A special treatment is therefore required in these situations and it is detailed later in this section. As an overview and an insight of what we propose, when there is a lack of points, we enlarge to some extent the region where we look for data. If still there are not enough points, the assignment of the QI coefficient for that LR B-spline is not done by solving a least squares system, but rather by inheritance, that is, a weighted average of the QI coefficients computed for some of the LR B-splines defined on the previous, coarser, LR meshes. Furthermore, in order to avoid singular or nearly singular local least squares systems a smoothing correction is adopted [32]. Nevertheless, the inheritance routine might still be necessary and invoked, despite the use of such smoothing correction if the system is still badly conditioned.

Let $\mathcal{L}^A$ be the collection of all the LR B-spline sets computed so far in the adaptive cycle. Let then $\mathcal{L} \in \mathcal{L}^A$ be the latest LR B-spline set computed, that is, the set of LR B-splines on the current LR mesh and for which we want to compute the coefficients of the quasi-interpolant for a better approximation of the dataset. Let also X be the collection of scattered points forming such dataset and $f(X)$ the corresponding set of function values. For any LR B-spline $B \in \mathcal{L}$, we define X_B as the subset of points in X lying in $\text{supp}\, B := [a,b] \times [c,d]$. Let us assume, at first, that X_B has at least $m \geq 3$ points, with m a threshold set by the user, that is, we assume the amount of points in X_B acceptable to set the local least squares system. This is the simplest case, we shall see later what to do when this condition is not verified. Let $\mathbb{C}_B$ be the collocation matrix at the points in X_B with respect to the polynomial power basis $\Pi_B^{\mathbf{p}}$ of bi-degree $\mathbf{p} = (p_1, p_2)$, centred in $\text{supp}\, B$, that is,

$$\Pi_B^{\mathbf{p}} := \left\{ \left(\frac{x-a}{b-a}\right)^i \left(\frac{y-c}{d-c}\right)^j \ : \ i = 0,\dots,p_1; j = 0,\dots,p_2 \right\}.$$

Provided a so-called smoothing matrix $\mathbb{S}_B$, that we shall define at the end of this section, we then assemble the least squares with smoothing matrix $\mathbb{L}_B := \mathbb{C}_B^T\mathbb{C}_B + \mu_B\mathbb{S}_B$, with $\mu_B \geq 0$ a parameter which scales the smoothing effect yield by $\mathbb{S}_B$ in the approximation. At this stage we have two cases, depending on the displacement of the points in $\operatorname{supp} B$: $\mathbb{L}_B$ is ill-conditioned or not. In the former case, we will proceed as in the case of X_B of cardinality less than m. Therefore we postpone the treatment of this instance for the time being. Let us hence assume that $\mathbb{L}_B$ is not ill-conditioned. We solve the least squares problem

$$\mathbb{L}_B\mathbf{u} = \mathbb{C}_B^T\mathbf{z} \tag{15}$$

with $\mathbf{z} := f(X_B)$. Let $\mathcal{B}_B$ be the tensor product B-spline basis defined on the open tensor mesh set up from the knots of B. Note that $B \in \mathcal{B}_B$. If we denote by $\mathbb{R}_B$ the representation matrix of the polynomial basis $\Pi_B^{\mathbf{p}}$ in terms of the tensor product B-splines in $\mathcal{B}_B$ (see, e.g., [33, Corollary 3.6]), then we define $\lambda = \mathbf{u}^T\mathbb{R}_B$. If λ_B is the coefficient of $B \in \mathcal{B}_B$ and $\mathbf{q} := \{q_B \in \mathbb{R} \,:\, B \in \mathcal{L}\}$ is the set of QI coefficients we are looking for the LR B-spline basis $\mathcal{L}$, we assign $q_B = \lambda_B$, that is, we use the same coefficient λ_B computed as solution of the local least squares problem in terms of the tensor product B-spline basis $\mathcal{B}_B$ also for the global quasi-interpolation problem in the LR B-spline basis $\mathcal{L}$.

If X_B has less than m points, let $\widehat{\mathcal{L}} \in \mathcal{L}^A$ be the second last set of LR B-splines provided by the jump driven adaptivity cycle, just before $\mathcal{L}$. Let then $\widehat{\mathcal{L}}_B \subseteq \widehat{\mathcal{L}}$ be the set of LR B-splines in $\widehat{\mathcal{L}}$ whose supports contain the support of B. We will refer to these functions as the mothers of B (one could actually create a direct graph with the LR B-splines created during the refinement procedure of a mesh as nodes and oriented edges based on the support inclusions when the mesh has the $\mathrm{N_2S}$ property, see [3, Appendix A]). Let us define $\mathcal{R} \subseteq \Omega$ as the region given by the union of the supports of the LR B-splines in $\widehat{\mathcal{L}}_B$, i.e.,

$$\mathcal{R} := \bigcup_{\widehat{B} \in \widehat{\mathcal{L}}_B} \operatorname{supp} \widehat{B}.$$

Let $X_{\mathcal{R}} \supseteq X_B$ be the set of points in X lying in $\mathcal{R}$. If the cardinality of $X_{\mathcal{R}}$ is greater or equal to m, we add to X_B the closest $m - |X_B|$ points of $X_{\mathcal{R}} \setminus X_B$ to the centre of $\operatorname{supp} B$. Thereby, we ensure that m points are now in X_B.

However, it could happen that even by enlarging the region where we look for data to $\mathcal{R}$ we still cannot find an acceptable number of points to set the local least squares system. This happens more often as we reach higher levels of fineness in the LR mesh. In fact, as the LR B-splines support become smaller and smaller, the number of data points in $\mathcal{R}$ decreases.

As possible workaround to this occurrence, we propose the following inheritance algorithm. As first step we recursively call the SOLVE module in order to have a set of coefficients $\widehat{\mathbf{q}}_B \subseteq \widehat{\mathbf{q}}$ for the mothers of B, obtained either by solving local least squares systems or by inheritance recursively, when needed. Let then $c_{\widehat{B}}$ be the coefficient of B in the expression of $\widehat{B}$ in terms of a set of tensor product B-splines with B among them. The QI coefficient of B is assigned as the following weighted average

$$q_B := \sum_{\widehat{B} \in \widehat{\mathcal{L}}_B} c_{\widehat{B}}\widehat{q}_{\widehat{B}}.$$

Note that the sum of $\{c_{\widehat{B}}\}_{\widehat{B} \in \widehat{\mathcal{L}}_B}$ is the weight to assign to B for the partition of unity in $\mathcal{L}$, according to [3, Section 7]. By the $\mathrm{N_2S}$ property, such weight is equal to one, that is, the $\{c_{\widehat{B}}\}_{\widehat{B} \in \widehat{\mathcal{L}}_B}$ sum to one and therefore q_B is a convex combination of the coefficients of the mothers of B. As previously mentioned, the inheritance algorithm is also invoked whenever the local least squares system (15) is (nearly) singular.

It remains to define the smoothing matrix $\mathbb{S}_B$. Such matrix has been introduced in [32] to ensure existence and uniqueness of a solution to the regularized least squares system, in case of non collinear points. Beyond this, the smoothing matrix yields other benefits to the approximation, such as, lowering the risk of overfitting which results in a more reasonable behaviour of the computed surface, especially in presence of outliers in the dataset. Given a pair of indices $i \in \{0, \dots, p_1\}$ and $j \in \{0, \dots, p_2\}$ assume to order the basis $\Pi_B^{\mathbf{p}}$ by flattening the double index in some way and let $(ij) := r \in \{1, \dots, (p_1+1)(p_2+1)\}$ be the flattened index corresponding to i, j. If we denote by

$$\pi_{(ij)} := \left(\frac{x-a}{b-a}\right)^i \left(\frac{y-c}{d-c}\right)^j$$

such (ij)th basis function, the general (r,s)th entry of matrix $\mathbb{S}_B$ is given by

$$(\mathbb{S}_B)_{r,s} := \int_{\operatorname{supp} B} (\partial_{xx}\pi_{(i_1j_1)})(\partial_{xx}\pi_{(i_2j_2)}) + 2(\partial_{xy}\pi_{(i_1j_1)})(\partial_{xy}\pi_{(i_2j_2)}) + (\partial_{yy}\pi_{(i_1j_1)})(\partial_{yy}\pi_{(i_2j_2)})\mathrm{d}x\mathrm{d}y, \tag{16}$$

with $r = (i_1j_1)$ and $s = (i_2j_2)$. By linearity, we can therefore write

$$\mathbb{S}_B = \mathbb{S}_B^1 + \mathbb{S}_B^2 + \mathbb{S}_B^3$$

where $\mathbb{S}_B^k$ has (r,s)th entry given by the kth term in (16). After a simple change of variables, we rewrite this latter sum of matrices as

$$\mathbb{S}_B = |\operatorname{supp} B|\left(\frac{1}{(b-a)^4}\mathbb{S}^1 + \frac{2}{|\operatorname{supp} B|^2}\mathbb{S}^2 + \frac{1}{(d-c)^4}\mathbb{S}^3\right) \tag{17}$$

with $\mathbb{S}^k$ obtained by changing the domain of integration from $\operatorname{supp} B$ to the unit square $[0,1]^2$ and by replacing the basis $\Pi_B^{\mathbf{p}}$, used in the expression of $\mathbb{S}_B^k$, with the classical power polynomial basis $\Pi^{\mathbf{p}}$, whose (ij)th element is

$$\bar{\pi}_{(ij)} := x^i y^j.$$

The pseudo-code of the SOLVE module is presented in Algorithm 4. Note that we give the smoothing parameter as input, that is, we set μ_B equal to μ for all the LR B-splines $B \in \mathcal{L}$. Obviously, this is not the only possible choice. The smoothing effect could be dynamically changed according to the size of the supports and/or number of data points available, in order, e.g., to cope with outliers and noise.

Algorithm 4: SOLVE module, $\mathbf{q} = \texttt{solve}(\mathcal{L}, \mathcal{L}^A, X, f(X), m, \mu, \kappa_M)$

Input:

- $\mathcal{L}$: LR B-splines for which we want to compute the QI coefficients,
- $\mathcal{L}^A$: Archive of all the LR B-spline sets computed so far,
- $X, f(X)$: Dataset of scattered points and corresponding set of function values,
- m: Minimal number of points in X considered sufficient to set a least squares system,
- μ: Smoothing parameter to scale the smoothing effect,
- κ_M: Condition number threshold above which we consider a matrix ill-conditioned.

Output:

- $\mathbf{q}$: vector of the QI coefficients for the LR B-spline set $\mathcal{L}$.

for $B \in \mathcal{L}$
- Define X_B as the set of points of X lying in $\operatorname{supp} B$;
- **if** $|X_B| < m$ **then**
  - Extract from $\mathcal{L}^A$ the set of mothers of B, $\widehat{\mathcal{L}}_B$. Define $\mathcal{R} := \bigcup_{\widehat{B} \in \widehat{\mathcal{L}}_B} \operatorname{supp} \widehat{B}$; **Enlargement**
  - Define $X_{\mathcal{R}} \supseteq X_B$ as the set of points of X contained in $\mathcal{R}$;
  - **if** $|X_{\mathcal{R}}| \geq m$ **then**
    - Add to X_B the closest $m - |X_B|$ points of $X_{\mathcal{R}} \setminus X_B$ to the centre of $\operatorname{supp} B$;
  - **else**
    - Define $\widehat{\mathbf{q}}^B = \texttt{solve}(\widehat{\mathcal{L}}_B, \mathcal{L}^A, X, f(X), m, \mu)$; **Inheritance**
    - **for** $\widehat{B} \in \widehat{\mathcal{L}}_B$
      - Express $\widehat{B}$ in terms of a set of tensor product B-splines with B among them;
      - Let $c_{\widehat{B}}$ be the coefficient of B in such expression;
    - Set $q_B = \sum_{\widehat{B} \in \widehat{\mathcal{L}}_B} c_{\widehat{B}} \widehat{q}^B_{\widehat{B}}$;
    - Continue;
- Set $\mathbb{C}_B$ as the collocation matrix at the points in X_B with respect to the polynomial basis $\Pi^{\mathbf{p}}_B$;
- Compute the smoothing matrix $\mathbb{S}_B$ for B, as in Equation (17);
- Assemble the least squares with smoothing matrix $\mathbb{L}_B := \mathbb{C}_B^T \mathbb{C}_B + \mu \mathbb{S}_B$;
- **if** $\mathbb{L}_B$ *is ill-conditioned, that is,* $\kappa(\mathbb{L}_B) > \kappa_M$ **then**
  - Define $\widehat{\mathcal{L}}_B$ as the set of mothers of B, extracted from $\mathcal{L}^A$; **Inheritance**
  - Set $\widehat{\mathbf{q}}^B = \texttt{solve}(\widehat{\mathcal{L}}_B, \mathcal{L}^A, X, f(X), m, \mu)$;
  - **for** $\widehat{B} \in \widehat{\mathcal{L}}_B$
    - Express $\widehat{B}$ in terms of a set of tensor product B-splines with B among them;
    - Let $c_{\widehat{B}}$ be the coefficient of B in such expression;
  - Set $q_B = \sum_{\widehat{B} \in \widehat{\mathcal{L}}_B} c_{\widehat{B}} \widehat{q}^B_{\widehat{B}}$;
- **else**
  - Let $\mathbf{z} := f(X_B)$. Then, solve $\mathbb{L}_B \mathbf{u} = \mathbb{C}_B^T \mathbf{z}$;
  - Let $\mathbb{R}_B$ be the representation matrix of the basis $\Pi^{\mathbf{p}}_B$ in terms of tensor product B-splines $\mathcal{B}_B$;
  - Compute $\boldsymbol{\lambda} := \mathbf{u}^T \mathbb{R}_B$;
  - Extract λ_B from $\boldsymbol{\lambda}$;
  - Set $q_B = \lambda_B$;

We highlight that if $|X| > m$ there always exist a sufficiently large condition number threshold k_M and a coarse initial tensor mesh for which Algorithm 4 is able to produce an LR QI approximant, provided that the scattered data distribution is not degenerate (e.g., all aligned data-sites). In fact, under these assumptions the algorithm is able to assign a first set of QI coefficients for the B-splines on such coarse tensor mesh, without the need of enlargements or inheritances, that can be instead used for such routines when requested.

Remark 4.5. There are two main reasons why we solve the local least squares problems in the power basis of polynomials and then we write the solution in terms of a local tensor B-spline basis, instead of solving them directly in this latter basis, as done for instance in [32]. First, the assemble of the smoothing matrix is direct and very fast with the power polynomial basis, as one can see from Eq. (17). we just compute once the matrices $\mathbb{S}^1, \mathbb{S}^2, \mathbb{S}^3$ and then, for any $B \in \mathcal{L}$, scale them accordingly to the sizes of $\operatorname{supp} B$. As second reason, there would be not a straightforward way to set the tensor mesh out of the support of the LR B-spline at hand to define the local tensor B-spline basis, when enlarging the region where to seek for data points to $\mathcal{R}$.

We conclude this section by investigating the polynomial reproduction capabilities of our method. The theoretical result in [7, Proposition 4.4] ensures that, under the $\mathrm{N_2S}$ property of the mesh, for LR QI splines of bidegree $\mathbf{p}$, if the local approximation scheme used to define the coefficient associated with each LR basis function has polynomial reproduction up to bidegree $\mathbf{p}$, the same property is inherited by the final LR QI spline. More in general, under the $\mathrm{N_2S}$ property, using the same argument one shows that if the local scheme ensures polynomial reproduction of bidegree up to $\mathbf{d} \leq \mathbf{p}$ or, even less, up to a total degree $d \leq \max\{p_1, p_2\}$, then such reduced polynomial reproduction capability is shared with the final LR QI spline of bidegree $\mathbf{p}$. For this reason we are able to prove the next Proposition, in which the linear polynomial space shall be denoted by Π_1. Note that $\Pi_1 \subsetneq \Pi^{\mathbf{1}}$ the space of bilinear polynomials.

Proposition 4.6. *Assume $p_i \geq 1, i = 1, 2$, and that Algorithm 4 is able to produce a QI of bidegree $\mathbf{p}$ in an LR spline space defined on an input LR mesh with the $\mathrm{N_2S}$ property. Then, if $\mu > 0$ and the function to be approximated is a polynomial $P \in \Pi_1$, then the created QI reproduces exactly P.*

Proof. We observe that the space of linear bivariate polynomials Π_1 is included in $\mathcal{L}$, since it is $p_i \geq 1, i = 1, 2$. Thus, the restriction to Ω of any $P \in \Pi_1$ has a unique representation in $\mathcal{L}$, that is, $P_{|\Omega} = \sum_{B \in \mathcal{L}} P_B B$. Furthermore, the restriction $P_{|\operatorname{supp} B}$ can be also represented in any tensor product B-spline basis $\mathcal{B}_B$ in $\operatorname{supp} B$. In the Algorithm, we choose $\mathcal{B}_B$ such that it includes B and, as already noticed in [7, Proposition 4.4], this implies that the coefficient assigned to B for representing $P|_{\operatorname{supp} B}$ is still P_B.

Denoting with $\mathfrak{Q}(P) = \sum_{B \in \mathcal{L}} q_B B$ the QI spline produced by the algorithm when applied to P, what we need then to prove is that $\mathfrak{Q}(P) = P$, that is, $q_B = P_B, \forall B \in \mathcal{L}$. Now, considering the control flow in the algorithm, essentially two different strategies can be adopted for defining q_B, depending on the size of the local set $X_B \subset X$ and also matrix conditioning considerations. The first consists in solving a local approximation problem which relies on functional information at the subset $X_B \subset X$ (or at a suitable enlargement of X_B, i.e., $X_{\mathcal{R}}$). More precisely, referring for simplicity to X_B, in this case q_B is defined as the coefficient assigned to B in the basis $\mathcal{B}_B$ of the polynomial $g = g(P) \in \Pi_{\mathbf{p}}$, such that $g = g(P) := \operatorname{argmin}_{Q \in \Pi_{\mathbf{p}}} A_B(P, Q) + \mu I_B(Q)$, with

$$A_B(P, Q) := \sum_{(x,y) \in X_B} (Q(x, y) - P(x, y))^2 , \quad I_B(Q) := \int_{\operatorname{supp} B} \left(\partial_{xx} Q(x, y)\right)^2 + 2\left(\partial_{xy} Q(x, y)\right)^2 + \left(\partial_{yy} Q(x, y)\right)^2 \, \mathrm{dxdy} .$$

Now, being $\mu > 0$, such polynomial g is unique, for details we refer to [32, Appendix], and it necessarily coincides with P, since $A_B(P, P) = 0$ and also $I_B(P) = 0$, being $P \in \Pi_1$. Hence the coefficient q_B coincides with P_B in this case. In the second alternative, q_B is inherited from coefficients associated with mothers of B belonging to $\widehat{\mathcal{L}}$ which is a coarsening of $\mathcal{L}$ and it is extracted from the archive $\mathcal{L}^A$ of all the LR B-spline sets in input to the algorithm. In this case the proof can be obtained by using an induction argument and considering that P_B can be also defined exactly through the same inheritance formula, starting from the expression of P in the coarsest LR space in $\mathcal{L}^A$. □

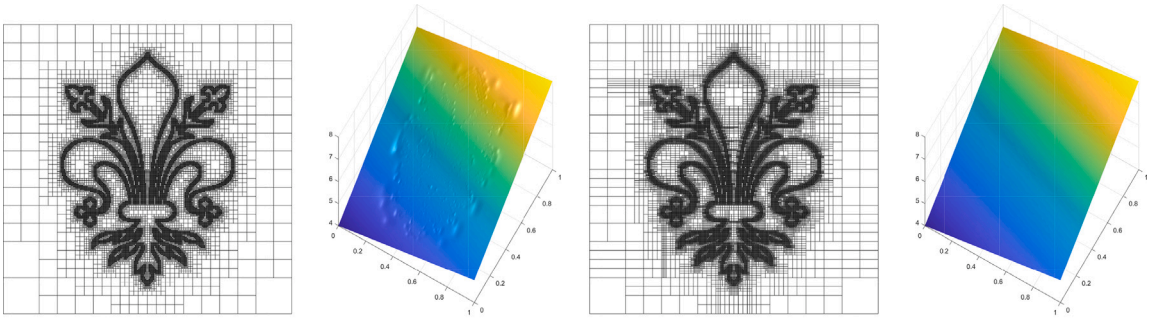
We consider such linear reproduction capability of the scheme highly demanded and indispensable in the context of shape approximation, as a faithful reconstruction of the flat areas of a surface can be accomplished only through it. This is highlighted by Fig. 6 where two different approximations of a plane are compared. They are both generated by our SOLVE module but rely on two different LR meshes. Even if both are refined in the same (arbitrary) regions, one is obtained with the $\mathrm{N_2S_2}$ refinement strategy and the other with the structured mesh approach [29] which does not guarantee the $\mathrm{N_2S}$ property. One can clearly see the artefacts (bumps) in the approximation when using the LR B-splines on the structured mesh.

5. Numerical tests

In this section we show the performance of the quasi-interpolation scheme for LR B-splines of Section 4 in some numerical tests. The location and number of iterations of the refinements on the LR meshes will be provided by the fault detections and jump estimates of Section 3.

In order to verify the quality of the jump driven approach, we shall compare the results with the purely fault guided method, in which the refinements are performed everywhere along the fault curves detected. This procedure has been proved to be reliable [18], providing approximation precision close to the error driven algorithms.

In the tests of this section we always consider scattered datasets composed of one million points, either generated by evaluation of functions at pseudo-random Halton points or acquired by LiDAR or similar technologies from real world landscapes and openly distributed on the internet. We quantify the quality of the approximation comparing the RMSE of the reconstructed surface using our jump based approach and the RMSE of the surface obtained with the purely fault guided method, both computed at all the scattered data-sites. In addition to saving a remarkable number of degrees of freedom while achieving an error of the same magnitude, as we shall see, it also turns out that the jump based method is faster than the fault based procedure both for assembling the quasi-interpolant and for evaluating it, as also shown in Table 1 for the two most challenging datasets considered, due to the smaller



(a) SM, max err = 3.24e-02, RMSE = 2.86e-03 (b) N_2S_2, max err = 4.78e-13, RMSE = 5.70e-14

Fig. 6. Linear reproduction without and with the N_2S property. Let $\mathbf{p} = (2,2)$. In figure (a) we use 9 levels of structured mesh (SM) refinement localized along the "giglio of Florence" curve to build an LR mesh in $[0,1]^2$. In figure (b) we do the same with the N_2S_2 strategy. We test the reproduction of the linear polynomial $f(x,y) = x + 3y$ using the quasi-interpolation technique described in this section with the LR B-spline sets on such meshes. In order to do so, we generate 1 million Halton points and create a scattered dataset by evaluating f at these points. As the structured mesh refinement lacks of the N_2S property, f is not reproduced, or even well approximated, and artefacts appear on the computed surface. Instead, the N_2S_2 strategy ensures the reproduction of f, despite the smoothing parameter that here has been set as $\mu = 10^{-6}$. In the captions we have reported the maximum (pointwise) error in the Halton points and the RMSE for the two approximations.

Table 1
Speedups of the isotropic jump based (JB) and anisotropic jump based (AJB) procedures compared to the purely fault driven (FD) method for the numerical tests on datasets sampled from the real world landscapes considered in Section 5 and bidegree $(3,3)$. More precisely, we measure the ratio of the times of execution of the fault driven over the jump driven approaches, that is, time(FD)/time(JB) and time(FD)/time(AJB) for the different steps of the adaptive cycle, i.e., Marking and Refining (M+R), Solving (S) and Evaluating/Estimating (E), as well as for the Total time (T). The times are taken for both the minimum and maximum $\bar{L}$ considered in the tests of the section. The code has been developed in Matlab R2023b and the tests have been run on a machine with a Intel i7-13700H CPU and 32 GB RAM.

	Barringer Crater								Fajada Butte							
	$\bar{L}=5$				$\bar{L}=7$				$\bar{L}=6$				$\bar{L}=8$			
	M+R	S	E	T	M+R	S	E	T	M+R	S	E	T	M+R	S	E	T
JB	22.9	10.7	10.0	14.02	6.82	4.86	4.73	5.64	16.9	7.98	8.57	11.54	4.15	2.60	2.73	3.31
AJB	23.2	11.4	10.5	14.63	6.79	5.00	5.00	5.76	17.55	8.24	9.45	12.19	3.12	2.95	3.13	3.09

number of basis functions to handle. The marking and refining steps are significantly faster as well, as most of the fault points are dismissed after applying $\bar{L}$ refinement steps in the jump based algorithm. For what concerns the settings, we fix an initial 2×2 tensor mesh in all the examples. We describe the results and show the approximations in the figures for bidegree $\mathbf{p} = (3,3)$. However, similar considerations hold for $\mathbf{p} = (2,2)$, as one shall see from Table 2. The maximal jump class will always be $L = 10$, while different minimal classes $\bar{L}$ will be adopted for the same experiment as reported in related tables. However, $\bar{L}$ will be the same both for ordinary and gradient faults. The smoothing parameter is fixed to $\mu = 10^{-6}$ and the minimum number of points m for setting the local least squares problem is fixed to $m = 5$. Furthermore, when considering anisotropic refinements, we start performing one directional insertions as soon as the refinement iteration reaches the minimal jump class, $L^A = \bar{L}$. Finally, the least squares matrix $\mathbb{L}$ shall be considered ill-conditioned if its condition number is above κ_M =1e05. However, this occurrence has never happened in the reported tests. Of course, these are not the only possible, and perhaps not optimal, settings: for instance, one may choose different minimal and maximal jump classes for ordinary and gradient faults, as well as an independent class from which to refine anisotropically. Nevertheless, the results show the potential of this kind of technique.

5.1. *Approximation of synthetic data from functional sampling at scattered points*

We test the algorithm for surface reconstruction from scattered data provided by the sampling of two functions, namely

$$f_1(x,y) := \begin{cases} \left| x - \frac{2}{5} - \frac{1}{10}\sin(2\pi y) \right| & \text{if } x \le \frac{7}{10} + \frac{1}{5}\sin(2\pi y), \\ \left| x - \frac{2}{5} - \frac{1}{10}\sin(2\pi y) - \frac{1}{5} \right| & \text{otherwise,} \end{cases} \tag{18}$$

sampled in $[0,1]^2$, and

$$f_2(x,y) := \mathrm{Im}(\arcsin(z) + \arcsin(i\bar{z})) \tag{19}$$

with $z := x + iy$, sampled in $[-4,4]^2$ (f_2 is a so-called *branch cut*, details can be found in, e.g., [34]).

f_1 restricted to $[0,1]^2$ has range $[0, 0.5]$ and it is characterized by two faults along sinusoidal curves: one ordinary fault, with a constant jump of $\frac{1}{5}$, and one varying gradient fault. The results of the jump driven quasi-interpolations are illustrated in Fig. 7 and

Table 2
Numerical tests for bidegree $\mathbf{p} = (2, 2)$. In the last row we report the outcomes for the pure fault driven (FD) method for comparison.

Function (18)					Function (19)			Barringer Crater					Fajada Butte				
	Isotropic		Anisotropic			Anisotropic			Isotropic		Anisotropic			Isotropic		Anisotropic	
$\bar{L}$	ndofs	RMSE	ndofs	RMSE	$\bar{L}$	ndofs	RMSE	$\bar{L}$	ndofs	RMSE	ndofs	RMSE	$\bar{L}$	ndofs	RMSE	ndofs	RMSE
–	–	–	–	–	1	885	8.19e−02	5	6708	6.51e−04	5519	6.54e−04	6	12 214	2.84e−03	10 580	2.88e−03
–	–	–	–	–	4	1127	8.15e−02	6	11 547	4.02e−04	10 119	4.31e−04	7	20 339	2.57e−03	17 659	2.63e−03
9	43 906	3.17e−03	42 089	3.19e−03	8	5258	6.65e−02	7	22 486	2.92e−04	19 810	3.07e−04	8	38 171	2.43e−03	32 995	2.47e−03
FD	45 240	3.17e−03	–	–	FD	22 068	5.44e−02	FD	140 689	2.41e−04	–	–	FD	105 336	2.34e−03	–	–

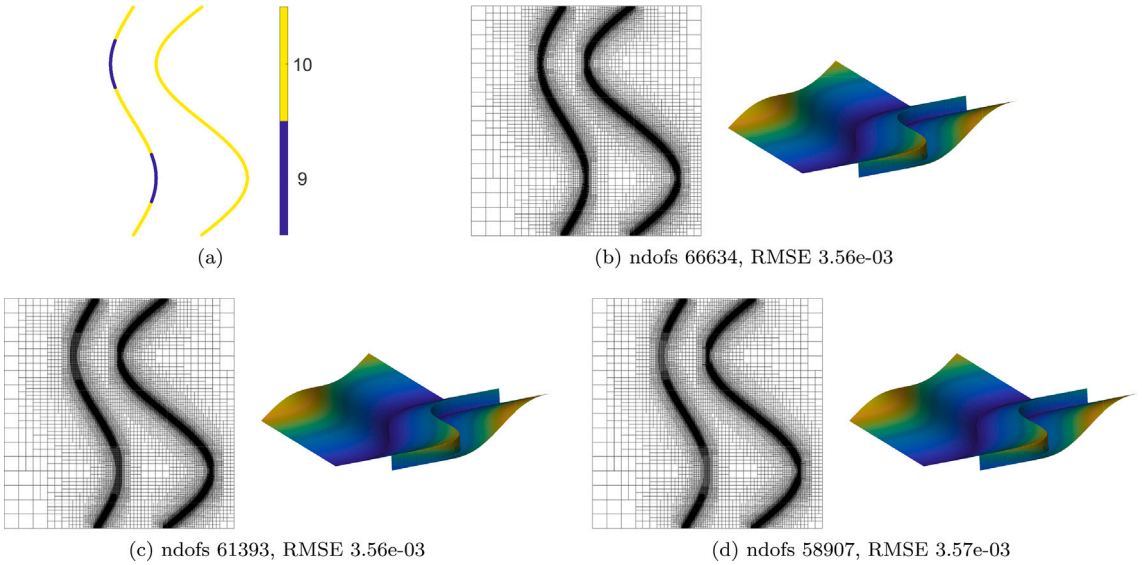


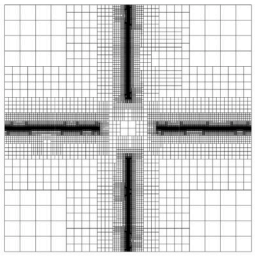
Fig. 7. QI approximation of scattered points from function (18). Such function has two sinusoidal fault curves. The one on the right is a constant ordinary fault and the one on the left is a varying gradient fault. The points detected for the ordinary fault have all been classified at the maximum level of refinement $\ell = 10$ by the jump classification algorithm. The points detected for the gradient fault have instead been split into two classes corresponding to levels $\ell = 9$ and $\ell = 10$, see figure (a). In figures (b) we see mesh and approximation provided by the purely fault based refinement. The number of dofs and RMSE are reported in the caption of the figure. In figures (c) and (d) we show the results of the isotropic and anisotropic jump driven refinements, respectively.

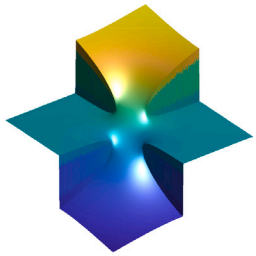
compared with the purely fault guided strategy. Here, the advantages of the jump driven strategies are mitigated by the fact that the jump range is relatively small compared to its overall magnitude. This leads to a small variety in the jump class distribution of the detected fault points. Moreover, the axis-aligned pieces of the fault curve are quite small and, for the gradient fault curve, related to the lower jump class, which reduces the impact of the anisotropic insertions. Nevertheless, as one can see from the figure, essentially the same RMSE of the fault driven algorithm is reached both with the isotropic and anisotropic refinement approaches, despite saving approximately 5200 and 7700 degrees of freedom, respectively, i.e., a reduction of about $1/12$ and $1/9$ of the total.

f_2 restricted to $[-4, 4]^2$ has (approximate) range $[-4.85, 4.85]$ and it shows four symmetric axis-aligned ordinary faults of varying jumps, with a decreasing intensity towards the centre, where it is actually continuous. The results of the jump driven quasi-interpolation with anisotropic refinements are reported in Fig. 8 and compared with the fault guided approach. This function has been chosen in order to highlight and stress the possible terrific reduction in the degrees of freedom, while retaining the same magnitude of RMSE, by allowing anisotropic insertions when (most of) the detected fault points can be labelled as axis-aligned, possibly after a suitable rotation of the point cloud, in combination with jump driven marking. Indeed, in this test we can reach essentially the same RMSE of the purely fault guided refinement with approximately $1/4$ (for $\bar{L} = 8$), $1/18$ (for $\bar{L} = 4$) and $1/20$ (for $\bar{L} = 1$), respectively, of the degrees of freedom.

This latter function has been adopted also for carrying a first, preliminary, test in the presence of data affected by noise. Namely, we have perturbed the dataset sampled from function (19) with a Gaussian distributed noise of magnitude 0.25, that is, about the 5.70% of the maximum jump value of the unperturbed function. Fig. 9 summarizes the results. As the jump becomes lower towards the centre of the surface, the detection struggles to distinguish it from the noise. Furthermore the noise is handed down to the jump estimates to some extent. This latter drawback has however little-to-none effect on the jump guided refinement ultimately. For what concerns the former, the loss of the part of fault where the jump is lower can be compensated by considering a finer initial tensor mesh. This is advisable also for mitigating the effect of the noise. Such finer mesh shall be reasonable for having a fair approximation of the fault where the jump is low, as the local refinements there would stop pretty soon anyway as lower jump values correspond to lower jump classes.

(a)

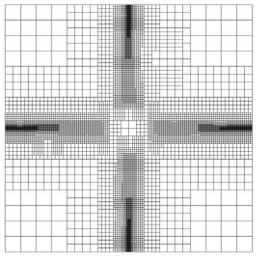


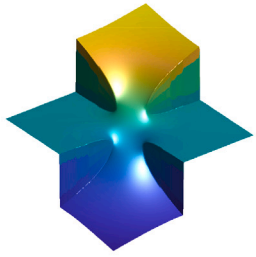


(b) ndofs 30935, RMSE 6.06e-02

$\bar{L}$	ndofs	RMSE
1	1558	8.37e-02
4	1696	8.37e-02
8	7310	7.28e-02

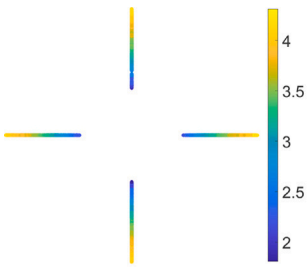
(c)





(d) $\bar{L} = 8$, ndofs 7310, RMSE 7.28e-02

Fig. 8. QI approximation of scattered data from function (19). The surface is characterized by four symmetric axis-aligned ordinary faults of decreasing intensity towards the centre. The points detected as fault points can be distributed on jump classes $\ell \in \{\bar{L}, \dots, L\}$, from any $\bar{L} \in \{1, \dots, 10\}$, using the jump classification algorithm. In figure (a), we see the classification for $\bar{L} = 1$. In figure (b) we show mesh and approximation when using the purely fault driven refinement, as it will be used for comparison with the jump based approach. In the caption of the figure we have recorded the number of degrees of freedom and RMSE. In table (c) we have instead the results for the anisotropic jump driven refinement for different jump classifications, obtained by picking different $\bar{L}$. In figure (d) we present mesh and approximation for the case $\bar{L} = 8$.



(a)

(b)

	$\mathbf{p} = (2,2)$		$\mathbf{p} = (3,3)$	
$\bar{L}$	ndofs	RMSE	ndofs	RMSE
4	912	9.60e-02	1463	1.05e-01
6	1847	8.71e-02	2592	9.01e-02
8	5289	8.06e-02	7377	8.39e-02

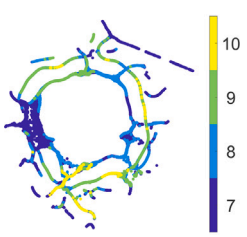
(c)

Fig. 9. A preliminary test for noisy datasets. We have perturbed the values of the dataset created for function (19) with a Gaussian noise of magnitude 0.25, i.e., about 5.70% of the maximal jump value. In figure (a) the jump estimates. In figure (b) the corresponding jump classification. In (c) the table providing the results for degrees $\mathbf{p} = (2,2)$ and $\mathbf{p} = (3,3)$ when choosing $\bar{L} = 4, 6, 8$ respectively.

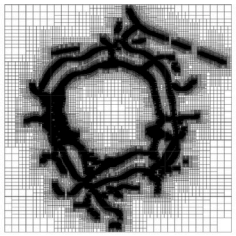
5.2. Approximation of landscape datasets

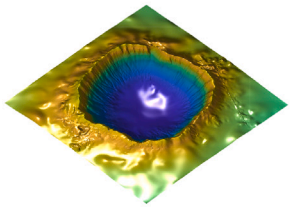
In this section we test our jump driven quasi-interpolation on some datasets acquired by the scanning of real world landscapes. The first point cloud we consider has been captured at the Barringer Crater in Arizona [35]. The sampling has been rescaled to be in the range $[0.5, 0.6]$. Fig. 10 shows the results both for isotropic and anisotropic refinements, compared to the fault driven approach in figure (b). In figure (a) we show the jump classified when $\bar{L} = 7$. The RMSE remains of the same magnitude for any choice of $\bar{L} \geq 5$ and the amount of details is also qualitatively comparable, as one can see from figures (d) and (e) for $\bar{L} = 7$. On the other hand, we achieve a remarkable reduction in the degrees of freedom as illustrated by the table in figure (c).

As second test, we present the approximation of a dataset of the Fajada Butte in New Mexico [36]. The sampling has been rescaled in the range $[0, 0.45]$. The results and surfaces can be seen in Fig. 11. As before, in figure (a) we have the jump classification for $\bar{L} = 8$ and in figure (b) we have mesh and surface of the fault driven strategy. In the caption we report the corresponding degrees of freedom and RMSE. In figure (c) we have a table schematizing the outcomes of the jump driven refinement, without and with anisotropic insertions, for different minimal jump classes. Also here, the reduction in the degrees of freedom is quite significant, despite producing approximation of the same quality of the fault guided approach, with respect to the RMSE. In figures (d) and (e) we show mesh and surfaces for $\bar{L} = 8$ in the isotropic and anisotropic, respectively, jump driven algorithms for a qualitative comparison with figure (b).



(a)

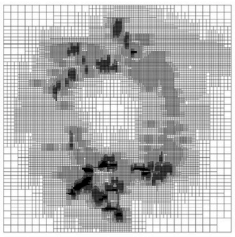


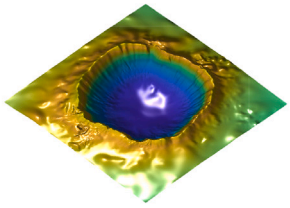


(b) ndofs 183802, RMSE 2.22e-04

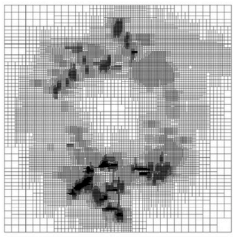
	isotropic		anisotropic	
$\bar{L}$	ndofs	RMSE	ndofs	RMSE
5	8174	7.70e-04	6725	7.99e-04
6	15425	4.39e-04	13396	4.55e-04
7	29910	2.93e-04	26897	3.06e-04

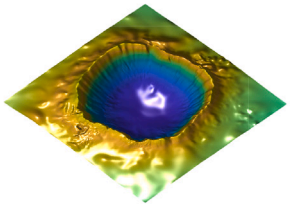
(c)





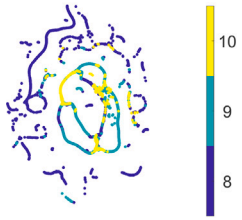
(d) $\bar{L} = 7$, isotropic, ndofs 29910, RMSE 2.93e-04



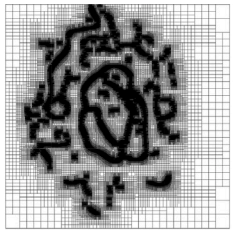


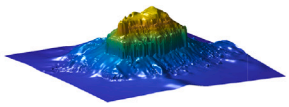
(e) $\bar{L} = 7$, anisotropic, ndofs 26897, RMSE 3.06e-04

Fig. 10. QI approximation of a one million scattered point cloud of the Barringer Crater in Arizona, USA [35]. In figure (a) the jump classification for $\bar{L} = 7$. In figure (b) we have mesh and approximation using the purely fault driven approach to localize the refinement, when setting the maximal level to $L = 10$. The number of degrees of freedom and RMSE are listed in the caption of the figure. In table (c) we report the results for the isotropic and anisotropic jump driven refinements, for different choices of $\bar{L}$ in the jump classification. In figures (d)–(e) we present meshes and approximations of the two approaches in the instance of $\bar{L} = 7$.



(a)

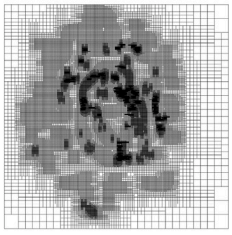


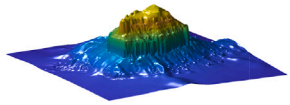


(b) ndofs 148700, RMSE 2.31e-03

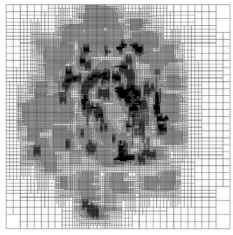
	isotropic		anisotropic	
$\bar{L}$	ndofs	RMSE	ndofs	RMSE
6	16820	2.87e-03	14536	2.93e-03
7	27734	2.59e-03	24237	2.63e-03
8	51607	2.42e-03	45324	2.47e-03

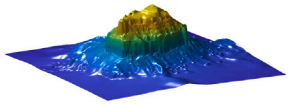
(c)





(d) $\bar{L} = 8$, isotropic, ndofs 51607, RMSE 2.42e-03





(e) $\bar{L} = 8$, anisotropic, ndofs 45324, RMSE 2.47e-03

Fig. 11. QI approximation of a one million scattered point cloud of the Fajada Butte in New Mexico, USA [36]. In figure (a) the jump classification for $\bar{L} = 8$. In figure (b) we have mesh and approximation using the purely fault driven approach to localize the refinement, when setting the maximal level to $L = 10$. The number of degrees of freedom and RMSE are listed in the caption of the figure. In table (c) we report the results for the isotropic and anisotropic jump driven refinements, for different choices of $\bar{L}$ in the jump classification. In figures (d)–(e) we present meshes and approximations of the two approaches in the instance of $\bar{L} = 8$.

6. Closure

In this work we have presented a procedure to estimate the jump of a function and its gradient across ordinary and gradient fault curves, respectively, when only values of such function at a scattered dataset are available. In the context of surface approximation, the larger the jump is, the more degrees of freedom are required to capture such behaviour. We proposed a new adaptive loop for surface reconstruction driven by the computed jump values rather than error estimates. Consequently, no intermediate approximations and evaluations are required. We have adopted LR spline spaces as adaptive spaces in order to gain the possibility for anisotropic refinements. This further feature allows to save more degrees of freedom when the fault curves are axis-aligned. As

future developments, we consider of interest the problem of estimating the jump in case of fault intersections, both from a theoretical and a practical point of view. Furthermore, we shall extend the procedures to handle more general point clouds, such as those that cannot be parametrized as graph of a function, and data affected by noise and/or outliers, extending the preliminary study carried at the end of Section 5.1.

CRediT authorship contribution statement

Cesare Bracco: Writing – review & editing, Writing – original draft, Investigation, Formal analysis, Conceptualization. **Carlotta Giannelli:** Writing – review & editing, Writing – original draft, Investigation, Supervision, Resources, Funding acquisition, Conceptualization. **Francesco Patrizi:** Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Alessandra Sestini:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization.

Acknowledgements

Supported by 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, Italy (CUP B83C22002830001). The authors are members of INdAM-GNCS; CB and CG were partially supported by an INdAM-GNCS Project, Italy (CUP E53C22001930001). CB, CG, and AS also acknowledge the partial support of the Italian Ministry of University and Research (MUR), Italy through the PRIN projects COSMIC (No. 2022A79M75) and NOTES, Italy (No. P2022NC97R), funded by the European Union - Next Generation EU.

References

[1] X. Li, J. Zheng, T.W. Sederberg, T.J.R. Hughes, M.A. Scott, On linear independence of T-spline blending functions, Comput. Aided Geom. Design. 29 (2012) 63–76.
[2] C. Giannelli, B. Jüttler, H. Speleers, THB-splines: the truncated basis for hierarchical splines, Comput. Aided Geom. Design. 29 (2012) 485–498.
[3] T. Dokken, T. Lyche, K.F. Pettersen, Polynomial splines over locally refined box-partitions, Comput. Aided Geom. Design. 30 (2013) 331–356.
[4] A. Bressan, Some properties of LR-splines, Comput. Aided Geom. Design. 30 (2013) 778–794.
[5] F. Patrizi, T. Dokken, Linear dependence of bivariate minimal support and locally refined B-splines over LR-meshes, Comput. Aided Geom. Design 77 (2020) 22, 101803.
[6] A. Bressan, B. Jüttler, A hierarchical construction of LR meshes in 2D, Comput. Aided Geom. Design. 37 (2015) 9–24.
[7] F. Patrizi, C. Manni, F. Pelosi, H. Speleers, Adaptive refinement with locally linearly independent LR B-splines: theory and applications, Comput. Methods Appl. Mech. Engrg. 369 (2020) 20, 113230.
[8] F. Patrizi, Effective grading refinement for locally linearly independent LR B-splines, BIT 62 (2022) 1745–1764.
[9] A.V. Vuong, C. Giannelli, B. Jüttler, B. Simeon, A hierarchical approach to adaptive local refinement in isogeometric analysis, Comput. Methods Appl. Mech. Engrg. 200 (2011) 3554–3567.
[10] N. Engleitner, B. Jüttler, Patchwork B-spline refinement, Comput.-Aided Des. 90 (2017) 168–179.
[11] N. Engleitner, B. Jüttler, Lofting with patchwork B-splines, in: Advanced Methods for Geometric Modeling and Numerical Simulation, in: Springer INdAM Ser., vol. 35, Springer, Cham, 2019, pp. 77–98.
[12] L. Romani, M. Rossini, D. Schenone, Edge detection methods based on RBF interpolation, J. Comput. Appl. Math. 349 (2019) 532–547.
[13] T. Galkowski, A. Krzyzak, Z. Filutowicz, A new approach to detection of changes in multidimensional patterns, J. Artif. Intell. Soft. Comput. Res. 10 (2020) 125–136.
[14] M. Bozzini, M. Rossini, The detection and recovery of discontinuity curves from scattered data, J. Comput. Appl. Math. 240 (2013) 148–162.
[15] M. Bozzini, L. Lenarduzzi, M. Rossini, Non-regular surface approximation, in: Mathematical Methods for Curves and Surfaces, in: Lecture Notes in Comput. Sci., vol. 8177, Springer, Heidelberg, 2014, pp. 68–87.
[16] C. Bracco, O. Davydov, C. Giannelli, A. Sestini, An application of numerical differentiation formulas to discontinuity curve detection from irregularly sampled data, Rend. Semin. Mat. Univ. Politec. Torino 76 (2018) 69–76.
[17] C. Bracco, O. Davydov, C. Giannelli, A. Sestini, Fault and gradient fault detection and reconstruction from scattered data, Comput. Aided Geom. Design 75 (2019) 20, 101786.
[18] C. Bracco, F. Calabrò, C. Giannelli, Discontinuity detection by null rules for adaptive surface reconstruction, J. Sci. Comput. 97 (2023) 37.
[19] M. Buhmann, J. Jäger, Quasi-interpolation, in: Cambridge Monographs on Applied and Computational Mathematics, vol. 37, Cambridge University Press, Cambridge, 2022.
[20] P. Sablonnière, D. Sbibih, M. Tahrichi, Error estimate and extrapolation of a quadrature formula derived from a quartic spline quasi-interpolant, BIT 50 (2010) 843–862.
[21] F. Mazzia, A. Sestini, Quadrature formulas descending from BS Hermite spline quasi-interpolation, J. Comput. Appl. Math. 236 (2012) 4105–4118.
[22] B. Degli Esposti, A. Falini, T. Kanduč, M.L. Sampoli, A. Sestini, IgA-BEM for 3D Helmholtz problems using conforming and non-conforming multi-patch discretizations and B-spline tailored numerical integration, Comput. Math. Appl. 147 (2023) 164–184.
[23] V. Skytt, O. Barrowclough, T. Dokken, Locally refined spline surfaces for representation of terrain data, Computers & Graphics 49 (2015) 58–68.
[24] V. Skytt, Q. Harpham, T. Dokken, H.E. Dahl, Deconfliction and surface generation from bathymetry data using LR B-splines, in: Mathematical Methods for Curves and Surfaces: 9th International Conference, MMCS 2016, Tønsberg, Norway, June 23–28, Springer, 2016, pp. 270–295, Revised Selected Papers 9.
[25] V. Skytt, T. Dokken, Scattered data approximation by LR B-spline surfaces: A study on refinement strategies for efficient approximation, in: Geometric Challenges in Isogeometric Analysis, in: Springer INdAM Ser., vol. 49, Springer, Cham, 2022, pp. 217–258.
[26] V. Skytt, G. Kermarrec, T. Dokken, LR B-splines to approximate bathymetry datasets: An improved statistical criterion to judge the goodness of fit, Int. J. Appl. Earth Obs. Geoinf. 112 (2022) 102894.
[27] G. Kermarrec, V. Skytt, T. Dokken, Optimal surface fitting of point clouds using local refinement: Application to GIS data, in: Springerbriefs in Earth System Sciences, first ed., Springer, Cham, 2023.
[28] H. Speleers, C. Manni, Effortless quasi-interpolation in hierarchical spaces, Numer. Math. 132 (2016) 155–184.
[29] K.A. Johannessen, T. Kvamsdal, T. Dokken, Isogeometric analysis using LR B-splines, Comput. Methods Appl. Mech. Engrg. 269 (2014) 471–514.
[30] O. Davydov, R. Schaback, Minimal numerical differentiation formulas, Numer. Math. 140 (2018) 555–592.
[31] I.K. Lee, Curve reconstruction from unorganized points, Comput. Aided Geom. Design. 17 (2000) 161–177.

[32] C. Bracco, C. Giannelli, D. Großmann, S. Imperatore, D. Mokriš, A. Sestini, THB-spline approximations for turbine blade design with local B-spline approximations, in: Mathematical and Computational Methods for Modelling, Approximation and Simulation, in: SEMA SIMAI Springer Series, vol. 29, Springer, Charm, 2022, pp. 63–82.
[33] T. Lyche, K. Mørken, Spline methods draft, 2018.
[34] M.J. Ablowitz, A.S. Fokas, Complex variables: introduction and applications, in: Cambridge Texts in Applied Mathematics, second ed., Cambridge University Press, Cambridge, 2003.
[35] M. Palucis, Meteor Crater, AZ, National Center for Airborne Laser Mapping (NCALM), 2011, Distributed by OpenTopography.
[36] W. Dorshow, 3D Landscape Reconstruction and Land Use Modeling, Chaco Canyon, NM 2016, National Center for Airborne Laser Mapping (NCALM), 2019, Distributed by OpenTopography.