Numerical Algorithms
https://doi.org/10.1007/s11075-023-01618-6

ORIGINAL PAPER

Detecting and approximating decision boundaries in low-dimensional spaces

Matthias Grajewski[1,2] · Andreas Kleefeld[1,3]

Received: 22 February 2023 / Accepted: 3 July 2023

Abstract
A method for detecting and approximating fault lines or surfaces, respectively, or decision curves in two and three dimensions with guaranteed accuracy is presented. Reformulated as a classification problem, our method starts from a set of scattered points along with the corresponding classification algorithm to construct a representation of a decision curve by points with prescribed maximal distance to the true decision curve. Hereby, our algorithm ensures that the representing point set covers the decision curve in its entire extent and features local refinement based on the geometric properties of the decision curve. We demonstrate applications of our method to problems related to the detection of faults, to multi-criteria decision aid and, in combination with Kirsch's factorization method, to solving an inverse acoustic scattering problem. In all applications we considered in this work, our method requires significantly less pointwise classifications than previously employed algorithms.

Keywords Fault detection · Fault approximation · Inverse scattering problem · MCDA

Mathematics Subject Classification (2010) 65-D10 · 65-D15 · 65-D17

✉ Matthias Grajewski
grajewski@fh-aachen.de

Andreas Kleefeld
a.kleefeld@fz-juelich.de

1 Faculty of Medical Engineering and Technomathematics, FH Aachen University of Applied Sciences, Heinrich-Mußmann-Str. 1, 52428 Jülich, Germany

2 Institute for Data-Driven Technologies, FH Aachen University of Applied Sciences, Heinrich-Mußmann-Str., 52428 Jülich, Germany

3 Forschungszentrum Jülich GmbH, Jülich Supercomputing Centre, Wilhelm-Johnen-Str., 52425 Jülich, Germany

1 Introduction

Let us consider a piecewise constant function $f : \mathbb{R}^m \supset \Omega \to \{1, 2, \ldots, n\}$ with Ω being compact, simply connected and equipped with a piecewise smooth boundary, $m \in \{2, 3\}$. Such f subdivides Ω into mutually disjoint subsets $\Omega_i := f^{-1}(i)$ with $\Omega = \cup_{i=1}^{n} \overline{\Omega_i}$ and $\Omega_i \cap \Omega_j = \emptyset$, if $i \neq j$. We assume that each Ω_i features a piecewise smooth boundary. We are interested in approximating $\Gamma_{i,j} := \overline{\Omega_i} \cap \overline{\Omega_j}$ relying on as few evaluations of f as possible and present in this work an algorithm for this task. We choose this quantity as a measure of efficiency because evaluating f can be arbitrarily costly in applications and dominates the runtime in such a case. Our problem can be immediately understood as a classification problem, such that we identify Ω_i with a class i and interpret the curves or surfaces of discontinuity $\Gamma_{i,j}$ as decision curves or surfaces.

One field of application is economics and operations research. Multi-criteria decision aid (MCDA) methods can help a decision maker to choose the best one from a finite number of alternatives based on different, even conflicting, criteria. MCDA methods assume that a decision depends on quantifiable parameters $(x_1, \ldots, x_m)$ ("input factors") and is drawn deterministically. For an overview over various MCDA approaches, applications and case studies, we refer to [1, 2] among many others and the references cited therein. In this context, Ω_i is the set of all input factors that lead to the decision for the i-th alternative in the MCDA method. Analysing the decision process with respect to the input factors means consistently describing all Ω_i based upon evaluating the MCDA method for arbitrary combinations of input factors. This can be achieved by approximating all $\Gamma_{i,j}$. Reconstructing an obstacle in three dimensions is a field of application in acoustic scattering theory. More precisely, one wants to determine the support of an inhomogeneous object (its boundary to be exact) from measured far-field data which typically is a desired task in non-destructive testing. The far-field data are obtained for different incident waves and measured points on the unit sphere. Several reconstruction algorithms to find the boundary of the unknown inhomogeneity are available such as iterative methods [3], decomposition methods [4] and sampling/probe methods (see [5] for a detailed overview). The latter ones can be further categorised into the linear sampling method, the generalised linear sampling method, the factorization method, the probe method and variants of it (refer to [5–9], respectively). However, here, we will focus on the classical factorization method for the acoustic transmission problem (refer also to [10]), with which one can decide if a given point is located inside or outside the obstacle. Therefore, the factorization method transfers the reconstruction of an obstacle to a classification problem and thus into a field of application of our method. Note that the far-field data within [10] has also been used in [11] and [12].

Finding and approximating the sets $\Gamma_{i,j}$, sometimes called fault lines, is important in the exploration of natural resources. The presence and the location of $\Gamma_{i,j}$ can provide useful insights for exploring and later on exploiting of oil reservoirs [13] and play a significant role in some geophysical applications [14]. The underlying mathematical problem is closely related to ours, albeit not the same, as usually, the function f considered does not provide integer values as in our case. Therefore, an additional algorithm for detecting fault lines is required then, and the classification of a single point may be not trivial anymore as it is in our case. Moreover, algorithms for fault detection may need to deal with noisy data (e.g. [15]), whereas we consider certain data only.

Many algorithms for detecting and approximating the sets $\Gamma_{i,j}$ have been proposed, like [13–18] among many others, which all feature strengths and weaknesses. Many of these methods are discussed in the wider picture of scattered data approximation [19] or adaptive scattered data approximation based on partion of unity methods [20]. Mirzai and Soodbakhsh [21] propose a method for detecting and approximating possibly intersecting fault lines using a partition of unity method. The fault lines they consider are not necessarily boundaries of subdomains Ω_i, such that there are differences to the problem we address in this paper. However, the vast majority of these approaches restrict to the 2D case, whereas we present a method for 2D and 3D. The algorithm proposed in [17] and the work cited therein were the starting point for our research.

Classification is one of the standard problems in machine learning. There are a lot of powerful and versatile algorithms available which could readily applied to our problem; we refer to [22] for an overview. These methods are however designed for uncertain data in high-dimensional spaces, whereas we consider secure data in low-dimensional spaces, a completely different use case.

Our method approximately describes the $\Gamma_{i,j}$ by providing a set of points with a guaranteed maximal normal distance to $\Gamma_{i,j}$. These points are intended for constructing a polygon (2D) or a surface triangulation in 3D. While there are more sophisticated and elegant ways of describing these sets, it allows us to (approximately) replace an actual classification by a simple and fast point-in-polygon test. A Python implementation and a Matlab implementation of our algorithm used for producing the results presented in this article are available at https://github.com/mgrajewski/faultapprox-matlab and https://github.com/mgrajewski/faultapprox-python. These codes include the examples and tests shown here.

This article is organised as follows: We describe our algorithm for 2D and 3D in Sect. 2 and elaborate on the direct and inverse acoustic scattering problem, one of our applications, in Sect. 3. We present results and applications of our algorithm to the detection of faults, to decision modelling and to inverse scattering in Sect. 4 and finally conclude in Sect. 5. We provide pseudo-codes of selected parts of our algorithm in the Appendix.

2 The algorithm

2.1 Detection and approximation of $\Gamma_{i,j}$ in 2D

In this section, we present our algorithm for approximating $\Gamma_{i,j} \neq \emptyset$ for $m = 2$ by sufficiently many well-distributed and ordered points sufficiently close to $\Gamma_{i,j}$. This

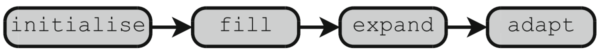


Fig. 1 General flow chart

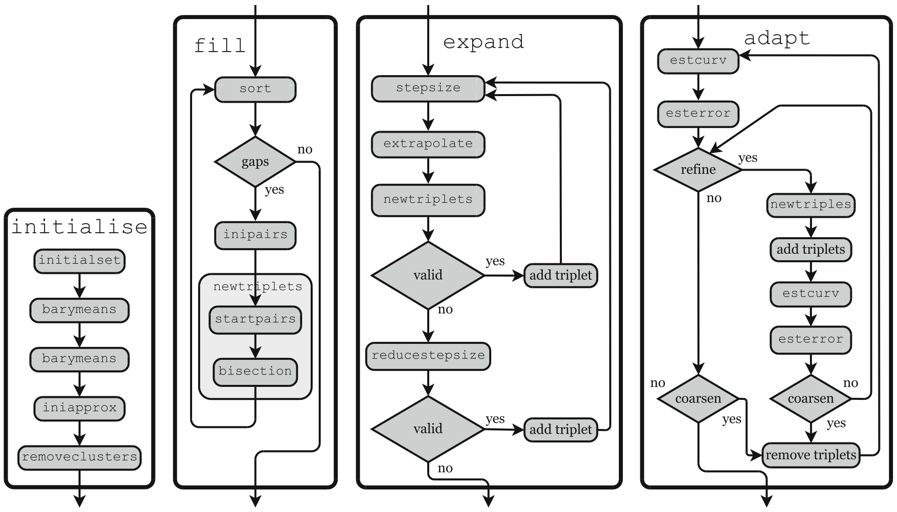


Fig. 2 Flow charts of the Algorithms `initialise`, `fill`, `expand` and `adapt`. Note that building block `barymeans` inside `initialise` is employed twice, albeit to different sets

implicitly constitutes a polygonal description of $\Gamma_{i,j}$. For convenience, we provide flow charts (Figs. 1 and 2) and describe selected building blocks in detail. In the flow charts, actual building blocks are typed in monospace lettering.

Algorithm 2.1 (`initialise`) This algorithm aims at providing initial approximations to any $\Gamma_{i,j}$; we refer to Fig. 1 (right) for an overview. From now on, we assume $\Gamma_{i,j} \neq \emptyset$. In building block `initialset`, we sample f on Ω rather coarsely and obtain an initial point set X along with the corresponding classification information. Building block `barymeans` creates additional sampling points in the vicinity of any $\Gamma_{i,j}$. We employ a k_{near}-nearest neighbour approach taken from Allasia et al. [17]: For any $x \in X$, let $N(x)$ be the set of the k_{near}- nearest neighbours of x in X. If $N(x) \cap \Omega_i \neq \emptyset$ and $N(x) \cap \Omega_j \neq \emptyset$, we consider x close to $\Gamma_{i,j}$. Let $N_\ell(x) = N(x) \cap \Omega_\ell$ for some ℓ. Hence, $N(x) = \bigcup_{i=1}^{r} N_{c_i}(x)$ for certain indices $c_1, \ldots, c_r$, $N_{c_i}(x) \neq \emptyset$. We compute the barycentres b_{c_i} of all $N_{c_i}(x)$, $1 \leq i \leq r$, and then their arithmetic means $y_{c_i,c_j} = 0.5(b_{c_i} + b_{c_j})$, $1 \leq i < j \leq r$. Let $M(x)$ be the set of all y_{c_i,c_j} generated from $N(x)$. If $N(x) \subset \Omega_\ell$ for some ℓ, we set $M(x) = \emptyset$. We end up with $\mathcal{M} = \bigcup_{x \in X} M(x)$. By definition of $\mathcal{M}$, there are no duplicate points; however, a practical implementation requires removing duplicates. After classifying the points in $\mathcal{M}$, we repeat `barymeans` on $\mathcal{M}$ obtaining sets $M^2(x)$ for any $x \in \mathcal{M}$, then $\mathcal{M}^2 = \bigcup_{x \in \mathcal{M}} M^2(x)$ and ultimately a further enriched set of sampling points $\overline{X} = X \cup \mathcal{M} \cup \mathcal{M}^2$. Building block `iniapprox` (for a pseudo-code, we refer to the Appendix, Algorithm 1 computes initial approximations for all $\Gamma_{i,j}$. For any $x \in (\mathcal{M} \cup \mathcal{M}^2) \cap \Omega_i$, we search the nearest point $x' \in \overline{X} \cap \Omega_j$, $j > i$. We use x and x' as starting points for a bisection algorithm on the line $\overline{x'x}$. If the bisection algorithm is successful, we end up with a point pair $x^{(i)} \in \Omega_i$ and $x^{(j)} \in \Omega_j$ with $\|x^{(i)} - x^{(j)}\| \leq 2\varepsilon_b$, where ε_b is a user-prescribed threshold. Then, the distance of

$x^{(i,j)} = 0.5(x^{(i)} + x^{(j)})$ to $\Gamma_{i,j}$ is at most ε_b. We subsume the bisection process up to ε_b and computing $x^{(i,j)}$ from its results in building block `bisection`. From now on, we consider point triplets for approximating $\Gamma_{i,j}$ only; for any such triplet x, the superscript (i) denotes the point in Ω_i, the superscript (j) its counterpart in Ω_j and the superscript (i, j) the arithmetic mean of the two points. We end up with a set of triplets $\widetilde{S}_{i,j}$. We moreover set $\widetilde{S}^{(i)}_{i,j} = \{x^{(i)} \mid x \in \widetilde{S}_{i,j}\}$ and $\widetilde{S}^{(j)}_{i,j} = \{x^{(j)} \mid x \in \widetilde{S}_{i,j}\}$. It may occur that some triplets in $\widetilde{S}_{i,j}$ are tightly clustered. We thin such clusters as they add to complexity but not to accuracy by removing appropriate triplets (Fig. 3). After cluster removal, `initialise` provides sets of triplets $S_{i,j}$.

Remark 2.1 Building block `initialise` detects $\Gamma_{i,j} \neq \emptyset$ by $\widetilde{S}_{i,j} \neq \emptyset$. However, depending on X, it may happen that $\widetilde{S}_{i,j} = \emptyset$ even if $\Gamma_{i,j} \neq \emptyset$. Reliably detecting all non-empty $\Gamma_{i,j}$ depends on a sufficiently large X and thus ultimately on the user.

The triplets in $S_{i,j}$ usually provide an incomplete approximation of $\Gamma_{i,j}$ only, feature gaps and lack ordering (e.g. Fig. 10, right).

Remark 2.2 The bisection-based approach in building block `iniapprox` implicitly assumes that $\overline{x'x}$ intersects $\Gamma_{i,j}$ and therefore may fail if this does not hold (Fig. 4). However, such failure modes occur only rarely in practical computations, and we implemented fallbacks in that case.

Algorithm `fill` (for an overview, we refer to the flow chart in Fig. 2) provides triplets with a maximal user-prescribed distance ε_b to $\Gamma_{i,j}$ and maximal mutual distance ε_{gap} based upon $S_{i,j}$, i.e. the result of Algorithm 2.1 (`initialise`). As we assume that $\varepsilon_b \ll \varepsilon_{\text{gap}}$, we define the distance of two triplets $x, y \in S_{i,j}$ as $\|x^{(i)} - y^{(i)}\|$ and the distance of x to $\Gamma_{i,j}$ as the distance of $x^{(i,j)}$ to $\Gamma_{i,j}$. Following a bottom-up approach, we start with discussing selected building blocks employed in `fill`.

Building block 2.1 (`sort`) This building block is to sort a set of triplets S according to their position along Γ. We omit the indices i and j for clarity. Let us assume that Γ is piecewise smooth and fulfils an inner cone condition with angle β_{angle}. We first

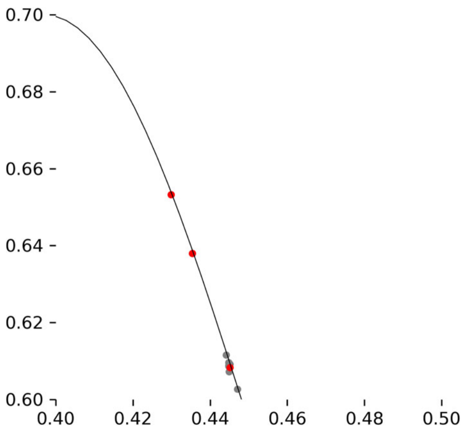


Fig. 3 Cluster removal by `removeclusters` in Example 4.1 for $\widetilde{S}_{1,2}$ (greyed out); points in $S^{(1)}_{1,2}$ are displayed in red. This figure is an excerpt of Fig. 10, right

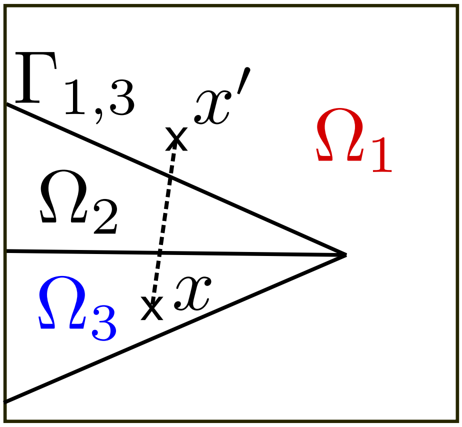


Fig. 4 Building block `bisection` can fail: Finding points near $\Gamma_{1,3}$ fails for the starting points $x \in \Omega_3$ and $x' \in \Omega_1$ shown, as $\overline{x'x}$ does not intersect $\Gamma_{1,3}$

search a triplet $x_{\text{start}} \in S$ closest to the boundary of Ω and assume x_{start} to be the first triplet in the sorted set. We initially set $\check{S} = \{x_{\text{start}}\}$. Let now the triplets in $\check{S} \subset S$ be already sorted with x_r being the last of those, $r > 1$. We consider the k_{sort} nearest triplets $y_1, \ldots, y_{k_{\text{sort}}}$ to x_r in $S \setminus \check{S}$, sorted by increasing distance to x_r. If for $s = 1$, $\angle(x_r - x_{r-1}, x_i - y_s) < \beta_{\text{angle}}$, we set $x_{r+1} = y_s$ and add x_{r+1} to $\check{S}$; otherwise, we repeat with $s + 1$. If $s > k_{\text{sort}}$, we store $\check{S}$, set $S = S \setminus \check{S}$, and repeat the sorting procedure until $S = \emptyset$ ending up with a finite number of disjoint ordered subsets. At the end, we combine all sorted subsets based upon the Euclidean distance between first and last points of the subsets reversing the order of a subset if necessary.

Remark 2.3 Allasia et al. [17] present a simpler sorting method than ours as they do not enforce the condition $\angle(x_r - x_{r-1}, x_i - y_s) < \beta_{\text{angle}}$. However, it may fail if x_{start} is not the true starting point and if additionally the points are unevenly distributed along Γ (Fig. 5) in contrast to ours. Of course, our sorting method can fail as well, but due to our experience, it is more reliable than Allasia's method and works sufficiently well. There are many more sophisticated sorting methods based, e.g. upon graph theory [23–

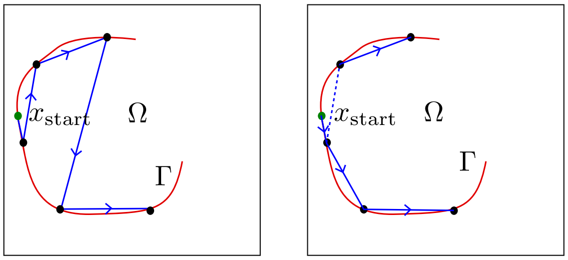


Fig. 5 Sorting due to Allasia [17] may fail (left; sorting indicated in blue), whereas our method succeeds in the present situation (right). We obtain two ordered subsets (direction of sorting is indicated by arrows which after combination yields the correct ordering). The dashed line represents a connection to the nearest neighbour rejected due to angle in our approach

26], which are more reliable than our approach, but more time-consuming and much harder to implement.

For describing building block `inipairs`, which is part of `fill`, we introduce another two building blocks, which will be used in `adapt` as well.

Building block 2.2 (`estcurv`) This building block estimates the curvature of a curve represented by a finite number of points near to it. Let be $S_{\mathrm{loc}} = \{x_1, \dots, x_r\}$, $r > 2$, a set of ordered points near Γ up to ε_b. This building block estimates the curvature c_ℓ of Γ in x_ℓ, $1 < \ell < r$ by least-squares fitting an approximation using Gaussian radial basis functions (RBFs, we refer to [19] for details) and then c_ℓ by the curvature of that approximation in x_ℓ. We choose the shape parameter of the RBFs as $\|x_1 - x_r\|/r$. We hereby assume that after shifting and suitable rotation, Γ can be locally represented as a graph of an unknown function. As the points in S_{loc} are located on Γ only up to ε_b, we penalise the second derivative of the RBF approximation subject to a maximal residual of ε_b. This coincides with the maximal deviation in the value at an approximation point. We employ Tikhonov regularisation with parameter estimation using Morozov's discrepancy principle.

If Γ cannot be considered a graph of a function even after rotation, we draw as a fallback a circle through the points with indices $\ell-1$, ℓ and $\ell+1$ and use the inverse of its radius for estimating c_ℓ. We estimate c_ℓ in the first or last point of S_{loc} by drawing a circle through the first or last three points in S_{loc}. For a pseudo-code of `estcurv`, we refer to the Appendix, Algorithm 2.

Building block 2.3 (`esterror`) This building block estimates the maximal deviation δ of a smooth curve from a straight line between two points on the curve with distance d. A straightforward calculation reveals that

$$\delta = 0.25cd^2 + 1/16c^3d^4 + \mathcal{O}(d^6), \tag{1}$$

where c denotes the maximal curvature of the curve between the two points. For S_{loc} as in Building block 2.2, we estimate the maximal deviation δ of Γ from the straight line between consecutive points x_ℓ and $x_{\ell+1}$ by replacing c with $\max\{c_\ell, c_{\ell+1}\}$ from Building block 2.2 (`estcurv`). Hereby, we rely on (1) and neglect higher order terms.

Now, we are prepared to discuss `fill`.

Algorithm 2.2 (`fill`) Building block 2.1 (`sort`) sorts all triplets in $S_{i,j}$ according to their position near $\Gamma_{i,j}$. We detect gaps in the representation of $\Gamma_{i,j}$ by $S_{i,j}$ by considering subsequent triplets x_ℓ and $x_{\ell+1}$. If the distance d_ℓ of x_ℓ to $x_{\ell+1}$ is larger than a user-prescribed threshold $\varepsilon_{\mathrm{gap}}$, we consider this a gap and aim to equidistantly add $R = \lceil d_\ell/\varepsilon_{\mathrm{gap}} \rceil$ triplets near $\Gamma_{i,j}$ between x_ℓ and $x_{\ell+1}$. To do so, building block `inipairs` places new points $z_{\ell,r}$, $1 \le r \le R$, equidistantly on $\overline{x_\ell^{(i,j)} x_{\ell+1}^{(i,j)}}$ and computes from these initial point pairs

$$x_{\ell,r}^{+} = z_{\ell,r} + \alpha n, \quad x_{\ell,r}^{-} = z_{\ell,r} - \alpha n, \quad 1 \le r \le R.$$

Here, n denotes the (estimated) outer normal unit vector of Ω_i near $x_{\ell,r}$. Applying Building block 2.3 (`esterror`) to $S_{\mathrm{loc}} = \{x_{\ell-2}^{(i,j)}, \ldots x_{\ell+2}^{(i,j)}\}$ and some safeguarding leads to

$$\alpha = \min\{\varepsilon_{\mathrm{safemax}} d_\ell, \max\{\delta, \varepsilon_{\mathrm{safemin}} \varepsilon_b\}\} \tag{2}$$

with user-prescribed safety factors $\varepsilon_{\mathrm{safemax}}$ and $\varepsilon_{\mathrm{safemin}}$. For algorithmic details, we refer to the pseudo-code in the Appendix, Algorithm 4.

Remark 2.4 Having a local RBF approximation of Γ at hand when computing c in `estcurv`, it seems to be straightforward for efficiency reasons to choose points $z_{\ell,r}$ on that RBF approximation instead of just subdividing a straight line. Numerical experiments did not show any significant advantage of that approach compared to ours for `fill`. This could be related to the uneven distribution of points on Γ near gaps to fill, which may decrease the quality of approximation. Therefore, we stick to the easier approach presented here. However, estimating the curvature using Building block 2.2 (`estcurv`) is sufficiently reliable for efficiently computing starting pairs.

However, the pairs of starting points (aka starting pairs) obtained from `inipairs` are not necessarily valid. We call a starting pair for approximating $\Gamma_{i,j}$ valid, if one of its points belongs to Ω_i and the other one to Ω_j. Therefore, we introduce Building block 2.4 (`startpairs`).

Building block 2.4 (`startpairs`) This building block obtains a valid starting pair from a pair of points $(x_{\ell,r}^+, x_{\ell,r}^-)$. If the starting pair is already valid, we return it as the result. If $x_{\ell,r}^+$ or $x_{\ell,r}^-$ belongs to a third class, we stop without result. If both points belong to the same class, we reflect $x_{\ell,r}^+$ on $z_{\ell,r}$ obtaining $x'_{\ell,r}$ (Fig. 6). If both $(x_{\ell,r}^+, x'_{\ell,r})$ and $(x_{\ell,r}^-, x'_{\ell,r})$ still belong to the same class, we repeat this process with changing roles and escalating distances at most k_{rep} times (typically $k_{\mathrm{rep}} = 3$) and stop, if any of the resulting point pairs is valid or one of the points belongs to a third class. In the latter case, we search a valid starting pair by scattering around $0.5(x_{\ell,r}^- + x'_{\ell,r})$ as fallback. For a pseudo-code of `startpairs`, we refer to the Appendix, Algorithm 3.

Remark 2.5 1. We iterate the process of filling gaps in `fill`, as the arclength of Γ between two subsequent triplets may considerably exceed the length of the straight line between them, such that after a first pass, the mutual distance of subsequent triplets may still exceed $\varepsilon_{\mathrm{gap}}$ in some cases. In our implementation, we approximate the arclength of Γ by the arclength of the approximating polygonal line given by the triplets computed so far.

2. If only two or even less triplets are known on $\Gamma_{i,j}$, estimating curvature is impossible with `estcurv`, and computing α by (2) fails. For this case, we implemented fallbacks.

Using `bisection`, `fill` creates new triplets, yielding new sets $\overline{S}_{i,j}$.

Remark 2.6 If Ω_i is not simply connected, some $\Gamma_{i,j}$ may consist of several components. We detect this using `fill` (compare the pseudo-code of `fill` in the Appendix, Algorithm 4). If there are some significant gaps which can not be filled, it indicates the presence of several components. We then subdivide $\overline{S}_{i,j}$ correspondingly and proceed with every subset separately.

Fig. 6 Creating valid starting pairs in `fill` with `startpairs`: While $(x^+_{\ell,1}, x^-_{\ell,1})$ is valid, $(x^+_{\ell,2}, x^-_{\ell,2})$ is not, as both points belong to the same class. However, $(x^-_{\ell,2}, x'_{\ell,2})$ is valid

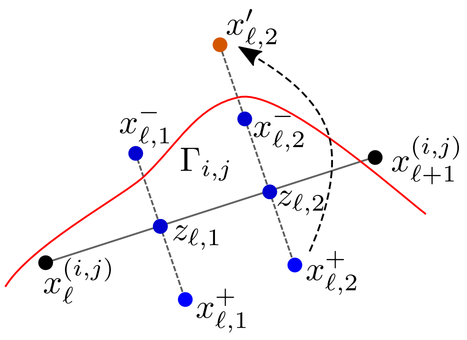


Figure 11 indicates that even with filling gaps, $\overline{S}_{i,j}$ may not appropriately represent $\Gamma_{i,j}$, as parts of $\Gamma_{i,j}$ before the first known triplet x_1 and after the last known may be neglected. Algorithm `expand` is used to expand $\overline{S}_{i,j}$ to a representation of the complete curve $\Gamma_{i,j}$. It relies on several building blocks, which we discuss first.

Building block 2.5 (`extrapolate`) For a given ordered set S with $n \geq 2$ triplets close up to ε_b to a curve Γ and with average distance d_{avg}, we fit a polynomial with degree $n-1$ in local coordinates. We compute these coordinates by least-squares-fitting a line to S. As points in S are located on Γ up to ε_b only, we do not interpolate, but penalise the second derivative in a least-squares approximation. Following Morozov's discrepancy principle, we regularise such that the maximal residual is approximately ε_b.

Building block 2.6 (`stepsize`) Provided that Γ extends sufficiently far before $x_1 \in S$, it seems to be straightforward to seek for a new triplet with distance ε_{gap} to x_1. However, we limit the step size for extrapolation based upon the curvature c of Γ in the vicinity of x_1. Extrapolating far is unreliable in case of large curvature c, and `adapt` will insert additional points afterwards in that region anyway for accuracy reasons, such that it is much more efficient to adjust the step length to the local properties of Γ beforehand. Let us assume that a polygonal final approximation of Γ may deviate at most by ε_{err} from Γ. We compute the step size which would lead to a deviation of ε_{err} from a straight line segment between x_1 and the new triplet yet to compute. This is a natural upper bound for the step size l_{extra} in extrapolation.

Rearranging (1) and neglecting higher-order terms leads to

$$l_{\max} = \frac{2}{c^2}\left(\sqrt{1 + 4c\varepsilon_{\text{err}}} - 1\right) \tag{3}$$

for the maximal admissible step length $l_{\max}$. However, evaluating (3) is numerically unstable if $c\varepsilon_{\text{err}}$ is small. We set $(c\varepsilon_{\text{err}})^2 = v$ and search for the roots $v_{\min}$ and $v_{\max}$ of $v^2 + 4v - 16cd$. According to Vieta, $v_{\min} = -\left(2 + \sqrt{4 + 16cd}\right)$ and $v_{\max} = -16cd/v_{\min}$. Resubstituting v yields $l_{\max} = 4\sqrt{d/(-cv_{\min})}$. Some safeguarding of this result leads to a step length of

$$l_{\text{extra}} = \min\{\varepsilon_{\text{gap}}, \beta_{\text{growth}} d_{\text{avg}}, l_{\max}\},$$

where we estimate c using Building block 2.2 (`estcurv`) applied to $S_{\text{loc}} = \{x_1^{i,j)}, \dots, x_{k_{\text{extra}}}^{(i,j)}\}$ with a user-defined parameter k_{extra}. The term $\beta_{\text{growth}} d_{\text{avg}}$ increases robustness, as extrapolation is reliable only near the points to extrapolate, and it may happen that $d_{\text{avg}} \ll \varepsilon_{\text{gap}}$.

Algorithm 2.3 (`expand`) This algorithm aims at finding triplets near $\Gamma_{i,j}$ beyond the first or last known in $\overline{S}_{i,j}$ until the start or end of $\Gamma_{i,j}$ is reached or $\Gamma_{i,j}$ turns out to be a closed curve. For the sake of simplicity, we refer in what follows to finding triplets before the first one in $\overline{S}_{i,j}$. Finding triplets near $\Gamma_{i,j}$ beyond the last one in $\overline{S}_{i,j}$ works analogously. We add a new triplet before x_1 by extrapolating an approximation γ of $\Gamma_{i,j}$ with Building block 2.5 (`extrapolate`) setting $S = \left\{x_1^{(i,j)}, \dots, x_{k_{\text{extra}}}^{(i,j)}\right\}$ with $x_\ell \in \overline{S}_{i,j}$ and choose the step size according to Building block 2.6 (`stepsize`). This way, we obtain an extrapolating curve γ and some s_0 such that $\|x_1 - \gamma(s_0)\| \approx l_{\text{extra}}$. We then create a point pair $(x_{s_0}^+, x_{s_0}^-)$ based on $\gamma(s_0)$ in a similar way as in `fill`. Computing a valid starting pair with Building block 2.4 (`startpairs`) and subsequent bisection yields a new triplet in $\overline{S}_{i,j}$. We repeat this process until we reach the true starting point of $\Gamma_{i,j}$. As heuristic criterion for $\gamma(s_0)$ exceeding this starting point, we consider

$$x_{s_0}^+ \notin (\Omega_i \cup \Omega_j) \vee x_{s_0}^- \notin (\Omega_i \cup \Omega_j) \tag{4}$$

(Fig. 7). In this case, we employ Building block 2.7 (`reducestepsize`) for obtaining a valid pair of points $(x_{\tilde{s}}^+, x_{\tilde{s}}^-)$, which represents the starting point of $\Gamma_{i,j}$. After adding it to $\overline{S}_{i,j}$, `expand` terminates. If Γ_{ij} is closed, expanding $\overline{S}_{i,j}$ as described above would lead to an endless loop. Therefore, we start expanding $\overline{S}_{i,j}$, but detect after every addition of a triplet, if $\Gamma_{i,j}$ is closed. If so, we stop expanding and resort. We skip the details to keep the presentation uncluttered. Algorithm `expand` yields approximating sets $\breve{S}_{i,j}$.

Building block 2.7 (`reducestepsize`) Using the parameter value s_1 corresponding to $x_1^{(i,j)}$ as lower bound and s_0 as upper bound, we obtain $\tilde{s}$, which leads to a valid starting pair $(x_{\tilde{s}}^+, x_{\tilde{s}}^-)$ based upon $\gamma(\tilde{s})$ and fulfils $\|\gamma(\tilde{s}) - \gamma(s')\| < \varepsilon_b$ by bisection with respect to condition (4). Here, s' denotes the second to last parameter in the bisection process (compare Fig. 7).

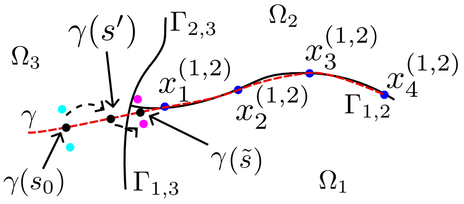


Fig. 7 Scheme of expanding $S_{1,2}$ until its end. Starting from $x_1^{(1,2)}, \dots, x_4^{(1,2)}$ (displayed as blue dots), we construct γ by `extrapolate`. As $(x_{s_0}^+, x_{s_0}^-)$, displayed as cyan-coloured dots, fulfils (4), we apply bisection with respect to the parameter s until a valid starting pair based upon $\gamma(\tilde{s})$ can be constructed (displayed in magenta), from which we compute the final triplet in $S_{1,2}$ by `bisection`

Algorithm 2.4 (`adapt`) Based on `esterror`, this algorithm inserts a triplet (approximately) halfway between consecutive triplets x_ℓ and $x_{\ell+1}$, if `esterror` indicates an error larger than ε_{err} and removes a triplet, if the estimated error of both line segments the triplet belongs to is smaller than $\varepsilon_{\text{coarse}}$. In contrast to `fill`, we employ the local RBF approximation from `estcurv` for computing an initial point x_{new} between x_ℓ and $x_{\ell+1}$ when refining. With $x^{\pm}_{\text{new}} = x_{\text{new}} \pm \alpha' n$, we then proceed as in `fill`. We compute α' according to (2), but replace δ by $\delta' = 1/16c^3d^4$, as the error due to (1) refers to an approximation by line segments and is overly pessimistic for an RBF approximation. For robustness, we never delete consecutive triplets in one pass of the adaptive loop. After at most k_{adap} refinement and coarsening sweeps, we end up with final sets $\hat{S}_{i,j}$.

2.2 Approximating $\Gamma_{i,j}$ in 3D

For approximating $\Gamma_{i,j}$ in three dimensions, we stick to the general procedure for approximating these sets in two dimensions (Fig. 1). While `initialise` remains unchanged, `fill`, `expand` and `adapt` differ from their 2D counterparts, as these exploit ordering of the points on a decision curve or fault line. There is however no straightforward ordering of points on a surface. Because `fill` and `adapt` rely on `esterror` as in two dimensions, we discuss error estimation first.

Building block 2.8 (`esterror`) This building block aims at estimating the maximal error e of a linear triangulation of Γ based upon a finite set of points S near Γ up to ε_b. Let $\mathcal{T}$ be a Delaunay triangulation of $S_{\text{loc}} \subset S$ and let us assume that Γ can be represented on the support of $\mathcal{T}$ by an unknown function g after appropriate change of coordinates. For some triangle $T \in \mathcal{T}$, it holds according to [27, Theorem 4.1]

$$\|g - I_T g\|_{T,\infty} \leq \frac{1}{2}\left(R^2 - d^2\right)|g|_{2,\infty,T}.$$

Here, R describes the radius of the circumcircle of T, d the distance from its centre to T and $I_T g$ the Lagrange interpolant of g on T; with $|g|_{2,\infty,T}$, we denote the L_∞-norm of the second derivative of g on T. It remains to estimate $|g|_{2,\infty,T}$. As g is unknown, we employ an RBF approximation φ as in Building block 2.2 (`estcurv`) instead and approximate $|g|_{2,\infty,T}$ by evaluating its second derivative in the vertices of T and its centre. The maximum of these four values yields the desired approximation ϕ of $|g|_{2,\infty,T}$ and therefore

$$e \approx \frac{1}{2}\left(R^2 - d^2\right)\phi \tag{5}$$

Algorithm 2.5 (`fill`) For any triplet x in $S_{i,j}$, we search the k_{near} nearest neighbours $x_1, \ldots, x_{k_{\text{near}}}$ and switch to a local 2D coordinate system by computing the optimal fitted plane in the sense that the sum of the squared distances between the points and the plane is minimal (see [28]). We then compute in local coordinates a Delaunay triangulation of the patch $S_{\text{loc}} = \{x, x_1, \ldots, x_{k_{\text{near}}}\}$. If the maximal edge length of a triangle in this triangulation exceeds ε_{gap}, we mark the centre of that triangle as starting point for constructing a valid starting pair similar as in Building `inipairs`, as long as the triangle is not too anisotropic, i.e. does not contain angles approaching 0° or 180°. However, large gaps in $S_{i,j}$ are not covered by this local approach (Fig. 8). If there is a gap in $S_{i,j}$ and if the current point is at the boundary of this gap, it will be at the boundary of the current patch. We detect this by exploiting the local coordinate system. We assume that a patch in the vicinity of a large gap is at least slightly elongated in tangential direction to the patch boundary. Therefore, we consider the first local coordinate. If the first local coordinate of the current point is almost the minimum or maximum of all respective first coordinates of the patch, then it is at its boundary, and we assume a large gap to fill. In this case, we construct x_{new} on the elongated line between the centre of gravity of the patch and x, enrich S_{local} by x_{new} and continue as described above. Looping over all triplets in $S_{i,j}$, removing duplicate starting points and starting points which are extremely close enables us to compute triplets using Building block `bisection` similar as in Algorithm 2.2 (`fill`). This yields an enriched representation of $\Gamma_{i,j}$. Repeating this procedure with the enriched set until no gaps are detected anymore or a maximal number $k_{textrep}$ of filling cycles is reached leads to $\overline{S}_{i,j}$. We present a pseudo-code of `fill` in the Appendix, Algorithm 5, to which we refer for algorithmic details.

Algorithm 2.6 (`expand`) When expanding a representation $\overline{S}_{i,j}$ of $\Gamma_{i,j}$, we distinguish between expansion towards inner boundaries $\Gamma_{i,j} \cap \Gamma_{k,\ell}$ and expansion towards outer boundaries $\Gamma_{i,j} \cap \partial\Omega$. We consider expanding to inner boundaries first. Let $\overline{S}^I_{i,j} \subseteq \overline{S}_{i,j}$ be all triplets closer than $0.75\varepsilon_{\text{gap}}$ to a triplet in another $S_{k,\ell}$. For some $x \in \overline{S}^I_{i,j}$, let x^* be the triplet in $S_{k,\ell}$ closest to x. We estimate the normal vector $\mathfrak{n}$ of $\Gamma_{i,j}$ in x and project $\overline{xx^*}$ on the corresponding tangential plane at x, yielding $\mathfrak{t}$. Let $x' \in \overline{S}_{i,j}$ be the nearest triplet to x fulfilling $\angle(-\mathfrak{t}, \overline{xx'}) < \alpha_{\text{expand}}$ and $\mathcal{E}$ the plane which contains x and is spanned by $\overline{xx'}$ and $\mathfrak{n}$. We reduce the expansion to the two-dimensional case by expanding the curve $\Gamma_{i,j} \cap \mathcal{E}$ with Algorithm 2.3 (`expand`) from Sect. 2.1, using x and x' as initial set of triplets (compare Fig. 8, right).

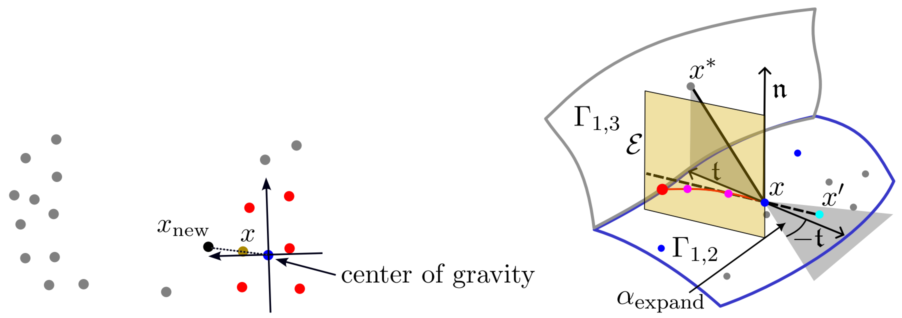


Fig. 8 Left: Gradually filling large gaps in $S_{i,j}$ (displayed as grey dots). The triplet x is represented in golden colour; the triplets in S_{local}, displayed in red, define a local coordinate system sketched with black arrows. The centre of gravity of S_{local} is shown as a blue dot. Right: Expansion of $\overline{S}_{1,2}$ to the inner boundary $\Gamma_{1,2} \cap \Gamma_{1,3}$. Triplets in $\overline{S}_{1,2} \setminus \overline{S}^I_{1,2}$ are displayed as grey dots, triplets in $\overline{S}^I_{1,2}$ as blue dots. 2D-Expanding $\{x, x'\}$ on $\mathcal{E} \cap \Gamma_{1,2}$ yields a new triplet near $\Gamma_{1,2} \cap \Gamma_{1,3}$, displayed as red dot

For expanding to outer boundaries, we construct $\overline{S}_{i,j}^{B}$ as follows: For each coordinate direction i, we select $n_{\text{expand},i}$ points in $S_{i,j}$ which are minimal or maximal with respect to this coordinate and set $\overline{S}_{i,j}^{B} = \overline{S}_{i,j}^{B} \setminus \overline{S}_{i,j}^{I}$ in order to avoid duplicate triplets. For determining $n_{\text{expand},i}$, we rely on the axis-parallel bounding box of $\overline{S}_{i,j}$ with sizes b_1, b_2, b_3. Then, $n_{\text{expand},i} = \lceil \max\{b_{i+1 \mod 3}, b_{i \mod 3+1}\}/\varepsilon_{\text{gap}} \rceil$. From these triplets, we select the ones which are either closer than ε_{gap} to one of the boundary facets or fulfil

$$\angle(\mathfrak{n}, \mathfrak{n}_{\text{outer}}) > \alpha_{\text{expbound}}, \tag{6}$$

where $\mathfrak{n}_{\text{outer}}$ stands for the outer normal vector of the assigned boundary facet. Figure 9 illustrates the purpose of condition (6). For any triplet in $\overline{S}_{i,j}^{B}$, we proceed as for expanding to inner boundaries, but with $\mathfrak{t} = \mathfrak{n}_{\text{outer}}$. All new triplets in the same facet F of $\partial\Omega$ constitute the set $\overline{S}_{i,j}^{F}$. As this set represents the curve $\Gamma_{i,j} \cap F$ in the plane F, we are confronted with approximating a decision curve or fault line in two dimensions. Therefore, we apply Algorithms 2.2 (`fill`) and 2.3 (`expand`) to $\overline{S}_{i,j}^{F}$. After adding the new triplets on the boundary of Γ, we apply Algorithm 2.5 again and end up with an enlarged set $\breve{S}_{i,j}$.

Algorithm 2.7 (`adapt`) In contrast to the two-dimensional case, we do not adaptively coarse the set of triplets. While this is possible and does not harm accuracy, it complicates computing a surface mesh from the set of triplets representing $\Gamma_{i,j}$.

For adaptive refinement, we do not rely on a global triangulation of $\breve{S}_{i,j}$. Instead, for a given triplet $x \in \breve{S}_{i,j}$, let $\breve{S}_{\text{loc}}$ consist of the k_{near} nearest triplets in $\breve{S}_{i,j}$ to x. We least-squares fit a plane $\mathcal{E}$ to $\breve{S}_{i,j}^{\text{loc}}$ as in Algorithm 2.5 (`fill`) and create a Delaunay triangulation $\mathcal{T}$ of $\breve{S}_{i,j}^{\text{loc}}$ projected to $\mathcal{E}$. For each non-degenerated triangle in $\mathcal{T}$, we estimate the error applying `esterror` to $\breve{S}_{\text{loc}}^{(i,j)}$. If the estimated error exceeds ε_{err}, we employ the centre of the triangle as a starting point for adding a new triplet. Looping over all $x \in \breve{S}_{i,j}$ naturally leads to many duplicate or very close starting points which need to be eliminated before adding triples via bisection. However, these duplicates do not harm the efficiency of our method, as no function evaluations are required before computing triplets from starting points. We enrich $\breve{S}_{i,j}$ with the new triples created and repeat the refinement procedure at most k_{adap} times or until no more starting points for computing triplets have been created.

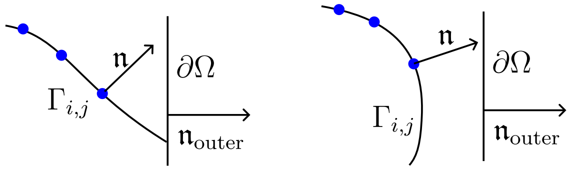


Fig. 9 Suitable (left) and unsuitable (right) triplets for expanding to a domain boundary

3 Direct and inverse acoustic scattering problem

As a real-world application of our algorithm, we will consider the inverse acoustic scattering problem in Section 4.3, which forms the mathematical background for some types of non-destructive testing and imaging and therefore is of high practical interest. As a preparation to our numerical experiments, we provide in this section a brief introduction to direct and inverse scattering for the acoustic transmission problem. For more details, see [10].

Let the scatterer $D \subset \mathbb{R}^3$ be a given bounded domain with boundary ∂D assumed to be of class $C^{2,\alpha}$. The normal unit vector ν points into the exterior $E = \mathbb{R}^3 \setminus \overline{D}$ of the scatterer. The exterior E assumed to be simply connected is an infinite homogeneous isotropic non-absorbing acoustic medium which is characterised by the mass density ϱ_e, mean compressibility κ_e, and sound speed $c_e = 1/\sqrt{\kappa_e \varrho_e}$. Likewise, the interior of D is characterised by ϱ_i, κ_i, and $c_i = 1/\sqrt{\kappa_i \varrho_i}$. The given scatterer is excited by a time-harmonic acoustic incident plane wave of the form

$$u^{\mathrm{inc}}(x; \widehat{d}) = \mathrm{e}^{\mathrm{i}k_e x \cdot \widehat{d}}, \qquad x \in \mathbb{R}^3, \tag{7}$$

where $k_e = \omega / c_e$ is the wave number of the acoustic wave in the host medium, $\omega > 0$ the angular frequency, and $\widehat{d} \in \mathbb{S}^2$ the direction of incidence with $\mathbb{S}^2 = \{x \in \mathbb{R}^3 : \|x\| = 1\}$ the unit sphere, where $\|\cdot\|$ denotes the standard Euclidean norm in $\mathbb{R}^3$. Note that the incident field also depends on k_e, but this dependence is suppressed.

The incident wave (7) interferes with the penetrable scatterer and creates two waves. The first wave is the scattered field $u^{\mathrm{sca}}(x; \widehat{d})$ defined for $x \in E$ propagating outward and the second wave is the transmitted field $u^{\mathrm{int}}(x; \widehat{d})$ defined for $x \in D$. The total field in E denoted by $u^{\mathrm{ext}}(x; \widehat{d})$ is the superposition of $u^{\mathrm{int}}(x; \widehat{d})$ and $u^{\mathrm{sca}}(x; \widehat{d})$ each of which satisfies the Helmholtz equation (the reduced wave equation) in E with wave number k_e. Likewise, the transmitted field satisfies the Helmholtz equation in D with wave number k_i. Precisely, we have

$$\begin{aligned} \Delta u^{\mathrm{int}}(x; \widehat{d}) + k_i^2 u^{\mathrm{int}}(x; \widehat{d}) &= 0, & x &\in D, \\ \Delta u^{\mathrm{ext}}(x; \widehat{d}) + k_e^2 u^{\mathrm{ext}}(x; \widehat{d}) &= 0, & x &\in E. \end{aligned}$$

Due to the continuity of the acoustic pressure and the normal component of the particle velocity across ∂D yields the transmission boundary conditions

$$u^{\mathrm{int}}(x; \widehat{d}) = u^{\mathrm{ext}}(x; \widehat{d}) \quad \text{and} \quad \partial_\nu u^{\mathrm{int}}(x; \widehat{d}) = \tau \partial_\nu u^{\mathrm{ext}}(x; \widehat{d}), \qquad x \in \partial D,$$

where $\tau = \varrho_i / \varrho_e > 0$ is the mass density ratio of the two media. To ensure a well-posed boundary value problem, the scattered field $u^{\mathrm{sca}}(x; \widehat{d})$ needs to satisfy the Sommerfeld radiation condition

$$\lim_{r \to \infty} r \left(\partial_r u^{\mathrm{sca}}(x; \widehat{d}) - \mathrm{i}k_e u^{\mathrm{sca}}(x; \widehat{d}) \right) = 0$$

with $r = \|x\|$. The classical acoustic transmission problem reads: find the functions $u^{\text{int}}(x; \widehat{d}) \in C^2(D) \cap C^1(\overline{D})$ and $u^{\text{sca}}(x; \widehat{d})) \in C^2(E) \cap C^1(E)$ satisfying

$$\Delta u^{\text{int}}(x; \widehat{d}) + k_i^2 u^{\text{int}}(x; \widehat{d}) = 0, \quad x \in D, \tag{8}$$

$$\Delta u^{\text{ext}}(x; \widehat{d}) + k_e^2 u^{\text{ext}}(x; \widehat{d}) = 0, \quad x \in E, \tag{9}$$

$$u^{\text{int}}(x; \widehat{d}) - u^{\text{sca}}(x; \widehat{d}) = u^{\text{inc}}(x; \widehat{d}), \quad x \in \partial D, \tag{10}$$

$$\frac{1}{\tau}\partial_\nu u^{\text{int}}(x; \widehat{d}) - \partial_\nu u^{\text{sca}}(x; \widehat{d}) = \partial_\nu u^{\text{inc}}(x; \widehat{d}), \quad x \in \partial D, \tag{11}$$

$$\lim_{r\to\infty} r\left(\partial_r u^{\text{sca}}(x; \widehat{d}) - \mathrm{i}k_e u^{\text{sca}}(x; \widehat{d})\right) = 0, \quad r = \|x\| \tag{12}$$

3.1 The direct acoustic transmission problem

Given the incident field (i.e. the direction of incidence $\widehat{d}$ and the wave number k_e), the scatterer D (hence also its boundary ∂D), the wave number k_i and the parameter τ, one has to solve (8)–(12) for $u^{\text{int}}(x; \widehat{d})$ and $u^{\text{sca}}(x; \widehat{d})$. In the direct problem, one is only interested in the far-field $u^\infty(\widehat{x}; \widehat{d})$ of $u^{\text{sca}}(x; \widehat{d})$ which is given by

$$u^{\text{sca}}(x; \widehat{d}) = \frac{\mathrm{e}^{\mathrm{i}k_e\|x\|}}{\|x\|} u^\infty(\widehat{x}; \widehat{d}) + \mathcal{O}\left(\|x\|^{-2}\right), \qquad \|x\| \to \infty$$

uniformly in all directions $\widehat{x} \in \mathbb{S}^2$. The far-field can be found by evaluating an integral equation over ∂D given two density functions determined by first solving a 2×2 system of integral equation of the second kind over ∂D (see [10, Sect. 4.1]). Of course, the integral equations at hand cannot be solved analytically and have to be solved numerically for example by the boundary element collocation method (see [29, Chapter 5] for more details).

To sum up, in the direct acoustic transmission problem, one is interested in $u^\infty(\widehat{x}; \widehat{d})$ for $\widehat{x} \in \mathbb{S}^2$ given the scatterer's boundary ∂D and the direction of incidence $\widehat{d} \in \mathbb{S}^2$. The parameters k_e, k_i and τ are given.

3.2 The inverse acoustic transmission problem

The parameters k_e, k_i, and τ are given. In the inverse acoustic transmission problem one tries to find/reconstruct the domain's boundary from the knowledge of the far-field patterns $u^\infty(\widehat{x}; \widehat{d})$ for all $\widehat{x}, \widehat{d} \in \mathbb{S}^2$. This can be achieved with the factorization method originally invented by Kirsch (see [8]). The theoretical justification of the factorization method for the acoustic transmission problem is given in [10, Chapter 3] and shown to work practically in [10, Chapter 4]. We briefly outline the algorithm: Assume that the far-field data are given for $\hat{x}_i$ and $\hat{d}_j$ with $i, j \in \{1, \ldots, m\}$ stored in the matrix $A \in \mathbb{C}^{m\times m}$. First, compute a singular decomposition of $A = U\Lambda V^*$ with $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_m)$. For a given point $z \in \mathbb{R}^3$ compute the expansion coefficient of

$$r_z = \left(\exp\left(-\mathrm{i}k_e z\cdot \hat{d}_j\right)\right)_{j=1,\ldots,m} \in \mathbb{C}^m$$

with respect to V by

$$\varrho_\ell^{(z)} = \sum_{j=1}^{M} V_{j,\ell} \mathrm{e}^{-\mathrm{i}k_e z \cdot d_j} , \quad \ell = 1, \dots, M ,$$

which is a matrix–vector multiplication $\varrho^{(z)} = V^\top r_z$. Finally, we compute

$$W(z) = \left[\sum_{\ell=1}^{M} \frac{|\varrho_\ell^{(z)}|^2}{|\lambda_\ell|} \right]^{-1}$$

and plot the isosurfaces of $z \mapsto W(z)$. The values of $W(z)$ should be much smaller for $z \notin D$ than those lying within D. The threshold value can be approximately determined by a scatterer such as the unit sphere and then reused for other scatterers as well. Note that until now, an equidistant set of points N within a predefined box have been used to find the values of $W(z)$ leading to an amount of $N \times N \times N$ function evaluations for such a "sampling" method (see also [11] for other sampling methods). This can be considerably reduced as shown in the next section.

4 Numerical tests and applications

4.1 Test cases related to the detection of faults

Test problem 4.1 For $\Omega = [0, 1]^2$, we consider the following partition: Let $\Omega_3 := \{(x-1)^6 + (y-0.5)^6 < 0.005\} \cap \Omega$, $\Omega_1' := \{y \leq 0.6 + 0.1 \sin(10\pi x^{1.5})\} \cap \Omega$ and $\Omega_2' := \Omega \setminus \Omega_1'$. Then, we set $\Omega_1 := \Omega_1' \setminus \Omega_3$ and $\Omega_2 := \Omega_2' \setminus \Omega_3$ and study the partition $\Omega = \Omega_1 \cup \Omega_2 \cup \Omega_3$ (Fig. 10, left).

Example 4.1 For Test problem 4.1, we choose an initial sampling set X consisting of 50 Halton-distributed points (Fig. 10, left) and set $k_{\text{near}} = 10$ and $\varepsilon_b = 0.001$. The set $\mathcal{M}$ consists of 47 additional points represented as black dots; $\mathcal{M}^2$ contains 49 points displayed in blue. $\widetilde{S}_{1,2}$ contains 43 triplets, $\widetilde{S}_{1,3}$ contains 26, and $\widetilde{S}_{2,3}$ contains 7 (Fig. 10, right). After cluster removal, we obtain $|S_{1,2}| = 23$, $|S_{1,3}| = 13$ and $|S_{2,3}| = 4$ (Fig. 3 and Fig. 10, right). We then ordered the sets $S_{i,j}$ according to Building block 2.1 with $\beta_{\text{angle}} = \arccos(-0.9) \approx 154°$ and $k_{\text{sort}} = 5$. By filling gaps with $\varepsilon_{\text{gap}} = 0.05$, $\varepsilon_{\text{safemin}} = 0.95$ and $\varepsilon_{\text{safemax}} = 0.25$, we obtain $|\overline{S}_{1,2}| = 33$, $|\overline{S}_{1,3}| = 18$, $|\overline{S}_{2,3}| = 4$ (Fig. 11). We will stick to the values of the parameters given here for all subsequent numerical examples. We set $k_{\text{extra}} = 4$, $\varepsilon_{\text{err}} = 0.001$, $\varepsilon_{\text{coarse}} = 0.0001$ and obtain $|\breve{S}_{1,2}| = 43$, $|\breve{S}_{1,3}| = 28$, and $|\breve{S}_{2,3}| = 18$ (Fig. 12, left). Algorithm `adapt` with $k_{\text{adap}} = 4$ yields $|\hat{S}_{1,2}| = 77$, $|\hat{S}_{1,3}| = 26$, and $|\hat{S}_{2,3}| = 22$ (Fig. 12, right). Computing these sets requires 1088 classifications (Fig. 13).

For approximating the RMS error for the polygonal lines defined by the sets $\hat{S}_{i,j}$, we replace the corresponding $\Gamma_{i,j}$ by fine polygonal approximations $\tilde{\Gamma}_{i,j}$ constructed such that the maximal deviation from the true $\Gamma_{i,j}$, which are known in our examples,

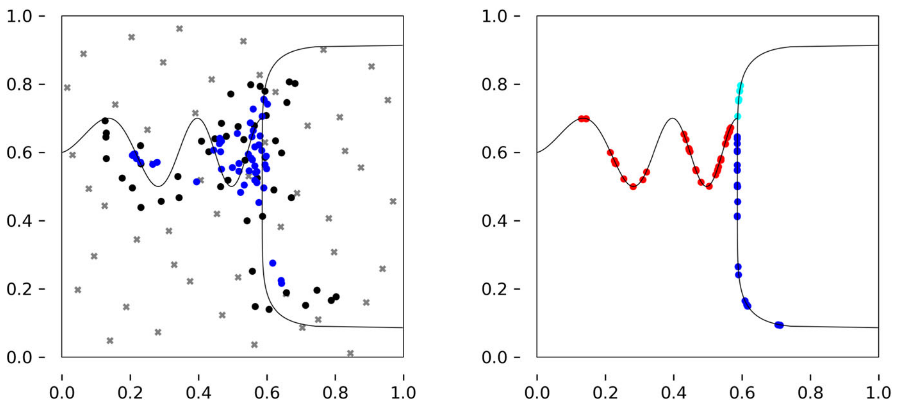


Fig. 10 Left: Partition of $\Omega = [0,1]^2$ according to Example 4.1 indicated by grey solid lines with the initial point set X displayed as grey crosses. We moreover show $\mathcal{M}$ (black points) and $\mathcal{M}^2$ (blue points); Right: $S_{1,2}^{(1)}$ (red points), $S_{1,3}^{(1)}$ (blue points), and $S_{2,3}^{(2)}$ (cyan-coloured points)

does not exceed 10^{-7}. We then construct a test set by choosing 500 equidistant points on each polygonal line defined by $\hat{S}_{i,j}$ and compute the RMS error of this test set with respect to corresponding $\tilde{\Gamma}_{i,j}$. For $\varepsilon_b = \varepsilon_{\text{err}} = 0.001$ and $\varepsilon_{\text{coarse}} = 0.0001$, the RMS error is $7.66 \cdot 10^{-4}$, whereas for $\varepsilon_b = \varepsilon_{\text{err}} = 10^{-4}$ and $\varepsilon_{\text{coarse}} = 10^{-5}$, the RMS error is $6.29 \cdot 10^{-5}$. In the latter case, our algorithm requires 2406 function evaluations.

We furthermore consider test cases which have been defined in the context of the detection of fault lines or fault surfaces.

Test problem 4.2 Following Allasia et al. [17] and Gutzmer et at. [13], we set $x_M = (0.5, 0.5)^\top$, $x = (x_1, x_2)^\top$, $\Omega = [0,1]^2$ and consider the function

$$f(x_1, x_2) = \begin{cases} 1 + 2\lfloor 3.5 \|(x_1, x_2)\|_2 \rfloor , & \|(x - x_M)\|_2 < 0.4 \\ 0 & , \text{otherwise} \end{cases} \tag{13}$$

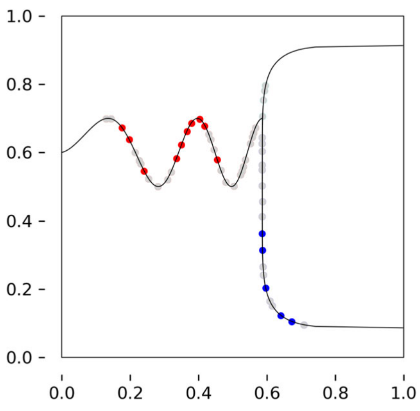


Fig. 11 Algorithm 2.2 (`fill`) for Test problem 4.1. Previously existing points are greyed out in $\overline{S}_{i,j}^{(i)}$. Otherwise, we stick to the colouring scheme of Fig. 10

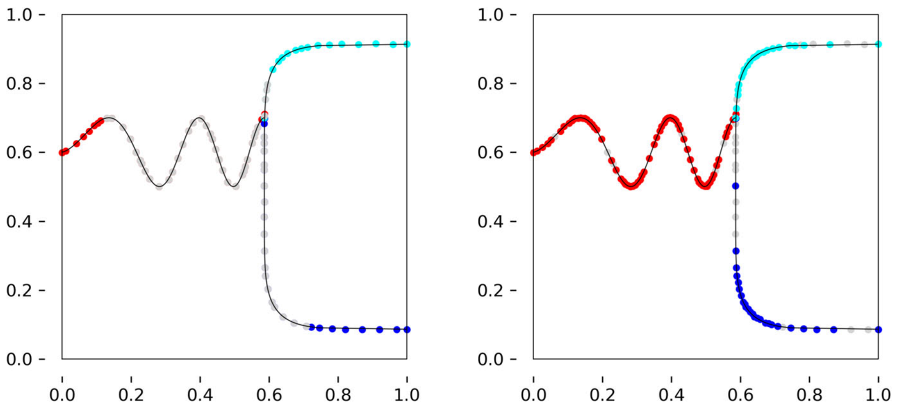


Fig. 12 Sets $\breve{S}_{i,j}^{(i)}$ (left) and $\hat{S}_{i,j}^{(i)}$ (right) for Test problem 4.1. We stick to the colouring scheme of Fig. 10

This function is piecewise constant with smooth fault lines; it holds

$$\begin{aligned}\Gamma = \bigcup \Gamma_{i,j} = \quad & \left\{x \in \mathbb{R}^2 \ : \ \|x - x_m\|_2 = 0.4\right\} \\ & \cup \{\|x\|_2 = 4/7 \ : \ \|x - x_m\|_2 < 0.4\} \\ & \cup \{\|x\|_2 = 6/7 \ : \ \|x - x_m\|_2 < 0.4\} \ .\end{aligned}$$

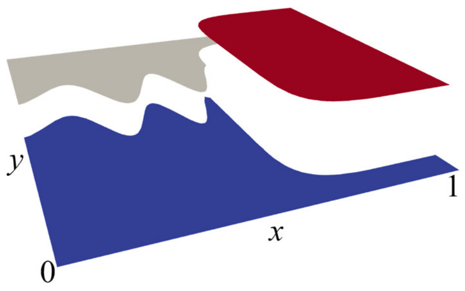
The set $\Gamma_{1,3}$ consists of two separate components (Fig. 14, left).

Test problem 4.3 We set

$$f(x_1, x_2) = \begin{cases} 3 \ , \ x_1 > 0.6 \\ 1 \ , \ x_1 < 0.5 \\ 2 \ , \ \text{otherwise} \end{cases} . \tag{14}$$

The set Γ consists of two straight lines and coincides with the two fault lines from [17], Example 4 (Fig. 14, middle).

Fig. 13 Reconstructed subdivision for Test problem 4.1



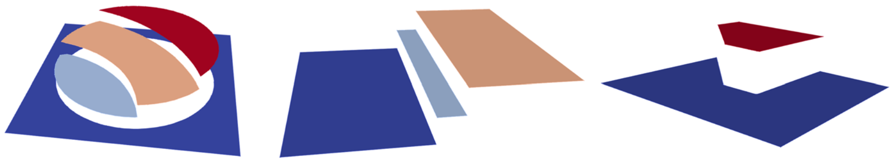


Fig. 14 Reconstructed subdivisions for Test problems 4.2, 4.3, and 4.4

Test problem 4.4 We set

$$f(x_1, x_2) = \begin{cases} 2 \ , \ x_1 > 0.4 \wedge x_2 > 0.4 \wedge x_2 < 0.2 + x_1 \\ 1 \ , \ \text{otherwise} \end{cases} \tag{15}$$

The set $\Gamma_{1,2}$ coincides with the fault line from [17], Example 5 (Fig. 14, right).

We compute all three test problems starting with X from Example 4.1 and the same parameters used there and in the subsequent examples in Sect. 2.1. Figures. 15, 16 and 17 provide an overview over the approximations of all $\Gamma_{i,j}$. We present details in Tables 1 and 2 as well as the corresponding RMS errors combined with the number of function evaluations for these three test problems and for Test problem 4.1 in Table 3. Algorithm `iniapprox` is the most demanding building block in terms of function evaluations in all examples considered. We interpret the average number of classifications per triplet as an efficiency indicator, because our specification of a polygonal approximation of a fault implies a certain number of triplets and thus of classifications, which cannot be undercut even by an optimal algorithm. For such an algorithm, the average number of classifications per triplet would be 2, because a mere verification of the triplets requires so many classifications. Averaged over Test problems 4.1, 4.3 and 4.4, it takes on average 4.0 classifications for adding one triplet in `fill`. However, it takes on average 12.2 classifications per triplet added for Test problem 4.2. This is due to the fact that $\Gamma_{1,3}$ consists of two components, which is detected by a failed attempt of filling the large gap between these two components (Remark 2.6). This process adds classifications, but no triplets. In average over the four examples, `expand` requires 5.5 classifications per triplet added. A significant part of the classifications is required for the very first and very last step, as finding them requires bisection on the extrapolation curve γ, which involves at least one classification per iteration step. Considering the same ratio for `adapt` would be misleading, as triplets are added and removed.

Test problem 4.5 For $\Omega = [0, 1]^3$, we consider the following partition: Let $x_M = (1, 0.5, 0.5)^\top$ and $\Omega_3 := \{\|x - x_M\|_6 < 0.002\} \cap \Omega$, $\Omega_1' := \{y + 0.1z < 0.7 + 0.1\sin(10x^{1.5}) + 0.05\sin(5z^{1.5})\} \cap \Omega$ and $\Omega_2' := \Omega \setminus \Omega_1'$. Then, we set $\Omega_1 := \Omega_1' \setminus \Omega_3$ and $\Omega_2 := \Omega_2' \setminus \Omega_3$ and study the partition $\Omega = \Omega_1 \cup \Omega_2 \cup \Omega_3$ (Fig. 18).

Example 4.2 We compute Test problem 4.5 starting with 200 Halton-distributed points, $k_{\text{near}} = 10$ and $k_{\text{rep}} = 15$. All other parameters are set as in the computation of Test problem 4.1. For the corresponding numbers of triplets and the number of function evaluations, we refer to Table 4. Figures. 18, 19 and 20 provide visualisations of the sets of triplets.

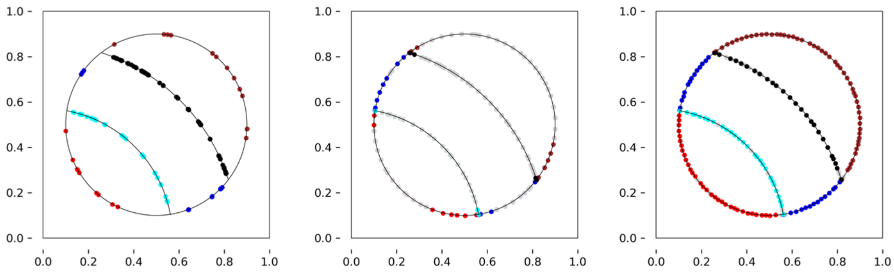


Fig. 15 $S_{i,j}^{(i)}$ (left), $\breve{S}_{i,j}^{(i)}$ (middle), and $\hat{S}_{i,j}^{(i)}$ (right) for Test problem 4.2. As for the computations of Example 4.1 in Sect. 2.1, previously existing points are greyed out

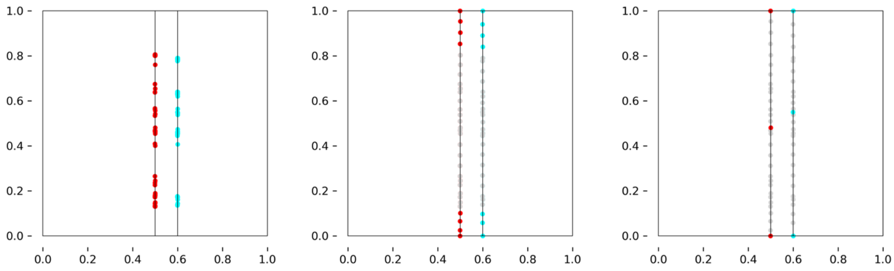


Fig. 16 $S_{i,j}^{(i)}$ (left), $\breve{S}_{i,j}^{(i)}$ (middle), and $\hat{S}_{i,j}^{(i)}$ (right) for Test problem 4.3

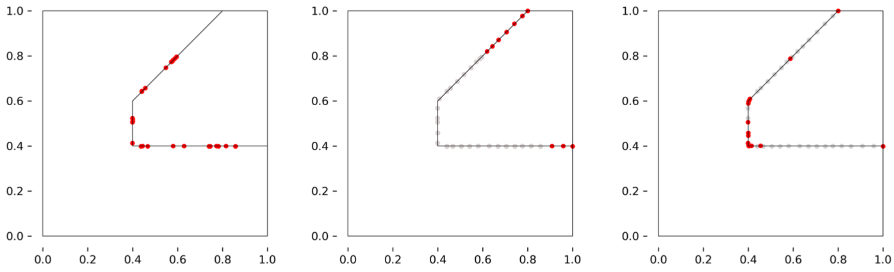


Fig. 17 $S_{1,2}^{(1)}$ (left), $\breve{S}_{1,2}^{(1)}$ (middle), and $\hat{S}_{1,2}^{(1)}$ (right) for Test problem 4.4

Table 1 Number of function evaluations per building block for reconstructing the subdomains

Test Problem	Up to $\mathcal{M}^2$	`iniapprox`	`fill`	`expand`	`adapt`
4.1	146	458	94	185	205
4.2	254	1017	465	350	127
4.3	176	595	26	74	0
4.4	87	202	34	44	83

Table 2 Total number of triplets after the respective Building block

Test Problem	`iniapprox`	`fill`	`expand`	`adapt`
4.1	40	55	89	125
4.2	66	98	138	171
4.3	32	45	60	6
4.4	17	26	36	18

Remark 4.1 (Choice of parameters) The algorithm presented relies on various parameters, whose choice may depend on the user's requirements, e.g. regarding accuracy but as well on the shape and size of the faults. The choice of the majority of the parameters is not critical in the sense that the values we presented worked well in all tests we performed during the development of the algorithm, and altering them had only minor influence on accuracy and efficiency, if at all. In what follows, we discuss the parameters whose choice is significant in practical computations and whose values may need to be adapted to the underlying classification problem. Our recommendations refer to a domain with size in the range of 1, which can be easily achieved by scaling. The initial set X must be chosen such that any Ω_i contains at least one point to get detected. This is up to the user and depends on the underlying problem such that the authors cannot give any general advice here. However, if $\Omega_i \subset \overline{X} \neq \emptyset$ by coincidence for such Ω_i, our implementation incorporates this class and proceeds as intended.

The parameter ε_{gap} must be chosen that $\overline{S}_{i,j}$ represents all features of $\Gamma_{i,j}$ sufficiently after `fill` in the sense that `adapt` initiates further local refinement if necessary. This however does not happen if a local feature of $\Gamma_{i,j}$ remains completely undetected. If $\Gamma_{i,j}$ consists of several components, ε_{gap} must be chosen smaller than their distance, as otherwise `fill` cannot detect this (compare Remark 2.6). If so, distance-based sorting of $\overline{S}_{i,j}$ using `sort` will fail subsequently. Usually, the polygonal lines based on the sorted $\overline{S}_{i,j}$ are self-intersecting in this case, which we detect in our implementation. If so, we suggest the user to decrease ε_{gap}.

If the user is interested in a polygonal approximation of $\Gamma_{i,j}$ and not just a representing set of points, it makes sense to choose $\varepsilon_{\text{err}} = \varepsilon_b$ in order to balance the associated sources of error. We suggest choosing $\varepsilon_{\text{err}} > 10 \cdot \varepsilon_{\text{coarse}}$ in order to avoid that line segments just refined in `adap` get coarsened in the next cycle of the adaptive loop and so forth.

Table 3 Total number of function evaluations and RMS error

Test Problem	$\varepsilon_b = \varepsilon_{\text{err}} = 10^{-3}$; $\varepsilon_{\text{coarse}} = 10^{-4}$		$\varepsilon_b = \varepsilon_{\text{err}} = 10^{-4}$; $\varepsilon_{\text{coarse}} = 10^{-5}$	
	no. of evals	RMS error	no. of evals	RMS error
4.1	1088	$7.66 \cdot 10^{-4}$	2406	$6.29 \cdot 10^{-5}$
4.2	2213	$3.55 \cdot 10^{-4}$	4202	$4.53 \cdot 10^{-5}$
4.3	871	$4.24 \cdot 10^{-4}$	1293	$4.02 \cdot 10^{-5}$
4.4	450	$2.75 \cdot 10^{-4}$	764	$4.26 \cdot 10^{-5}$

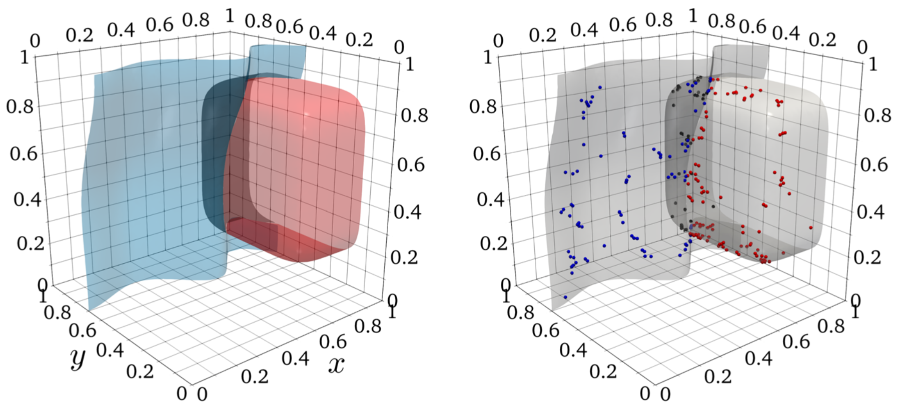


Fig. 18 Left: Partition of $[0, 1]^3$ for Test problem 4.5 displayed by $\Gamma_{1,2}$ (red), $\Gamma_{1,3}$ (blue) and $\Gamma_{2,3}$ (grey; partially obscured by $\Gamma_{1,3}$); Right: approximating sets $S_{i,j}^{(i)}$ after removing clusters

Table 4 Number of triplets per $\Gamma_{i,j}$ and number of function evaluations for Example 4.2

Number of triplets	Up to $\mathcal{M}^2$	`iniapprox`	`fill`	`expand`	`adapt`
approx. of $\Gamma_{1,2}$		102	1459	1704	2460
approx. of $\Gamma_{1,3}$		83	1685	2058	2774
approx. of $\Gamma_{2,3}$		36	745	981	1510
Function evaluations	616	2157	25,510	14,232	5218

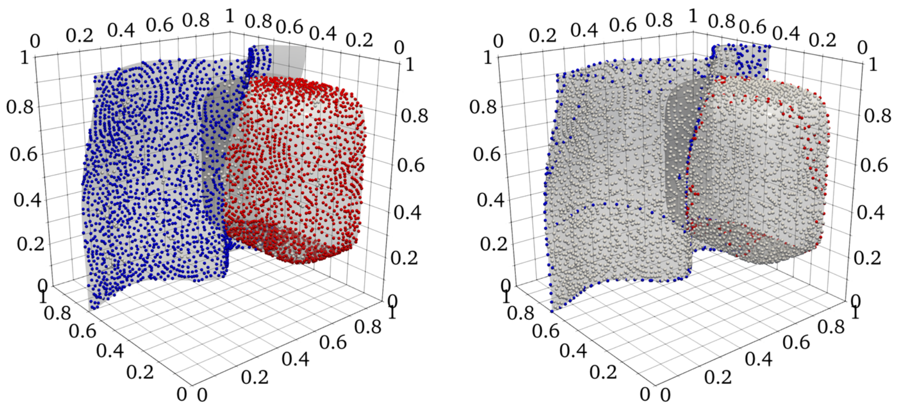


Fig. 19 Left: Sets $\overline{S}_{1,2}^{(1)}$, $\overline{S}_{1,3}^{(1)}$ obtained applying Algorithm 2.5 (`fill`) for Test problem 4.5. Right: $\check{S}_{1,2}^{(1)}$, $\check{S}_{1,3}^{(1)}$ after Algorithm 2.6 (`expand`) for the same Test problem. Previously existing points are displayed in light grey; we omit $\overline{S}_{2,3}^{(2)}$ and $\check{S}_{2,3}^{(2)}$ in the visualisation for the sake of clarity

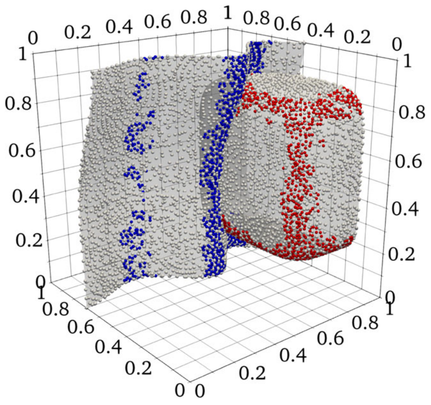


Fig. 20 Final sets $\hat{S}^{(1)}_{1,2}$ and $\hat{S}^{(1)}_{1,3}$ for Test problem 4.5

4.2 Decision analysis and modelling

MCDA methods depend on several parameters called "input factors". Almost all common MCDA models require to set up a *performance matrix* $P = (p_{i,j})$, where $p_{i,j}$ encodes the objective benefit or cost of the i-th alternative with respect to the j-th criterion. Both finding suitable criteria for modelling the decision process and setting up P requires expertise (e.g. [30] among many others) and is beyond the scope of this work. Instead, we assume that P is given and exactly known, although this assumption may be questioned in many practical applications. In addition to P, the decision maker needs to provide non-negative *weights* $w = (w_1, \ldots w_c)^\top$ which reflect the importance of the criteria from his point of view. Although originally intended as a decision support tool, MCDA methods have been used for some time for decision analysis, e.g. to predict decisions of actors under changed framework conditions, modelled by a changed performance matrix. In the context of decision analysis, w is in many practical applications not exactly known and hard to obtain, as, e.g. surveys are time-consuming and prone to bias due to socially accepted answers and other effects. Moreover, the restriction to fixed weights ignores possible diversity within actors. All of the above motivates a robustness analysis of the decision predicted, which in many cases focuses on examining the robustness of the decision with respect to perturbations in w. If P is not known exactly, one systematic approach is to consider the entries in P random variables and then to determine the distribution of the conditional probability $\mathbb{P}(i|P)$. For fixed P, it seems straightforward to model $\mathbb{P}(i|P)$ as a uniform distribution on Ω_i. For uncertain P, apart from special cases, one will have to resort to Monte Carlo methods here, within which, for a given realisation of P, the subdomains Ω_i can be approximated via the $\Gamma_{i,j}$ with our algorithm presented.

In what follows, we apply our algorithm for computing $\Gamma_{i,j}$ to such a robustness analysis and consider as example an application from the scientific monitoring of the mobility turnaround in Germany. Ball et al. [31] investigate car users' attitudes towards the purchase of hybrid (HEV) and electric vehicles (BEV) versus conventional cars with internal combustion engine (ICE). For this purpose, they identify 13 criteria and divide them into 5 categories ("Ecological","Economic", "Social", "Comfort", "Other"). The authors weight both the criteria within the categories and the categories themselves. Taking the former weights as given, we obtain the performance matrix in Table 5 from the data in [31]. In contrast to SAW (simple additive weighting) [32] as in [31], we use the more complex MCDA method SIR-TOPSIS [33] here, which includes the very widespread MCDA methods Promethee II [34] and SAW as special cases. We omit all details on how TOPSIS works for the sake of brevity and refer instead to [33].

SIR-TOPSIS requires as many other MCDA methods that the non-negative weights are normalised, i.e. $\sum_{i=1}^{n} w_i = 1$. Therefore, the set of normalised admissible weights is the standard simplex in $\mathbb{R}^n$. For visualisation, we consider $m = 3$ or $m = 4$ of the weights to be variable, and the rest to be fixed. Let be $w = w_v + w_f$, where w_v consists of the variable weights and zeros elsewhere, and w_f of the fixed ones, correspondingly. Therefore, normalisation implies $\sum w_{i,v} = 1 - \sum w_{i,f} := c_f$ such that the set of variable weights corresponds to a downscaled standard simplex in $\mathbb{R}^m$, which we embed in $\mathbb{R}^{m-1}$ by appropriate translation and rotation. This yields the equilateral triangle (for $m = 3$) and the regular tetrahedron (for $m = 4$) shown in Fig. 21.

For a 2D-visualisation of the decision space, we consider the weights corresponding to "Ecological", "Economic" and "Comfort" variable (Fig. 21) and colour all weights leading to a decision in favour of ICEs black, for HEVs blue and for BEVs green. We divide the weights $\tilde{w}$ proposed in [31] as a representation of car user's mindset in 2020 in a fixed and variable part, setting $\tilde{w} = \tilde{w}_v + \tilde{w}_f$ and display $\tilde{w}_v$ as bright red dot in Fig. 21. In contrast to [31], SIR-TOPSIS seems to predict a shift to HEVs under today's conditions. For measuring robustness, we consider the largest sphere around $\tilde{w}_v$ still fully contained in the set of weightings leading to HEVs W_{HEV} and propose its radius ρ as simple measure of robustness. As we have a polygonal approximation of W_{HEV} at hand, iteratively approximating ρ boils down to an intersection test of polygons, if we approximate the circle by a sufficiently fine polygon. We end up with $\rho \approx 0.06$, which reconciles the findings of Ball et al. and ours: As ρ is rather small, the decision in favour of HEVs is not very robust, as even small perturbations of the weightings may lead to a decision in favour of ICEs. In [31], the decision towards ICEs was found not to be very robust. However, both ours and Ball et al.'s results indicate that a broad shift towards BEVs is unlikely to happen under current circumstances.

For $m = 3$ and $m = 4$, we have $c_f = 0.9942$. The downscaled standard simplex is rotated and translated to the equilateral triangle with vertices $c_f(0.4082, -0.7071)^\top$, $c_f(0.4082, 0.7071)$, and $c_f(-0.8165, 0)$. Therefore, we set $\Omega = c_f[-0.9, 0.4082] \times$

Table 5 Performance matrix $\tilde{P}$ with corresponding criteria generated from [31], and weightings from the car users' point of view prior to normalisation (last row)

	Ecological	Economic	Social	Comfort	Other
BEV	0.5025	0.2792	0.6250	0.1497	0.1342
ICE	0.1256	0.4167	0.1250	0.4300	0.6710
HEV	0.3719	0.3042	0.2500	0.4202	0.1948
weightings	2	7	0.1	8	0

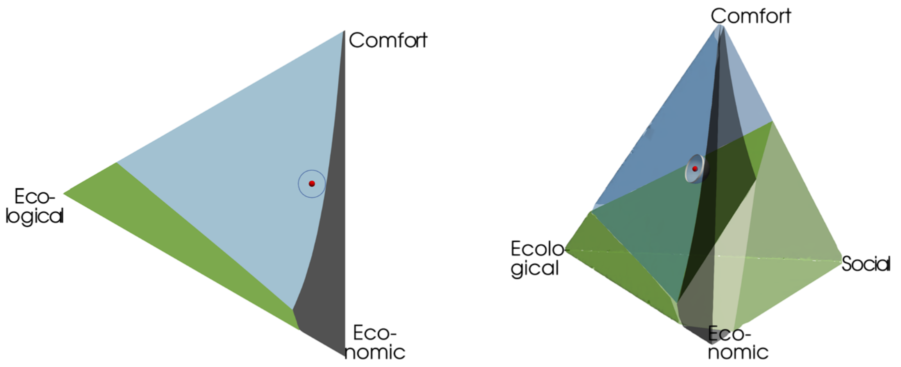


Fig. 21 Visualisation of the decision space for 3 (left) and 4 (right) variable criteria. The car users' weightings according to [31] are displayed as red dot along with the circumsphere with radius $\rho = 0.06$

$c_f[-0.9, 0.9]$ and employ an initial point set X consisting of 100 Halton-distributed points, where points far away from the triangle have been discarded, as they cannot aid approximating $\Gamma_{i,j}$ (Fig. 22, left). We choose the values of all algorithm-related parameters as for the examples in Sects. 2.1 and 4.1, except of $\varepsilon_{\text{gap}} = c_f \cdot 0.05$. For $m = 4$, we start with X consisting of 500 Halton-distributed points in the vicinity of the tetrahedron and take all values for the parameters of the algorithm from Sect. 2.2, except of $k_{\text{adap}} = 3$. For the final sets $\hat{S}_{i,j}$, we refer to Fig. 22 (right and middle) and display the number of function evaluations in Table 6. These sets have been used for approximating ρ; Fig. 21 was generated with the proposed algorithm, albeit using a finer initial point set and modified parameter settings, as analytical descriptions of the decision curves and surfaces, resp., are unknown.

Due to recent geopolitical events, car users today may attach a different importance to issues of security of supply, e.g. with fuel, than in 2021 when [31] was written. That kind of considerations are subsumed in the category "Social" which motivates to additionally consider the weights of that category to be variable. It turns out that with stronger emphasis on the category "Social", car users tend to prefer a BEV (Fig. 21, right). One reason for this may be that the dependence on oil imports makes the purchase of an ICE or even an HEV seem less attractive.

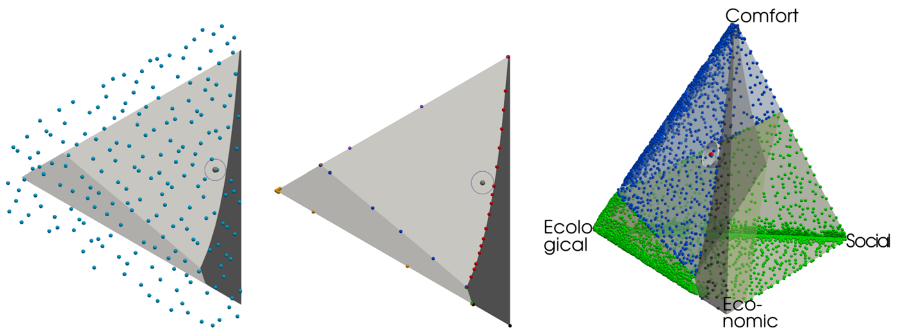


Fig. 22 Initial point set X (left) and final sets $\hat{S}_{i,j}$ (middle) for $m = 3$. Right: Selected final sets $\hat{S}_{i,j}$ for $m = 4$

Table 6 Number of function evaluations per building block for reconstructing the decision boundaries (see Sect. 4.2)

	Up to $\mathcal{M}^2$	`Iniapprox`	`Fill`	`Expand`	`Adapt`
Evaluations, $m = 3$	397	1325	198	283	44
Evaluations, $m = 4$	1518	4553	13352	14752	24344
No. of triplets, $m = 3$		100	145	174	45
No. of triplets, $m = 4$		335	1575	2401	6999

4.3 Surface reconstruction in 3D from scattering

In our tests, we will use different scatterers to apply our algorithm to such as the unit sphere, the ellipsoid, the peanut, the acorn, the cushion, the round short cylinder and the round long cylinder. Their surfaces are given parametrically in spherical coordinates $x = r_1 \sin(\phi)\cos(\theta)$, $y = r_2 \sin(\phi)\cos(\theta)$, and $z = r_3 \cos(\phi)$ with $\theta \in [0, 2\pi)$, $\theta \in [0, \pi]$ as $r_1 = r_2 = r_3 = 1$ for the unit sphere, $r_1 = r_2 = 1$ and $r_3 = 6/5$ for the ellipsoid, $r_1^2 = r_2^2 = r_3^2 = 9(\cos^2(\phi) + \sin^2(\phi)/4)/4$ for the peanut, $r_1^2 = r_2^2 = r_3^2 = 9(17/4 + 2\cos(3\phi))/25$ for the acorn, $r_1 = r_2 = r_3 = 1 - \cos(2\phi)/2$ for the cushion, $r_1^{10} = r_2^{10} = r_3^{10} = 1/((2\sin(\phi)/3)^{10} + \cos^{10}(\phi))$ for the round short cylinder, and $r_1^{10} = r_2^{10} = r_3^{10} = 1/((2\cos(\phi)/3)^{10} + \sin^{10}(\phi))$ for the round long cylinder, respectively. We will use $m = 1026$ number of incident and observation directions for the construction of the far-field data with the parameters $k_e = 2$, $k_i = 1$, and $\tau = 1/2$ as also used in [10, p. 18]. Therefore, the factorization algorithm appears as classification function f.

For our experiments, we set all algorithm-related parameters as in Sect. 2.2 except of $\varepsilon_{\text{err}} = 0.01$ and $\varepsilon_{\text{gap}} = 0.25$. The initial set X consists of 200 Halton-distributed points in $\Omega = [-1.5, 1.5]^3$ apart of the long and the short cylinder, where we set $\Omega = [-2, 2]^3$. The results (Figs. 23 and 24) indicate successful reconstructions. For the number of triplets after each part of the algorithm and the number of function evaluations, we refer to Tables 7 and 8. As no triplet in these tests fulfilled condition (6), no expansion took place, such that we omit the corresponding column in our tables. As all scatters feature one closed surface, this behaviour of our algorithm is as desired. In order to obtain a visually appealing reconstruction of the scatterers, Anagnostopoulos et al. [10] create a tensor product set of $55^3 = 166,375$ points and use the corresponding classifications to compute isosurfaces based upon these data for

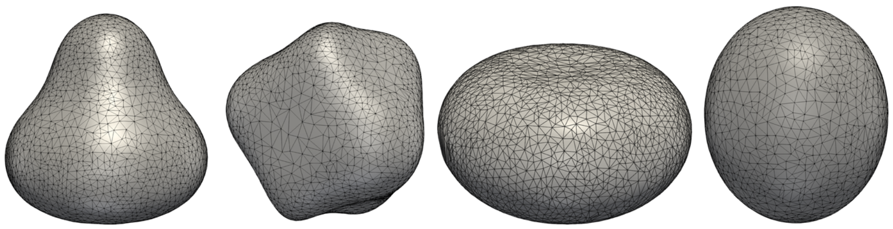


Fig. 23 Reconstructed scatterers acorn, bumpy sphere, cushion and ellipsoid

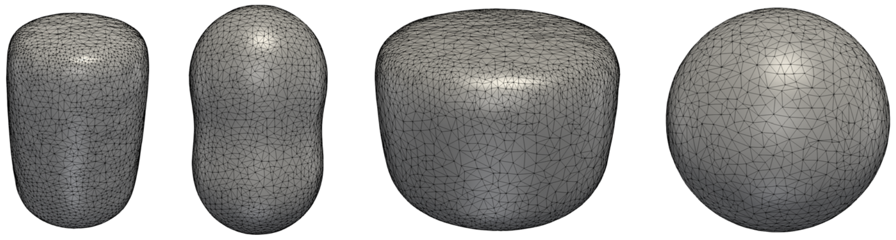


Fig. 24 Reconstructed Scatterers long cylinder, peanut, short cylinder and sphere

visualisation purposes. Our approach, on the other hand, requires only a fraction of these function evaluations. Since about half of the total computation time was used by Kirsch's factorization method in the surface reconstruction with a Matlab implementation of our method, our algorithm enables a significant speedup compared to [10].

Moreover, in contrast to the level set approach, our method provides a set of points near the scatterer, which can be used for computing a further refined surface representation like triangulation or higher-order interpolation. There are two major sources of inaccuracy in surface reconstruction: the inaccuracy of the factorization method itself and the error induced by the representation of the reconstructed surface, be it an isosurface or a set of points. As our algorithm controls the latter one, comparing with the ground truth aka the analytical description of the true surface allows for analysing the former error source far more easily than having only implicit surfaces available.

Remark 4.2 In our tests, we do not commit to inverse crime. The far field data used for reconstruction later one have been produced using boundary element collocation with high accuracy. The inverse problem is solved using Kirsch's factorization method combined with the algorithm proposed in this work.

5 Conclusions and outlook

In this article, we presented a method for approximating manifolds of discontinuity of a function f in 2D and 3D and demonstrated successful applications of our method to multi-criteria decision aid, to generic test cases connected with the detection of

Table 7 Number of function evaluations per building block for reconstructing the scatterers

Scatterer	Up to $\mathcal{M}^2$	`iniapprox`	`fill`	`adapt`
Acorn	463	1891	7242	8272
Bumpy sphere	384	1326	4844	8911
Cushion	473	1956	6237	7881
Ellipsoid	383	1329	4694	5458
Long cyl	378	1377	7119	9816
Peanut	394	1381	5149	6922
Short cyl	465	2033	9021	9667
Sphere	384	1314	3954	5102

Table 8 Total number of triplets after the respective building block for reconstructing the scatterers

Scatterer	`iniapprox`	`fill`	`adapt`
Acorn	222	1291	3839
Bumpy sphere	158	827	3103
Cushion	232	1125	3613
Ellipsoid	157	843	2305
Long cyl	149	1188	4040
Peanut	164	878	2864
Short cyl	217	1594	4543
Sphere	159	715	2120

faults and to an inverse acoustic scattering problem. In all cases, our method requires significantly fewer evaluations of f than previously existing algorithms we compared our method to. At least for the inverse acoustic scattering problem, this leads to a significant acceleration of the overall reconstruction, as computing the factorization method dominates the computational time even in our computations, where the number of such computations could be reduced by a factor of approx. 7 to 12.

Our algorithm could be easily enhanced with classification algorithms as in [17] in order to consider more general functions than we did, provided the classification is certain. This enables for tackling fault detection problems and for applying our method for interpolation of piecewise smooth functions assuming that f is smooth on $\Omega_1, \ldots, \Omega_n$ with $\Omega = \overline{\Omega_1} \cup \ldots \cup \overline{\Omega_n}$, but globally discontinuous. In [35], the authors propose interpolation with radial basis functions on each Ω_i in this setting. This however requires knowledge about the boundaries of each Ω_i which could be obtained using our method. For that purpose, [35] provides a fault detection algorithm based on local approximation properties. Combining these two approaches is the subject of our current research.

Appendix

Algorithm 1 Pseudo-code for Building block `iniapprox`

```
function INIAPPROX($\mathcal{M}_i$, $\overline{X}$, $j$, $\varepsilon_b$)
    ▷ $\mathcal{M}_i = (\mathcal{M} \cup \mathcal{M}^2) \cap \Omega_i$
    ▷ $\overline{X} = X \cup \mathcal{M} \cup \mathcal{M}^2$: enriched set of sampling points
    ▷ $j$: class index, $j > i$
    ▷ $\varepsilon_b$: threshold for distance to $\Gamma_{i,j}$
    $\tilde{S}_{i,j}^{(i)} = \emptyset$, $\tilde{S}_{i,j}^{(j)} = \emptyset$
    for all $x \in X_i$ do
        $x'$ = nearestNeighbour($x$, $\overline{X} \cap \Omega_j$)          ▷ nearest neighbour of $x$ in $\Omega_j$
        ($x^{(i)}, x^{(j)}$), success = bisection($x, x', 2\varepsilon_b$)
        if success then
            $\tilde{S}_{i,j}^{(i)} = \tilde{S}_{i,j}^{(i)} \cup \{x^{(i)}\}$
            $\tilde{S}_{i,j}^{(j)} = \tilde{S}_{i,j}^{(j)} \cup \{x^{(j)}\}$
        end if
    end for
    return $\tilde{S}_{i,j}^{(i)}$, $\tilde{S}_{i,j}^{(j)}$
end function
```

Algorithm 2 Pseudo code for Building block 2.2 (`estcurv`)

```
function ESTCURV($S_{\text{loc}} = \{x_1, \dots, x_r\}, \ell, \varepsilon_b$)
    ▷ $S_{\text{loc}}$: ordered set of points
    ▷ $\ell$: index of the point in $S_{\text{loc}}$ to estimate the curvature at
    ▷ $\varepsilon_b$: maximal distance of a point in $S_{\text{loc}}$ to $\Gamma$
    $S_{\text{loc}} = S_{\text{loc}} - x_\ell$                                    ▷ center $S_{\text{loc}}$ around $x_\ell$

    ▷ rotate $S_{\text{loc}}$ by a PCA such that the $x$-axis corresponds to the first coordinates of $S_{\text{loc}}$
    $S_{\text{loc}} = Q \cdot S_{\text{loc}}$

    ▷ interior points of sufficiently large $S_{\text{loc}}$ with ordered $x$-coordinates
    if $(r > 3) \wedge (1 < k < r) \wedge (\forall k : x_{k,1} < x_{k+1,1} \vee \forall k : x_{k,1} < x_{k+1,1}$ then
        $s = \frac{\|x_1 - x_r\|}{r}$                          ▷ scaling factor for RBF approximation
        $A_{i,j} = \Phi(||x_i - x_j||, s)$                     ▷ RBF interpolation matrix, $\Phi$: Gaussian RBF
        $B_{i,j} = \ddot{\Phi}(\|x_i - x_j\|, s)$               ▷ penalty matrix
        rescale $B$ such that $\|B^\top B\| \approx 10$         ▷ for numerical stability

        ▷ solve interpolation problem regularised with $B$ using Tikhonov regularisation
        $\alpha = \text{Tikhonov}(A, B, \varepsilon_b)$
        compute $\varphi, \dot{\varphi}, \ddot{\varphi}$         ▷ RBF approximation with derivatives
        $c_\ell = \frac{\dot{\varphi}(x_\ell)}{\left(1+\ddot{\varphi}(x_\ell)^2\right)^{1.5}}$
    else                                     ▷ fallback to drawing a circle through 3 consecutive points
        if $\ell = 1$ then
            ▷ $\rho$: radius of circle through $x_1, x_2, x_3$
            $\rho$ = computeRadius($x_1, x_2, x_3$)
        else if $\ell = r$ then
            $\rho$ = computeRadius($x_{r-2}, x_{r-1}, x_r$)
        else
            $\rho$ = computeRadius($x_{\ell-1}, x_\ell, x_{\ell+1}$)
        end if
        $c_\ell = 1/\rho$
    end if
    return $c_\ell$                                              ▷ $c_\ell$: estimated curvature
end function
```

Algorithm 3 Pseudo-code for Building block 2.4 (`startpairs`), Part 1

```
function STARTPAIRS($x^+, x^-, c^+, c^-, i, j, k_{\text{rep}}$)
    ▷ $x^+, x^-$: two points near $\Gamma_{i,j}$
    ▷ $c^+, c^-$: class indices of $x^+, x^-$
    ▷ $i, j$: class indices of a valid start pair
    ▷ $k_{\text{rep}}$: maximal number of escalation steps
    success = false                                    ▷ flag for successful termination
    $t$ = false                                        ▷ flag for third class involved
    $(x_0^+, x_0^-) = (x^+, x^-)$
    $k = 1$
    if $\{c^+, c^-\} \subseteq \{i, j\}$ then                       ▷ no third class involved
        for $k \leq k_{\text{rep}}$ do
            if $\{c^+, c^-\} \subseteq \{i, j\} \wedge c^+ \neq c^-$ then
                success = true
                exit loop
            else if $\{c^+, c^-\} \subseteq \{i, j\}$ then
                $x^+ = 2x^- - x^+$                               ▷ reflection step for $x^+$
                $c^+ = \text{classify}(x^+)$
            else
                success = false
                $t$ = true
                exit loop
            end if

            ▷ Reflecting $x^+$ did not succeed, so we reflect $x^-$.
            if $\{c^+, c^-\} \subseteq \{i, j\} \wedge c^+ \neq c^-$ then
                success = true
                exit loop
            else if $\{c^+, c^-\} \subseteq \{i, j\}$ then
                $x^- = 3x^+ - 2x^-$                              ▷ reflection step for $x^-$
                $c^- = \text{classify}(x^-)$
            else
                success = false
                $t$ = true
                exit loop
            end if
            $k = k + 1$
        end for
    else
        $t$ = true
    end if
```

Algorithm 3 Pseudo-code for Building block 2.4 (`startpairs`), Part 2

```
    ▷ fallback, if a third class is involved
    if $t \wedge$ !success then
        $n = x_0^+ - x_0^-$
        compute $X$ by scattering around $0.5(x_0^+ + x_0^-)$ in direction of $n$
        classify $X$
        if $\exists x^+, x^- \in X : \{c^+, c^-\} \subseteq \{i, j\} \wedge c^+ \neq c^-$ then
            success = true
            $t$ = false
        end if
    end if
    return $(x^+, x^-)$, success                       ▷ starting pair with success flag
end function
```

Algorithm 4 Pseudo-code for Algorithm 2.2 (`fill`), $2d$, Part 1

```
function FILL($S_{i,j}^{(i)}, S_{i,j}^{(j)}, i, j, \varepsilon_{\text{gap}}, k_{\text{rep}}$)
    ▷ $S_{i,j}^{(i)}, S_{i,j}^{(j)}$: points inside the triples in $S_{i,j}$
    ▷ $i, j$: class indices
    ▷ $\varepsilon_{\text{gap}}$: maximal distance between subsequent triplets; user-defined
    ▷ $k_{\text{rep}}$: maximal number of filling cycles; user-defined, usually set to 3
    Nogaps = false
    $k = 1$
    $m = |S_{i,j}^{(i)}|$                                    ▷ number of triplets in $S_{i,j}$
    while !Nogaps $\wedge$ $k \leq k_{\text{rep}}$ do
        Nogaps = true
        for $\ell = 1 : m - 1$ do
            if $d_\ell = \|x_\ell - x_{\ell+1}\| > \varepsilon_{\text{gap}}$ then
                NoGaps = false
                $R = \lceil d_\ell / \varepsilon_{\text{gap}} \rceil$
                for $r = 1 : R$ do                    ▷ $R$ equidistant points on $\overline{x_\ell^{(i,j)} x_{\ell+1}^{(i,j)}}$
                    $x_{\ell,r} = (1 - r/R) x_\ell^{(i,j)} + r/R x_{\ell+1}^{(i,j)}$

                    ▷ outer normal vector with respect to $\Omega_i$
                    $n = \texttt{normalVec}(x_\ell^{(i,j)}, x_{\ell+1}^{(i,j)})$

                    ▷ employ Building block 2.2 (estcurv)
                    ▷ As estimating curvature is less reliable at the end of a curve, we apply a safety factor
                    if $2 < \ell < |S_{i,j}^{(i)}| - 2$ then
                        $c = \texttt{estcurv}(\{x_{\ell-2}^{(i,j)}, \ldots, x_{\ell+2}^{(i,j)}\}, 3, \varepsilon_b)$
                    else if $\ell \leq 2$ then
                        $c = 2.0\ \texttt{estcurv}(\{x_1^{(i,j)}, \ldots, x_{\min\{5,m\}}^{(i,j)}\}, 2, \varepsilon_b)$
                    else
                        $c = 2.0\ \texttt{estcurv}(\{x_{\max\{m-5,1\}}^{(i,j)}, \ldots, x_m^{(i,j)}\}, m - 1, \varepsilon_b)$
                    end if
                    $\delta = \texttt{esterror}(c, d_\ell)$                    ▷ estimate error using (1)
                    $\alpha = \min\{\varepsilon_{\text{safemax}} d_\ell, \max\{\delta, \varepsilon_{\text{safemin}} \varepsilon_b\}\}$     ▷ compare (2)
                    $x_{\ell,r}^+ = x_{\ell,r} + \alpha n$, $x_{\ell,r}^- = x_{\ell,r} - \alpha n$
                    $c^+, c^- = \text{classify}(x_{\ell,r}^+, x_{\ell,r}^-)$

                    ▷ employ Building block 2.4 (startpairs)
                    $(x_{\ell,r}^+, x_{\ell,r}^-, \text{success}) = \texttt{startpairs}(x_{\ell,r}^+, x_{\ell,r}^-, c^+, c^-, i, j, k_{\text{rep}})$
```

Algorithm 4 Pseudo-code for Algorithm 2.2 (`fill`), $2d$, Part 2

```
                    if success then
                        $(x_{\ell,r}^+, x_{\ell,r}^-)$, successBisec = $\texttt{bisection}(x_{\ell,r}^+, x_{\ell,r}^-, 2\varepsilon_b)$
                        if successBisec then
                            $S_{i,j}^{(i)} = S_{i,j}^{(i)} \cup \{x_{\ell,r}^-\}$
                            $S_{i,j}^{(j)} = S_{i,j}^{(j)} \cup \{x_{\ell,r}^+\}$
                        end if
                    end if
                end for
            end if
        end for
        $S_{i,j} = \texttt{sort}(S_{i,j})$                    ▷ Building block 2.1 (sort)
        $k = k + 1$
    end while
    $\overline{S}_{i,j} = S_{i,j}$
    return $\overline{S}_{i,j}$, NoGaps        ▷ !NoGaps indicates that $\Gamma_{i,j}$ consists of several components
end function
```

Algorithm 5 Pseudo-code for Algorithm 2.5 (`fill`), 3d, Part 1

```
function FILL($S_{i,j}^{(i)}, S_{i,j}^{(j)}, i, j, \varepsilon_{\text{gap}}, k_{\text{rep}}, k_{\text{near}}$)
    ▷ $S_{i,j}^{(i)}, S_{i,j}^{(j)}$: points in the triples in $S_{i,j}$
    ▷ $i, j$: class indices
    ▷ $\varepsilon_{\text{gap}}$: parameter for minimal density of triplets, user-defined
    ▷ $k_{\text{rep}}$: maximal number of filling cycles; user-defined
    NoGaps = false
    $k = 1$
    $m = |S_{i,j}^{(i)}|$                                              ▷ number of triplets in $S_{i,j}$
    $S_{\text{new}} = \emptyset$, $N_{\text{new}} = \emptyset$, $E_{new} = \emptyset$
    while !NoGaps $\wedge k \leq k_{\text{rep}}$ do
        NoGaps = true
        for $x \in S_{i,j}$ do
            ▷ the $k_{\text{near}}$ nearest neighbours of $x$ in $X$
            $S_{\text{loc}} = \{x, x_1, \ldots, x_{k_{\text{near}}}\}$
            $x_{\text{mid}}$ = barycentre($S_{\text{loc}}$)                         ▷ compute barycentre of $S_{\text{loc}}$
            $S_{\text{loc}}^{\text{flat}} = Q(\cdot S_{\text{loc}} - x_{\text{mid}})$                      ▷ $Q$ obtained by a PCA of $S_{\text{loc}}$
            vals = $S_{\text{loc}}^{\text{flat}}[:, 3]$                             ▷ $z$-component of local coordinates

            ▷ $x$- and $y$-component of local coordinates
            points = $S_{\text{loc}}^{\text{flat}}[:, 1:2]$
            $x^{\text{flat}} = (Q \cdot (x - x_{\text{mid}}))[1:2]$                      ▷ $x$ in local coordinates
            $x_{\min}$ = min points [:, 1], $x_{\max}$ = maxpoints[:, 1]

            ▷ detect large gaps, default $\alpha = 0.1$
            if $(x^{\text{flat}}[1] < x_{\min} + \alpha(x_{\max} - x_{\min})) \vee x^{\text{flat}}[1] > x_{\min} + (1-\alpha)(x_{\max} - x_{\min}))$ then
                ▷ $x_{\text{mid}} = 0$ in local coordinates
                $x_{\text{new}}^{\text{flat}} = (1 + 1.5\varepsilon_{\text{gap}}/\|x - x_{\text{mid}}\|)x^{\text{flat}}$
                $S_{\text{loc}}^{\text{flat}} = S_{\text{loc}}^{\text{flat}} \cup \{x_{\text{new}}^{\text{flat}}\}$
            end if
            trias = delaunay($S_{\text{loc}}^{\text{flat}}$)                       ▷ compute Delaunay triangulation
            for $t \in$ trias do
                compute maximal edge length $l_{\max}$ of $t$
                if $l_{\max} > \varepsilon_{\text{gap}} \wedge t$ is not too anisotropic then
                    centre = barycentre($t$)

                    ▷ scaling factor for RBF approximation
                    $s = 2\sqrt{(x_{\max} - x_{\min})(\max \text{points}[:,2] - \min \text{points}[:,2])/|S_{\text{loc}}^{\text{flat}}|}$

                    ▷ Tikhonov-regularised RBF approximation using Gaussians as in Building block 2.2
(estcurv)
                    $\varphi$ = RBFApprox(points, vals, $s$, $\varepsilon_b$)
```

Algorithm 5 Pseudo-code for Algorithm 2.5 (`fill`), 3d, Part 2

```
                ▷ centre of triangle in global coordinates
                centre = [centre, $\varphi$(centre)]$\cdot Q' + x_{\mathrm{mid}}$
                estimate outer normal vector $n$ in centre
                $E$ = esterror($\varphi$, tria, centre)                ▷ Building block 2.8
                $S_{\mathrm{new}} = S_{\mathrm{new}} \cup \{\mathrm{centre}\}$
                $N_{\mathrm{new}} = N_{\mathrm{new}} \cup \{n\}$
                $E_{\mathrm{new}} = E_{\mathrm{new}} \cup \{E\}$
            end if
        end for
    end for

    if $S_{\mathrm{new}} = \emptyset$ then
        NoGaps = true
    else
        ▷ remove clusters in $S_{\mathrm{new}}$ and eliminate corresponding entries in $N_{\mathrm{new}}$, $E_{\mathrm{new}}$
        $S_{\mathrm{new}}, N_{\mathrm{new}}, E_{\mathrm{new}}$ = removeclusters($S_{\mathrm{new}}, N_{\mathrm{new}}, E_{\mathrm{new}}$)
        for $i = 1 : |S_{\mathrm{new}}|$ do
            compute $\alpha$ from $E_{\mathrm{new}}[i]$ by (2)
            $x^+ = S_{\mathrm{new}}[i] + \alpha N_{\mathrm{new}}[i],\ x^- = S_{\mathrm{new}}[i] - \alpha N_{\mathrm{new}}[i]$
            $c^+, c^-$ = classify($x^+, x^-$)

            ▷ employ building block 2.4 (startpairs)
            ($x^+, x^-$, success) = startpairs($x^+, x^-, c^+, c^-, i, j, \tilde{k}_{\mathrm{rep}}$)
            if success then
                ($x^+_{\ell,r}, x^-_{\ell,r}$), successBisec = bisection($x^+_{\ell,r}, x^-_{\ell,r}, 2\varepsilon_b$)
                if successBisec then
                    $S^{(i)}_{i,j} = S^{(i)}_{i,j} \cup \{x_-\}$
                    $S^{(j)}_{i,j} = S^{(j)}_{i,j} \cup \{x_+\}$
                end if
            end if
        end for
        $S^{(i)}_{i,j}, S^{(j)}_{i,j}$ = removeclusters($S^{(i)}_{i,j}, S^{(j)}_{i,j}$)
    end if
  end while
  $\overline{S}_{i,j} = S_{i,j}$
  return $\overline{S}_{i,j}$, NoGaps
end function
```

Acknowledgements The authors want to thank Rodin Eybesh and Luis Hasenauer for porting a significant part of our Matlab implementation to python and the two anonymous reviewers for their valuable comments which helped to improve the paper.

Author Contributions Matthias Grajewski developed the algorithm, implemented it and wrote the main manuscript text except of Section 3. This was written by Andreas Kleefeld who proofread the entire manuscript and contributed an implementation of Kirsch's factorization method along with the test cases regarding inverse scattering in Section 4.3. All authors reviewed the manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data availability The codes used for producing the results presented are available at https://github.com/mgrajewski/faultapprox-matlab and https://github.com/mgrajewski/faultapprox-python.

Declarations

Ethical approval Not applicable

Competing interests The authors declare no competing interests.

References

1. Figueira, J.: Multiple criteria decision analysis: state of the art surveys. SpringerLink Bücher, vol. 78. Springer, New York (2005). https://doi.org/10.1007/b100605
2. Papathanasiou, J., Nikolaos, P.: Multiple criteria decision aid: methods, examples and Python implementations. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-91648-4
3. Kleefeld, A., Lin, T.-C.: The nonlinear Landweber method applied to an inverse scattering problem for sound-soft obstacles in 3D. Comput. Phys. Commun. **182**(12), 2550–2560 (2011). https://doi.org/10.1016/j.cpc.2011.07.023
4. Zeng, F., Suarez, P., Sun, J.: A decomposition method for an interior inverse scattering problem. Inverse Probl. Imaging **7**(1), 291–303 (2013). https://doi.org/10.3934/ipi.2013.7.291
5. Potthast, R.: A survey on sampling and probe methods for inverse problems. Inverse Probl. **22**(2), 1 (2006). https://doi.org/10.1088/0266-5611/22/2/R01
6. Colton, D., Kirsch, A.: A simple method for solving inverse scattering problems in the resonance region. Inverse Probl. **12**(4), 383–393 (1996). https://doi.org/10.1088/0266-5611/12/4/003
7. Audibert, L., Haddar, H.: A generalized formulation of the linear sampling method with exact characterization of targets in terms of farfield measurements. Inverse Probl. **30**(3), 035011 (2014). https://doi.org/10.1088/0266-5611/30/3/035011
8. Kirsch, A., Grinberg, N.: The factorization method for inverse problems. Oxford University Press, Oxford (2008). https://doi.org/10.1093/acprof:oso/9780199213535.001.0001
9. Ikehata, M.: The probe method and its applications. In: Nakamura, G., Saitoh, S., Seo, J., Yamamoto, M. (eds.) Inverse problems and related topics. Research Notes in Mathematics, vol. 419. CRC Press, London (2000). https://doi.org/10.1201/9780429187841-4
10. Anagnostopoulos, K.A., Charalambopoulos, A., Kleefeld, A.: The factorization method for the acoustic transmission problem. Inverse Probl. **29**(11), 115015 (2013). https://doi.org/10.1088/0266-5611/29/11/115015
11. Bazán, F.S.V., Kleefeld, A., Leem, K.H., Pelekanos, G.: Sampling method based projection approach for the reconstruction of 3D acoustically penetrable scatterers. Linear Algebra Appl. **495**, 289–323 (2016). https://doi.org/10.1016/j.laa.2015.12.020
12. Harris, I., Kleefeld, A.: Analysis of new direct sampling indicators for farfield measurements. Inverse Probl. **35**(5), 054002 (2019). https://doi.org/10.1088/1361-6420/ab08be
13. Gutzmer, T., Iske, A.: Detection of discontinuities in scattered data approximation. Numer. Algorithms **16**, 155–170 (1997). https://doi.org/10.1023/A:1019139130423
14. Gout, C., Le Guyader, C., Romani, L., Saint-Guirons, A.-G.: Approximation of surfaces with fault(s) and/or rapidly varying data, using a segmentation process, Dm-splines and the finite element method. Numer. Algorithms **48**, 67–92 (2008). https://doi.org/10.1007/s11075-008-9177-8
15. Bozzini, M., Rossini, M.: The detection and recovery of discontinuity curves from scattered data. J. Comput. Appl. Math. **240**, 148–162 (2013). https://doi.org/10.1016/j.cam.2012.06.014

16. Arge, E., Floater, M.: Approximating scattered data with discontinuities. Numer. Algorithms **8**, 149–166 (1994). https://doi.org/10.1007/BF02142688
17. Allasia, G., Besenghi, R., Cavoretto, R., De Rossi, A.: Efficient approximation algorithms. Part I: approximation of unknown fault lines from scattered data. Dolomites Res. Notes Approx. 3, 7–38 (2010). https://doi.org/10.14658/pupj-drna-2010-1-2
18. Allasia, G., Besenghi, R., Cavoretto, R.: Adaptive detection and approximation of unknown surface discontinuities from scattered data. Simul. Model. Pract. Theory **17**(6), 1059–1070 (2009). https://doi.org/10.1016/j.simpat.2009.03.007
19. Wendland, H.: Scattered data approximation. Cambridge monographs on applied and computational mathematics, vol. 17. Cambridge University Press, Cambridge and New York (2010). https://doi.org/10.1017/CBO9780511617539.016
20. Cavoretto, R.: Adaptive radial basis function partition of unity interpolation: a bivariate algorithm for unstructured data. J. Sci. Comput. 87(41), (2021). https://doi.org/10.1007/s10915-021-01432-z
21. Mirzaei, D., Soodbakhsh, N.: A fault detection method based on partition of unity and kernel approximation. Numer. Algorithms (2023). https://doi.org/10.1007/s11075-022-01488-4
22. Bishop, C.M.: Pattern recognition and machine learning. Springer, New York (2006)
23. Althaus, E., Mehlhorn, K.: Traveling salesman-based curve reconstruction in polynomial time. SIAM J. Comput. **31**(1), 27–66 (2001). https://doi.org/10.1137/S0097539700366115
24. Amenta, N., Bern, M., Eppstein, D.: The crust and the ß-skeleton: combinatorial curve reconstruction. Graphic. Models Image Process. **60**(2), 125–135 (1998). https://doi.org/10.1006/gmip.1998.0465
25. Dey, T.K., Mehlhorn, K., Ramos, E.A.: Curve reconstruction: connecting dots with good reason. Comput. Geom. **15**(4), 229–244 (2000). https://doi.org/10.1016/s0925-7721(99)00051-6
26. Ohrhallinger, S., Mudur, S.: An efficient algorithm for determining an aesthetic shape connecting unorganized 2D points. Comput. Graph. Forum **32**(8), 72–88 (2013). https://doi.org/10.1111/cgf.12162
27. Waldron, S.: The error in linear interpolation at the vertices of a simplex. SIAM J. Numer. Anal. **35**(3), 1191–1200 (1998). https://doi.org/10.1137/S0036142996313154
28. Shakarji, C.M.: Least-squares fitting algorithms of the NIST algorithm testing system. J. Res. Natl. Inst. Stand. Technol. **103**(6), 633–641 (1998)
29. Kleefeld, A.: The transmission problem for the Helmholtz equation in $\mathbb{R}^3$. Comput. Methods Appl. Math. **12**(3), 330–350 (2012). https://doi.org/10.2478/cmam-2012-00088
30. Vögele, S., Ball, C., Kuckshinrichs, W.: Multi-criteria approaches to ancillary effects: the example of E-mobility. In: Buchholz, W., Markandya, A., Rübbelke, D., Vögele, S. (eds.) Ancillary Benefits of Climate Policy: New Theoretical Developments and Empirical Findings, pp. 157–178. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-30978-7_9
31. Ball, C.S., Vögele, S., Grajewski, M., Kuckshinrichs, W.: E-mobility from a multi-actor point of view: uncertainties and their impacts. Technol. Forecast. Soc. Change **170**, 120925 (2021). https://doi.org/10.1016/j.techfore.2021.120925
32. Churchman, C.W., Ackoff, R.L.: An approximate measure of value. J. Oper. Res. Soc. Am. **2**(2), 172–187 (1954). https://doi.org/10.1287/opre.2.2.172
33. Xu, X.: The SIR method: a superiority and inferiority ranking method for multiple criteria decision making. European J. Oper. Res. **131**(3), 587–602 (2001). https://doi.org/10.1016/S0377-2217(00)00101-6
34. Brans, J.P., Vincke, Ph., Mareschal, B.: How to select and how to rank projects: the PROMETHEE method. European J. Oper. Res. **24**, 228–238 (1986). https://doi.org/10.1016/0377-2217(86)90044-5
35. Lenarduzzi, L., Schaback, R.: Kernel-based adaptive approximation of functions with discontinuities. Appl. Math. Comput. **307**, 113–123 (2017). https://doi.org/10.1016/j.amc.2017.02.043