Numerical Algorithms (2023) 93:1759–1794
https://doi.org/10.1007/s11075-022-01488-4

ORIGINAL PAPER

A fault detection method based on partition of unity and kernel approximation

Davoud Mirzaei[1,2] · Navid Soodbakhsh[1]

Received: 24 January 2022 / Accepted: 8 December 2022 / Published online: 18 January 2023

Abstract
In this paper, we present a scattered data approximation method for detecting and approximating the discontinuities of a bivariate function and its gradient. The new algorithm is based on partition of unity, polyharmonic kernel interpolation, and principal component analysis. Localized polyharmonic interpolation in partition of unity setting is applied for detecting a set of fault points on or close to discontinuity curves. Then a combination of partition of unity and principal component regression is used to thinning the detected points by moving them approximately on the fault curves. Finally, an ordered subset of these narrowed points is extracted and a parametric spline interpolation is applied to reconstruct the fault curves. A selection of numerical examples with different behaviors and an application for solving scalar conservation law equations illustrate the performance of the algorithm.

Keywords Partition of unity · Radial basis functions · Polyharmonic splines · Fault curves · Fault points · Principal component regression

Mathematics Subject Classification (2010) 65Nxx · 41Axx

1 Introduction

Fault detection or approximation of curves across which a function is discontinuous is one of the useful and interesting problems with applications in different areas such

✉ Davoud Mirzaei
davoud.mirzaei@it.uu.se; d.mirzaei@sci.ui.ac.ir

Navid Soodbakhsh
n.soodbakhsh@sci.ui.ac.ir

[1] Department of Applied Mathematics and Computer Science, Faculty of Mathematics and Statistics, University of Isfahan, 81746-73441, Isfahan, Iran

[2] Division of Scientific Computing, Department of Information Technology, Uppsala University, Uppsala, Sweden

as edge detection in image processing, geophysical science, oil industry, and tomography [3–5, 9, 27, 44, 49, 55]. In all cases, a segmentation of an enclosed area that is related to a particular phenomenon is needed. In addition, in some approximation methods, the discontinuity curves and regions should be known in advance in order to obtain accurate solutions [6, 8].

For a given bivariate function, the curves on which the function itself and gradient of the function are discontinuous are called *ordinary* and *gradient* faults, respectively. Most of fault detection methods cannot distinguish between these two kinds of faults. However, recently some algorithms have been developed for determining both ordinary and gradient faults of an underlying function [11–13].

The basis of all fault detection methods includes the use of a derivative operator to indicate a cloud of points near the discontinuity curve. Some algorithms use a pure interpolation operator and Gibbs phenomenon measurements near discontinuity [2, 28, 36, 37, 44], and some others use gradient and Laplacian operators to define an indicator [12]. In [11], a central difference operator and a statistical procedure are applied and in [5], a local Taylor expansion and a polynomial annihilation criterion are used to indicate the fault clouds.

Assume that a set of scattered points with corresponding function values are available. In this paper, to deal with the non-uniform nature of scattered data set, we employ a localized meshfree approximation based on *polyharmonic* kernels in combination with *partition of unity* (PU) method for detecting a non-organized set of points on or close to the (unknown) faults of a function for which the data may have originated. A gradient- and a Laplacian-based indicators are defined for identifying a cloud of fault points from an original data set. In each PU patch, we apply the polyharmonic interpolation on *scaled* data points to prevent the instability of kernel matrices. Noting that, interpolation by polyharmonic kernels is the only scalable approximation among all *radial basis function* (RBF) approximations [20, 31].

At the first step, our algorithm results in an unorganized cloud of points near the discontinuity curve. To get an accurate reconstruction of the fault, we employ a *principal component regression* (PCR) [34, 35, 39] in combination with PU approximation to generate a second set of points which are supposed to be closer to the fault curve than the primary detected set. We apply PCR instead of linear least-squares approximation for better curve fittings especially in subregions on which the fault curve has a near vertical direction.

At the final step, an ordered subset of the previous narrowed fault points is extracted and a smooth parametric spline interpolation is employed to reconstruct the fault curve.

We also address the situations with multiple fault curves and special cases with intersections and multi-branch configurations.

Sufficient number of numerical examples illustrates the performance and efficiency of the method, including the detection and reconstruction of multi-branch and closed faults, and an application for solving a conservation law problem via the finite volume method (FVM).

2 Polyharmonic spline interpolation

Polyharmonic spline (PHS) interpolation is a particular case of interpolation with conditionally positive definite RBFs [24, 31, 53]. Assume that $\phi : \mathbb{R}^d \to \mathbb{R}$ is a conditionally positive definite function of order $m+1$, i.e., with respect to polynomial space $\mathbb{P}_m(\mathbb{R}^d)$. Let $\Omega \subset \mathbb{R}^d$ be a bounded region. The RBF interpolation of a function $f : \Omega \to \mathbb{R}$ on a discrete set $X = \{x_1, \ldots, x_N\} \subset \Omega$ is given by

$$s_{f,X}(x) = \sum_{j=1}^{N} \alpha_j \phi(x - x_j) + \sum_{n=1}^{Q} a_n p_n(x) \tag{1}$$

where $\{p_1, \ldots, p_Q\}$ is a basis for $\mathbb{P}_m(\mathbb{R}^d)$, and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)^T$ and $\boldsymbol{a} = (a_1, \ldots, a_Q)^T$ satisfy

$$\begin{bmatrix} K & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{a} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ 0 \end{bmatrix} \tag{2}$$

where $K \in \mathbb{R}^{N\times N}$ with $K_{jk} = \phi(x_j - x_k)$, $k, j = 1, \ldots, N$, $P \in \mathbb{R}^{N\times Q}$ with $P_{jn} = p_n(x_j)$, $j = 1, \ldots, N$, $n = 1, \ldots, Q$, and $\boldsymbol{f} = (f(x_1), \ldots, f(x_N))^T$. We also need to assume $N \geqslant Q$ and X is $\mathbb{P}_m^d$-unisolvent to have a full rank matrix P. On the other hand, since ϕ is conditionally positive definite of order $m+1$, the symmetric matrix K is positive definite on $\ker(P^T)$ as a subspace of $\mathbb{R}^N$. These all guarantee that the interpolation system is uniquely solvable. The interpolant $s_{f,X}$ can also be written in the Lagrange form as

$$s_{f,X}(x) = \sum_{j=1}^{N} u_j(x) f(x_j), \tag{3}$$

where $(u_1(x), \ldots, u_N(x))^T =: \boldsymbol{u}(x)$ satisfies

$$\begin{bmatrix} K & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}(x) \\ \boldsymbol{\lambda}(x) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\phi}(x) \\ \boldsymbol{p}(x) \end{bmatrix}, \tag{4}$$

for $\boldsymbol{\phi}(x) = (\phi(x - x_1), \ldots, \phi(x - x_N))^T$ and $\boldsymbol{p}(x) = (p_1(x), \ldots, p_Q(x))^T$. The Lagrange functions possess the property $u_j(x_k) = \delta_{kj}$.

In the case of PHS interpolation, the function ϕ is defined as $\phi(x) = \varphi(\|x\|_2)$ for all $x \in \mathbb{R}^d$ where

$$\varphi(r) := \begin{cases} r^\beta \log r, & \beta \text{ even} \\ r^\beta, & \text{otherwise} \end{cases} \tag{5}$$

for real number $\beta > 0$. The PHS function φ is (up to a sign) conditionally positive definite of order $m + 1 = \lfloor \beta/2 \rfloor + 1$.

Derivatives of f are approximated by corresponding derivatives of $s_{f,X}$, i.e.,

$$Ls_{f,X}(x) = \sum_{j=1}^{N} Lu_j(x) f(x_j),$$

where $[Lu_1(x), \ldots, Lu_N(x)]^T =: L\boldsymbol{u}(x)$ is solution of the same system (4) with the new right-hand side $\left[L\boldsymbol{\phi}^T(x)\;\; Lp^T(x)\right]^T$, i.e.,

$$\begin{bmatrix} K & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} L\boldsymbol{u}(x) \\ L\boldsymbol{\lambda}(x) \end{bmatrix} = \begin{bmatrix} L\boldsymbol{\phi}(x) \\ L\boldsymbol{p}(x) \end{bmatrix}. \tag{6}$$

Tackling the ill-conditioned system (4) is one the important issues in RBF approximation methods. As it is proved in [53, Chap. 12], for polyharmonic kernels, the condition number of this system grows algebraically with respect to the minimum spacing distance between interpolation points. To overcome this problem, the polyharmonic interpolation matrix can be formed and solved on scaled data points with a spacing distance of $\mathcal{O}(1)$. This is motivated by the 5-point star classical FD formula for $\Delta u(0,0)$ on points $X = \{(0,0), (h,0), (-h,0), (0,h), (0,-h)\}$. If points are scaled to

$$\frac{X}{h} = \{(0,0), (1,0), (-1,0), (0,1), (0,-1)\}$$

then the stencil weights are obtained as $[4, -1, -1, -1, -1]$, and the original weights are scaled as $h^{-2}[4, -1, -1, -1, -1]$. Here, 2 on the power of h is the scaling order of Δ. In the RBF context and on scattered points, this approach is applicable only for PHS and excludes all other well-known kernels. For more details, see [20, 31, 32]. In a more general case, assume that X is a set of points in a local domain $\mathcal{D}$ with fill distance h. Assume further that the polyharmonic approximation of $Lu(x)$ for a fixed $x \in \mathcal{D}$ is sought. Assume that L is a homogeneous operator with scaling order (homogeneity) s. For example, the scaling order of $L = \Delta$ is $s = 2$ and the scaling order of $L = D^\alpha$ is $s = |\alpha|$. If X is blown up (scaled) to points $\frac{X}{h}$ of average fill distance 1 and Lagrange functions Lu_j are calculated for the blown-up situation, then the Lagrange functions of the original situation are scaled as $h^{-s}Lu_j$. When polynomials of degree m are appended and monomials $\{x^\alpha\}_{|\alpha| \leqslant m}$ are used as a basis for $\mathbb{P}_m(\mathbb{R}^d)$, it is recommended to *shift* the points by the center of $\mathcal{D}$ and then scale by h to benefit from the local behavior of monomial basis functions around the origin. See [23, 24, 33, 43] for some applications of this strategy in localized RBF methods for numerical solution of PDEs and rational interpolation of singular functions.

3 Partition of unity approximation

The partition of unity (PU) approximation will be used twice in our detection algorithm. In approximation theory, the PU method is an efficient technique to obtain a sparse global approximation by joining a set of localized and well-conditioned approximations. The first combination of partition of unity with RBF interpolations goes back to [52] and [53] by Holger Wendland (see also [24]). Then, this approach has been extensively used for numerical solution of partial differential equations [14–18, 40, 43, 45]. A recent application to implicit surface reconstruction is provided in [21].

In PU methods, the global domain Ω is covered by a set of open, bounded, and overlapped subdomains Ω_ℓ, $\ell = 1, 2, \ldots, N_c$, where $\Omega \subset \cup_\ell \Omega_\ell$, and a set of PU weights is defined on this covering. The nonnegative and compact support functions

$w_\ell : \mathbb{R}^d \to \mathbb{R}$, $\ell = 1, 2, \ldots, N_c$, with supports $\overline{\Omega_\ell}$ are called PU with respect to covering $\{\Omega_\ell\}$ if

$$\sum_{\ell=1}^{N_c} w_\ell(x) = 1, \quad \forall x \in \Omega.$$

If we start with an overlapping covering $\{\Omega_\ell\}$ of Ω and we assume V_ℓ is an approximation space on Ω_ℓ and $s_\ell \in V_\ell$ is a local approximation of a function f on Ω_ℓ, then

$$s = \sum_{\ell=1}^{N_c} w_\ell s_\ell$$

is a global PU approximation of f on Ω. This global approximation is formed by joining the local approximants s_ℓ via PU weights w_ℓ. A possible choice for w_ℓ is given by Shepard's weights

$$w_\ell(x) = \frac{\psi_\ell(x)}{\sum_{j=1}^{N_c} \psi_j(x)}, \quad 1 \leqslant \ell \leqslant N_c,$$

where ψ_ℓ are nonnegative, nonvanishing, and compactly supported functions on Ω_ℓ. In a usual way, derivatives of f are approximated by derivatives of s, i.e.,

$$D^\alpha f \approx D^\alpha s = \sum_{\ell=1}^{N_c} D^\alpha (w_\ell s_\ell),$$

or in a general form, for a linear differential operator L with constant coefficients, we have

$$Lf \approx Ls = \sum_{\ell=1}^{N_c} L(w_\ell s_\ell).$$

So, the Leibniz's rule should be applied to compute the derivatives of products $w_\ell s_\ell$. This standard technique is complicated, somewhat, and requires smooth PU weight functions. In [43], an alternative approach is suggested:

$$D^\alpha f \approx \sum_{\ell=1}^{N_c} w_\ell \, D^\alpha s_\ell =: s^\alpha,$$

where $D^\alpha f$ is directly approximated without any detour via the approximant s of f itself. Analogously, for a general case with operator L, we may write

$$Lf \approx \sum_{\ell=1}^{N_c} w_\ell \, Ls_\ell =: s^L.$$

Theoretical results show the same convergence properties for both standard and direct approaches [43], while the second approach is simpler and allows discontinuous PU weights to develop some faster algorithms for approximating the derivatives and solving partial differential equations.

The RBF-PU is a special case in which the local approximants Ls_ℓ are obtained by the RBF approximation on trial sets $X_\ell = X \cap \Omega_\ell$ in local domains $\Omega_\ell \cap \Omega$ with global index family $J_\ell := \{j \in \{1, \dots, N\} : x_j \in X_\ell\}$. In this case, we have

$$Ls(x) = \sum_{\ell=1}^{N_c} \sum_{j \in J_\ell} L(w_\ell(x) u_j(\ell; x)) f(x_j)$$

for the standard approach, and

$$s^L(x) = \sum_{\ell=1}^{N_c} \sum_{j \in J_\ell} w_\ell(x) L u_j(\ell; x) f(x_j) \tag{7}$$

for the direct approach. Here, $u_j(\ell; x)$ are Lagrange RBF functions on patches Ω_ℓ. A simple covering for Ω can be constructed via a set of overlapping balls $\Omega_\ell = B(y_\ell, \rho_\ell)$ where $y_\ell \in \mathbb{R}^d$ are patch centers and ρ_ℓ are patch radii. To have a fixed patch radius $\rho_\ell \equiv \rho$, we can use the following setup for points, parameters, and domain sizes. We assume the set X has fill distance

$$h = h_{X,\Omega} = \max_{x \in \Omega} \min_{x_k \in X} \|x - x_k\|_2.$$

The fill distance indicates how well the points in the set X fill out the domain Ω. Geometrically, h is the radius of the largest possible empty ball that can be placed among the data locations X inside Ω. A data set $\{y_1, \dots, y_{N_c}\}$ with space distance

$$h_{\text{cov}} = C_{\text{cov}} h, \quad C_{\text{cov}} > 1$$

is used for patch centers. The constant C_{cov} controls the number of patches, N_c, compared with the number of interpolation points in X. The radius ρ should be large enough to guarantee the inclusion $\Omega \subset \cup_\ell B(y_\ell, \rho_\ell)$, and to allow enough interpolation points in each patch for a well-defined and accurate local approximation. Thus, we assume

$$\rho_\ell = \rho = C_{\text{ovlp}} h_{\text{cov}},$$

and we let the overlap constant C_{ovlp} be large enough to ensure the above requirements. It is also possible to assign variable patch sizes ρ_ℓ per any patch center y_ℓ. For example, we can obtain ρ_ℓ is such way that there exists a certain number of interpolation points in each patch Ω_ℓ. In this case, we must sure that the inclusion property $\Omega \subset \cup_\ell \Omega_\ell$ is still satisfied. In numerical examples of Section 7, we use both fixed and variable patch radius strategies.

Smooth Shepard weight functions are frequently used in PU approximations (see, for example, [40, 45, 52]). A discontinuous PU weight is also suggested in [43] that highly simplifies the RBF-PU algorithms for solving partial differential equations. Assume the PU weight $w_\ell(x)$ takes the constant value 1 if y_ℓ is the closest center to x and the constant value 0, otherwise. For definition, let

$$I_{\min}(x) = \operatorname*{arg\,min}_{\ell \in I(x)} \|x - y_\ell\|_2$$

and $I_{\min,1}(x)$ be the first component of $I_{\min}(x)$, as $I_{\min}(x)$ may contain more than one index ℓ. The weight function is then defined by

$$w_\ell(x) := \begin{cases} 1, & \ell = I_{\min,1}(x) \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

With this definition, we give the total weight 1 to the closest patch and null weights to other patches. In fact, a local set $X_\ell = \Omega_\ell \cap X$ is a common interpolation set for all evaluation points x_k with $\|x_k - y_\ell\|_2 \leq \|x_k - y_j\|_2$ for $j = 1, \ldots, N_c$ and $j \neq \ell$. In another view in a 2D domain, by drawing the *Voronoi tiles* of centers $\{y_1, \ldots, y_{N_c}\}$, this means that all evaluation points in tile ℓ use the same local set X_ℓ as their interpolation set [43].

4 Principal component regression

Assume that $X \in \mathbb{R}^{d\times n}$ is a data matrix containing n points (n columns) in the d dimensional space. PCR finds the best k rank approximation to d dimensional data matrix X for $0 < k < d$. It is known that a low rank approximation can be solved by the singular value decomposition (SVD). Let $\mu_X \in \mathbb{R}^{d\times 1}$ be the vector of sample means along each row of X. Subtract each columns of X (each point) by μ_X and denote the resulted mean zero data matrix by X^0. The SVD of X^0 is given by

$$X^0 = U\Sigma V^T$$

where $U \in \mathbb{R}^{d\times d}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal matrices and $\Sigma \in \mathbb{R}^{d\times n}$ is a diagonal matrix carrying the singular values σ_j of X decreasingly on its diagonal. Equivalently

$$X^0 = \sum_{j=1}^{d} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^T$$

where principal components $\boldsymbol{u}_j$ and $\boldsymbol{v}_j$ are columns of U and V, respectively. Then for $k \leqslant d$

$$X_k^0 = \sum_{j=1}^{k} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^T = U_k \Sigma_k V_k^T$$

is the closest rank k matrix to X^0 (with respect to 2, Frobenius and any matrix norm that depends only on singular values) where U_k and V_k consist of first k columns of U and V, respectively, and $\Sigma_k = \operatorname{diag}\{\sigma_1, \ldots, \sigma_k\}$ [51]. Clearly, the data matrix X_k which is obtained by adding the mean vector μ_X to all columns of X_k^0 is the closest rank k matrix to X. Columns of X_k (as points in $\mathbb{R}^d$) are located on a k dimensional subspace of $\mathbb{R}^d$.

In this paper, we need the special case $d = 2$ and $k = 1$ for narrowing a cloud of fault points around a discontinuity curve in a two-dimensional domain Ω (see Section 5.2).

5 Detection algorithm

Assume that f is a piecewise smooth real valued function with finite jump discontinuity across some curves on a two-dimensional domain Ω. If the union of such curves is denoted by $\mathcal{F}$, we assume that the measure of $\mathcal{F}$ is zero and f is smooth in $\Omega \setminus \mathcal{F}$. Assume further that a set of scattered points $X \subset \Omega$ and associated function values $f(x)$, $x \in X$, are given. The algorithm consists of three steps: (1) picking out a set of points, called *fault points* as a subset of X, on or close to discontinuity curves using a partition of unity polyharmonic-based approximation, (2) narrowing the fault points using a partition of unity and PCR algorithm, and (3) constructing the fault curve using a parametric interpolation. In the following subsections, we will describe these steps.

5.1 Fault point detection

We aim to detect a small subset F of X consisting of points close to $\mathcal{F}$ by considering a procedure based on local kernel interpolation. The points in F are called fault points. In [12], a minimal numerical differentiation formula (MNDF) based on local multivariate polynomial reproduction is given to approximate the function derivatives. This approach is a generalized finite difference (FD) method and the stencil weights are uniquely determined by minimizing their weighted ℓ_1 or ℓ_2 norm in a suitable way [19].

In this paper, we use the *direct* PU approximation for localization and PHS kernels for local approximations. We measure and compare the approximate gradient and Laplacian values with some threshold parameters to find a set of fault points F close to the fault curve $\mathcal{F}$.

In each patch Ω_ℓ, we apply the discontinuous PU weight (8) and PHS kernels $\varphi_1(r) := r$ and $\varphi_3(r) := r^3$ to approximate gradient and Laplacian functions, respectively. Note that, φ_1 and φ_3 are conditionally positive definite of orders 1 and 2, respectively, and thus polynomials of orders at least 1 and 2 need to be appended to their corresponding RBF expansions to obtain well-defined interpolations on unisolvent sets.

According to the PU procedure, each point x_k is subjected to a local set $X_\ell = \Omega_\ell \cap X$ for an index $\ell \in \{1, \ldots, N_c\}$ in which $\ell = I_{\min,1}(x_k) =: \ell(k)$. The difference between this approach and the RBF-FD method is that in the new method a set X_ℓ may be shared with many points x_k while in the RBF-FD, each stencil X_k is associated with a unique evaluation point x_k. This special type of the *direct* RBF-PU (D-RBF-PU) method [43] is similar to (but not identical with) the *overlapped* RBF-FD method of [46]. From (7) and (8), we have

$$\begin{aligned} Lf(x_k) \approx s^L(x_k) &= \sum_{\ell=1}^{N_c} \sum_{j \in J_\ell} w_\ell(x_k) L u_j^*(\ell; x_k) f(x_j) \\ &= \sum_{j \in J_{\ell(k)}} L u_j^*(\ell(k); x_k) f(x_j) =: \sum_{j \in J_{\ell(k)}} \xi_{j,k}^L f(x_j), \end{aligned}$$

where $\xi^L_{j,k} = Lu^*_j(\ell(k); x)|_{x=x_k}$ are generalized (related to operator L) Lagrange function values on set $X_\ell = \{x_j : j \in J_\ell\}$ evaluated at x_k. In programming, we loop over patches and look for indices of points x_k in which a prescribed patch ℓ is their closest patch. Such index family will be denoted by $k(\ell)$.

Usually, the gradient and the Laplacian operators (∇ and Δ) are used in fault curve detection algorithms. An *ordinary fault indicator* is defined as

$$I_\nabla(x_k) := \|s^\nabla(x_k)\|_2, \quad k = 1, \ldots, N, \tag{9}$$

and a *gradient fault indicator* is defined as

$$I_\Delta(x_k) := |s^\Delta(x_k)|, \quad k = 1, \ldots, N. \tag{10}$$

A large value of $I_\nabla(x_k)$ shows that the smoothness of f is deficient at or near x_k, and the large value of $I_\Delta(x_k)$ indicates the same behavior for both f and its gradient. For quantification, we define

$$F(X, L, \delta) := \{x_k \in X : I_L(x_k) > \delta\},$$

where L stands for either ∇ or Δ, and δ is a proper threshold parameter. Thus, we mark $F(X, \nabla, \delta_1)$ and $F(X, \Delta, \delta_2)$ as points which are close to ordinary and gradient faults, respectively, where δ_1 and δ_2 should be set appropriately to get accurate detections. In [12], a marking method based on the median values of the computed indicators is applied. Since for functions with constant (linear) values in large areas of the domain, the indicator I_∇ (I_Δ) is close to zero at points belonging to those areas, the medians fall around the zero, and thus many points are indicated as fault points even if they are not close to a fault curve. To overcome this possible problem, a doubly nested marking strategy is used in [12]. Here, we still use medians but with a different strategy. Using notations $I_\nabla(X)$ and $I_\Delta(X)$ for $\{I_\nabla(x_k) : x_k \in X\}$ and $\{I_\Delta(x_k) : x_k \in X\}$, respectively, we set

$$\delta_2 = C_M \mathrm{median}(I_\Delta(F(X, \Delta, \alpha_2))), \quad \alpha_2 = C_L/h \tag{11}$$
$$\delta_1 = C_M \mathrm{median}(I_\nabla(F(F(X, \Delta, \delta_2), \nabla, \alpha_1))), \quad \alpha_1 = C_G/\sqrt{h} \tag{12}$$

where h is the fill-distance of X, and C_M, C_G, and C_L are proper constants that should be set by the user manually. In the above process, we first obtain the threshold parameter δ_2 by (11) to form the set $F(X, \Delta, \delta_2)$ containing points close to both ordinary and gradient discontinuities. Then the points close to ordinary faults can be extracted from this set instead of the initial large set X. Thus, the threshold parameter δ_1 is obtained by (12) and ordinary fault points F_∇ are obtained as

$$F_\nabla = F(F(X, \Delta, \delta_2), \nabla, \delta_1). \tag{13}$$

Obviously the set of points close to gradient discontinuities, F_Δ, is

$$F_\Delta = F(X, \Delta, \delta_2) \setminus F_\nabla. \tag{14}$$

Experiments show that F_Δ still contains some points in the neighborhoods of ordinary discontinuities. Following [12], we modify F_Δ as follows:

$$F_\Delta = \{x \in F_\Delta : \ |B(x, \eta) \cap F_\nabla| < \frac{1}{5}|B(x, \eta)|\} \tag{15}$$

where $B(x, \eta)$ is a ball with center x and radius η on set $F(X, \Delta, \delta_2)$, and $|B|$ denotes the cardinality of set B. Parameter η is set proportional to fill-distance of the initial data set such that $B(x, \eta)$ contains a sufficient number of points.

Algorithm 1 presents the fault point detection procedure step by step. Algorithm 2 is called in Algorithms 1, and 3 is called in Algorithm 2.

```
Data: Function $f : \Omega \to \mathbb{R}^2$, initial set of points $X \subset \Omega$ of size $N \times 2$, patch
      centers $Y$ of size $N_c \times 2$, and $n$ (number of points in each patch), $C_L$,
      $C_G$, $C_M$, and $\eta$.
Result: Fault sets $F_\nabla$ and $F_\Delta$.
1: Obtain the index matrix $\mathcal{I}$ of size $N_c \times n$ where row $\ell$ contains indices of $n$
points from $X$ located in patch $\Omega_\ell$, i.e., $J_\ell$;
2: Set $\rho_\ell$ as the maximum distances between $y_\ell$ and its surrounding $n$
interpolation points;
3: Compute $s^\nabla$ by calling Algorithm 2 with inputs $X$, $Y$, $\{\rho_\ell\}$, $\mathcal{I}$, $L = \nabla$, and
$f|_X$;
4: Compute $s^\Delta$ by calling Algorithm 2 with inputs $X$, $Y$, $\{\rho_\ell\}$, $\mathcal{I}$, $L = \Delta$, and
$f|_X$;
5: Compute indicators $I_\nabla$ and $I_\Delta$ via (9) and (10);
6: Extract fault point sets $F_\nabla$ and $F_\Delta$ via (13) and (14) and parameters $C_L$, $C_G$
and $C_M$;
7: Update $F_\Delta$ using (15) and parameter $\eta$;
8: Return $F_\nabla$ and $F_\Delta$;
```

Algorithm 1 Fault points generation.

We note that steps 1 and 2 of Algorithm 1 can be run simultaneously using a proper nearest search algorithm.

```
Data: Centers $X$ of size $N \times 2$, patch centers $Y$ of size $N_c \times 2$, patches $\{\rho_\ell\}$, index
      matrix $\mathcal{I}$, operator $L$, and vector of function values $f$.
Result: Approximate vector $s^L$ of size $N \times 1$.
1: Initialize $s^L = 0$;
2: for $\ell = 1 : N_c$ do
    Extract $X_\ell = X \cap \Omega_\ell$ using the global index family $J_\ell$ from $\mathcal{I}$;
    Find the index family $k(\ell)$ of points from $X_\ell$ for which $y_\ell$ is their closest patch
    center;
    Set $\widehat{X}_\ell = X(k(\ell), :)$;
    Call Algorithm 3 to compute Lagrange matrix $U$ for operator $L$ based on $X_\ell$ at $\widehat{X}_\ell$;
    Set $f_\ell = f(J_\ell)$ and compute the matrix-vector product $s_\ell = Uf_\ell$;
    Update $s^L$ via $s^L(k(\ell)) = s_\ell$;
3: Return $s^L$.
```

Algorithm 2 The RBF-PU subroutine with constant generated PU weight.

```
Data: Test set $\widehat{X}$ of size $m \times 2$, trial set $X$ of size $n \times 2$, the center $y_\ell$ and radius $\rho_\ell$ of
      patch $\Omega_\ell$, and operator $L$.
Result: Lagrange matrix $U$ of size $m \times n$.
1: Set $X \leftarrow (X - y_\ell)/\rho_\ell$ and $\widehat{X} \leftarrow (\widehat{X} - y_\ell)/\rho_\ell$ (shift and scale);
2: Compute kernel matrix $K$ and polynomial matrix $P$ on $X$, and $L\boldsymbol{\phi}$ and $L\boldsymbol{p}$ on $\widehat{X}$;
3: Solve (6) for $U$;
4: Determine $s$ the scaling order of $L$;
5: Update (re-scale) $U$ via $U \leftarrow \rho_\ell^{-s} U$ ;
6: Return $U$;
```

Algorithm 3 PHS Lagrange functions.

5.2 Narrowing step

In this subsection, we give a narrowing procedure to move a cloud of points approximately on a fault curve that may be of either ordinary or gradient type. Let us denote the set of detected fault points by

$$F = \{x_1, \ldots, x_M\}$$

as a small subset of the initial set X. To approximate the fault curve, the set F needs to be narrowed more.

In [12], an orthogonal distance least squares regression is used in this step to handle the cases in which the points near a parametric curve are distributed *almost vertically*. In fact, the least-squares approximation is used twice: for a coordinates rotation at first and for a curve fitting then (see also [50]). One of advantages of this method is that it can be used to obtain regressions of arbitrary orders. In this paper, we use the standard PCR method to obtain a linear regression using SVD, which seems enough for our algorithm. First, we consider the set F as a $2 \times M$ matrix where each column stands for a point in $\mathbb{R}^2$. Then we compute the centered data matrix $F^0 = F - \mu_F$ and its reduced SVD $F^0 = U\Sigma V^T$. Here μ_F is the sample mean of rows of F. Finally, we accept the best first rank matrix approximation $F_1 = U_1\Sigma_1 V_1^T + \mu_F$ as narrowing points (see Fig. 1).

But since the fault curves are usually nonlinear, we apply this procedure on local subdomains and blend the local approximants in a proper way to obtain a global nonlinear configuration. For this purpose, we employ a new PU approximation based on a new set of patches $\{\Omega'_\ell\}$ for $\ell = 1, \ldots, N'_c$. In this step, we use a constant radius $\rho = C_{\text{ovlp}} h_{\text{cov}}$ for all patches, i.e., $\Omega'_\ell = B(y'_\ell, \rho)$ for $\ell = 1, \ldots, N'_c$. The set of patch centers $\{y'_\ell\}$ is a coarsened subset of detected fault points F which is obtained by Algorithm 5 (below) for $H_1 = H_2 = \rho$ and $\tilde{F} = F$. Note that, the size of the PU problem in this step is considerably less than that of the first PU approximation because now we are working on a much smaller set of points around a one-dimensional fault curve.

We assume $F'_\ell = F \cap \Omega'_\ell$ and $J'_\ell = \{j : x_j \in F'_\ell\}$ with $|J'_\ell| = n_\ell$. Now, the PCR algorithm is applied on each cloud F'_ℓ for $\ell = 1, \ldots, N'_c$ to get a new narrowing set

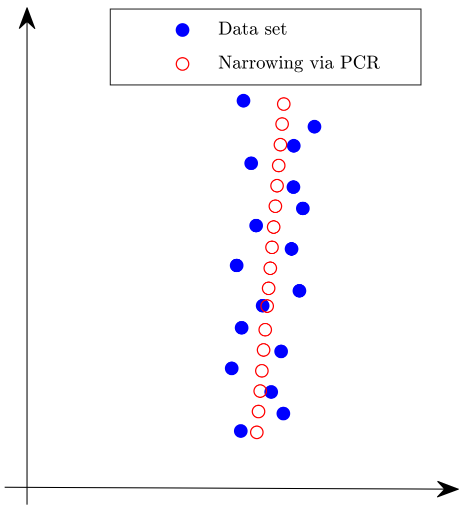

Fig. 1 Data set $F = U\Sigma V^T + \mu_F$ (blue and filled circles) and narrowing points $F_1 = U_1\Sigma_1 V_1^T + \mu_F$ (red circles)

F_ℓ. As described above, if F'_ℓ and F_ℓ are considered as $2 \times n_\ell$ matrices and the SVD of $F'_\ell - \mu_{F'_\ell}$ is denoted by $U\Sigma V^T$ then $F_\ell = U_1\Sigma_1 V_1^T + \mu_{F'_\ell}$.

Up to here, we have N'_c number of fault sets F_ℓ which are supposed to be closer than the set F to the (unknown) fault curve. Depending on the amount of overlap between patches, the cardinality of $\cup_\ell F_\ell$ is larger than that of the original set F. This means that a fault point x_k which belongs to more than one patches, say n patches, has n different approximation points from n different sets F_ℓ. To obtain a unique approximation for any fault point, we apply the PU approximation on covering $\{\Omega'_\ell\}$. If we use the smooth PU weights (3), then a smooth combination of these n approximations gives a unique approximation $\tilde{x}_k$ for x_k. We prefer to apply the discontinuous weight (8) due to its simplicity. In this case, depending on which center y'_ℓ is closer to x_k, the approximation point in its corresponding set F_ℓ is marked as the unique solution $\tilde{x}_k$ for x_k. Using this approach, we end with a new set

$$\tilde{F} = \{\tilde{x}_1, \ldots, \tilde{x}_M\}$$

which is obtained by thinning the cloud of detected set F by moving the points close to the fault curve. To further narrow the set of obtained points, we can apply the narrowing procedure once again by replacing F by $\tilde{F}$.

In programming, we apply this procedure by looping over patch centers rather than looping over fault points (see Algorithm 4).

Note that, we obtain local regressions per any patch and move several points x_k, $k \in k(\ell)$, close to the fault curve, simultaneously.

5.3 Fault curve reconstruction

Usually, the narrowed set $\tilde{F}$ includes lots of points giving us an opportunity to select an ordered subset of $\tilde{F}$ to reconstruct the fault curve using a parametric approximation method. We use a method similar to that is given in [12, 41].

```
Data: Fault set of points $F$, patch radius $\rho$
Result: Narrowed points $\tilde{F}$
1: Call Algorithm 5 to produce $N'_c$ centers $\{y'_\ell\}$ with $H_1 = H_2 = \rho$ and $\tilde{F} = F$;
2: Obtain the index family $\mathcal{I}$ for indices of points from $F$ located in each patch $\Omega'_\ell$;
3: Initialize the zero matrix $\tilde{F}$ with the same size as $F$;
4: for $\ell = 1 : N'_c$ do
    Extract $F'_\ell = F \cap \Omega'_\ell$ using the global index family $\mathcal{I}$;
    Find the index family $k(\ell)$ of fault points from $F'_\ell$ for which $\Omega'_\ell$ is their closest patch;
    Compute the row mean $\mu$ of $F'_\ell$ and set $F'_\ell \leftarrow F'_\ell - \mu$;
    Compute the reduced SVD $F'_\ell = U\Sigma V^T$;
    Compute new points $F_\ell = U(:,1)\Sigma_1 V(k(\ell),1)^T + \mu$;
    Update $\tilde{F}$ via $\tilde{F}(:,k(\ell)) = F_\ell$;
5: Return $\tilde{F}$;
```

Algorithm 4 Narrowing step using PU with weight function. (8) and PCR.

A point z from $\tilde{F}$ is selected randomly and a new ordered set F_{ord} is introduced which contains the only point z at the beginning but will be enlarged based on the following procedure.

First, we find the set of fault points F_z in H_1-neighborhood of z, i.e.,

$$F_z = \{x \in \tilde{F} : \|x - z\|_2 < H_1\}.$$

Then we obtain the direction u_z for which the variance of points in F_z is maximized. It is well known that this direction is the first column of the U factor in the reduced SVD $F_z^0 = U\Sigma V^T$ where F_z^0 is the mean zero data matrix. Now, two subsets F_z^+ and F_z^- of F_z are formed as

$$\begin{aligned} F_z^+ &= \{x \in F_z : (x - z) \cdot u_z > 0\} \\ F_z^- &= \{x \in F_z : (x - z) \cdot u_z \leqslant 0\}. \end{aligned}$$

Then points $z^+ \in F_z^+$ and $z^- \in F_z^-$, if any, are chosen such that

$$\begin{aligned} (z^+ - z) \cdot u_z &= \max_{x \in F_z^+} \{(x - z) \cdot u_z\} \\ (z^- - z) \cdot u_z &= \min_{x \in F_z^-} \{(x - z) \cdot u_z\} \end{aligned}$$

and are added to set F_{ord}. In fact, z^+ and z^- have maximum distances from z along the directions u_z and $-u_z$, respectively. This process is repeated from two points z^+ and z^- in both directions until no point is found in their neighborhood (see Fig. 2) or the distance between one of the newly selected points and one of the previously selected points in F_{ord} (or in all previously sets F_{ord} for cases with multiple fault curves (see Section 5.4)) is less than $H_1/2$. Using the last condition and checking the newly found points in the last iteration, it can be checked whether the fault intersects itself (see the left-hand side of Fig. 3) or whether it may approach another fault (see the right-hand side of Fig. 3) (see Algorithm 5 for a more general case).

Finally, we end with a sequence of ordered points allowing to reconstruct the fault curve using a parametric approximation method. We will apply the parametric *cubic*

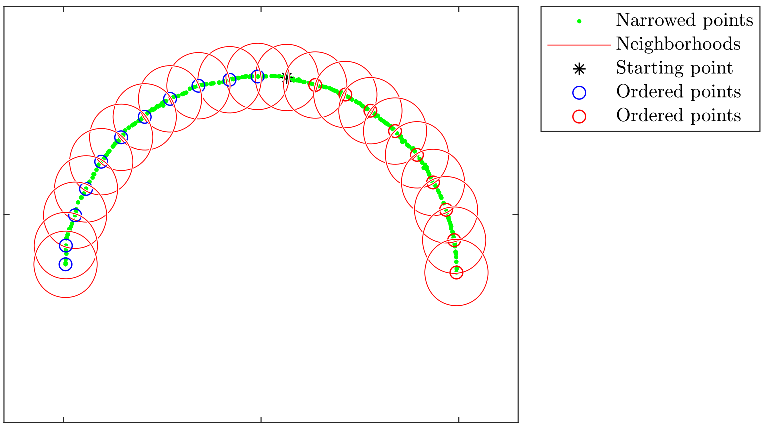

Fig. 2 Green dots are narrowed points and the black star is the first point where the algorithm starts. Red and blue circles are detected ordered points on positive and negative sides of the stating point

spline interpolation by adding a small smoothing parameter. Alternatively, one can simply connect the points successively by line segments to form a polygonal instead of a smooth spline curve.

5.4 Cases with multiple fault curves

There may be more than one fault (of either ordinary or gradient type) in the domain. So we continue the above algorithm to find other faults as follows. Consider a new set that includes all points from $\tilde{F}$ whose distance to points in F_{ord} is greater than a parameter H_2. If this new set is empty, the algorithm is finished and we have no other fault. Otherwise, we select a new random point z from the points in this new set and continue the previous procedure to obtain a new ordered set F_{ord} for the next fault (see right hand side of Fig. 3). Again the parametric spline interpolation is applied to reconstruct the new curve. This algorithm is repeated until none of points in $\tilde{F}$ has distance greater than H_2 to points in all previous sets F_{ord}.

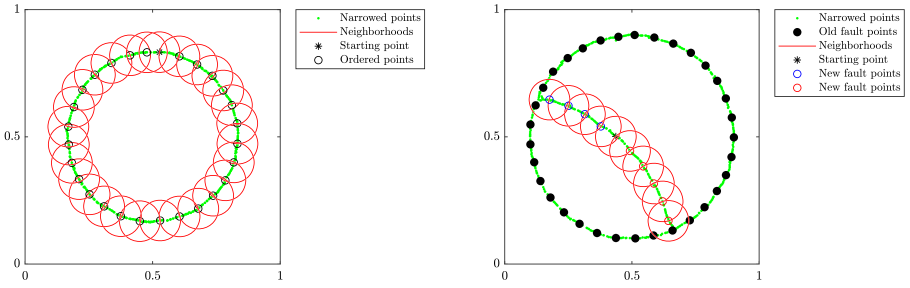

Fig. 3 Detected ordered points on a close curve (left) and starting to detect another set of ordered points on another fault (right)

In Algorithm 5, the steps of selecting a set of ordered fault points from the larger set $\tilde{F}$ are outlined. This algorithm works for cases with multiple fault curves as well.

```
Data: Set of points $\tilde{F}$, parameters $H_1$ and $H_2$
Result: A sequence of $m$ ordered set points $F_{ord,j}$ for $j = 1, 2, \dots, m$
1: Choose a points $z$ from $\tilde{F}$;
2: Set $m = 0$;
3: while exist a $z$ in $\tilde{F}$ do
    Set $m \leftarrow m + 1$;
    Set $F_{ord,m} = \{z\}$;
    Obtain $F_z, u_z, F_z^+, z^+, F_z^-, z^-$;
    while exist $z^+$ and $distance(z^+, \cup_{j=1}^m F_{ord,j}) \geqslant H_1/2$ do
        Update $F_{ord,m}$ by adding $z^+$;
        Set $z = z^+$;
        Determine $F_z, u_z, F_z^+$;
        Obtain $z^+$, if any;
    while exist $z^-$ and $distance(z^-, \cup_{j=1}^m F_{ord,j}) \geqslant H_1/2$ do
        Update $F_{ord,m}$ by adding $z^-$;
        Set $z = z^-$;
        Determine $F_z, u_z, F_z^-$;
        Obtain $z^-$, if any;
    Set $\tilde{F} \leftarrow \{x \in \tilde{F} : distance(x, \cup_{j=1}^m F_{ord,j}) > H_2\}$;
    Select a new $z$ from $\tilde{F}$, if any;
4: Return $F_{ord,j}$, $j = 1, 2 \dots, m$.
```

Algorithm 5 Selecting ordered fault points on multiple fault curves.

5.5 Cases with intersections

There may be some faults that intersect each other. Suppose that the described algorithm yields m different sequences of ordered points corresponding to m different faults. By construction, the sequences do not share the common points on different faults; thus, the reconstructed curves may not intersect even if their corresponding exact faults do intersect. To resolve this problem, we apply the following procedure which is similar to that is given in [12]:

1. The head and the tail of each fault is reconstructed by linear interpolation (line segments) based on the first two and the last two points, respectively (see upper panels of Fig. 4).
2. The line segments are extended out within Ω with length at most H_3 and new end points (if located in Ω) are determined (see upper panels of Fig. 4).
3. For each new end point e with corresponding line segment ℓ, its closest points z_p (on the positive side of ℓ) and z_n (on the negative side of ℓ) from other sequences are selected. If distances $\|e - z_p\|_2$ and $\|e - z_n\|_2$ are less than a prescribed

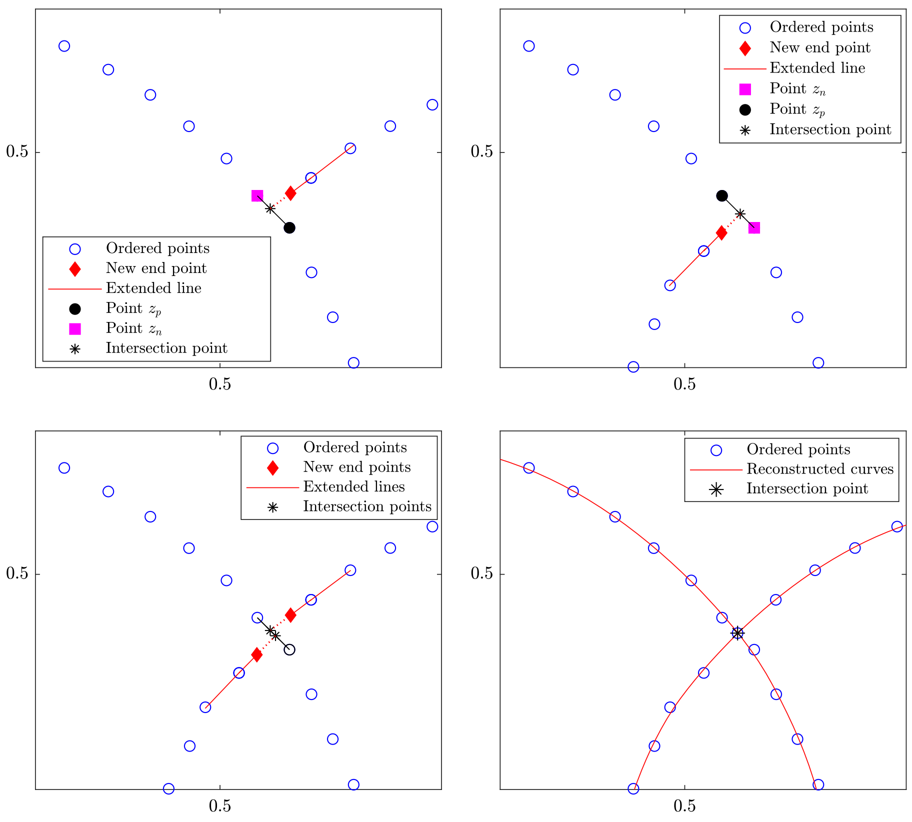

Fig. 4 Treating the fault curves with intersection

threshold H_4, we mark the intersection of (extension of) ℓ and the line segment between z_p and z_n as an intersection point (see upper panels of Fig. 4).

4. Intersection points whose distances are less than a prescribed threshold H_5 are replaced by their average (see the lower panels of Fig. 4).
5. The ordered sequences are updated either by adding one or two intersection points or by joining to another sequence and an intersection point.

In Algorithm 6, by $[[x, y]]$, we mean a line segment with x and y as its end points.

6 Parameter selection and the main algorithm

The parameter H_1 in Algorithm 5 is the approximate distance between two consecutive fault points in which the parametric interpolation is built upon. In order to have an accurate reconstruction, H_1 must be selected small enough, but on the other hand, it should be large enough to ensure that if the narrowed points on a fault are spaced, the fault will not split into two or more pieces. Usually, H_1 is set proportional to the

```
Data: Ordered point sets $F_j$, $j = 1, 2, \ldots, m$, parameters $H_3$, $H_4$ and $H_5$
Result: Updated ordered points $\widehat{F}_j$, $j = 1, 2, \ldots, n$, $n \leqslant m$
1: for $j = 1 : m$ do
    Obtain line segments $\ell^{head}$ and $\ell^{tail}$ between the first two and the last two points in set $F_j$, respectively;
    Extend out the line segments with length $H_3$ to obtain new end points $e^{head}$ and $e^{tail}$ ;
    Obtain for each new end point their closest points $z_p^{head}$, $z_n^{head}$, $z_p^{tail}$, $z_n^{tail}$ out of $F_k$, $k \neq j$;
    if $\|e^{head} - z_p^{head}\|_2 \leqslant H_4$ and $\|e^{head} - z_n^{head}\|_2 \leqslant H_4$ then
        Compute $x_j^{head}$ the intersection of $\ell^{head}$ and $[[z_p^{head}, z_n^{head}]]$
    else
        Leave $x_j^{head}$ empty
    if $\|e^{tail} - z_p^{tail}\|_2 \leqslant H_4$ and $\|e^{tail} - z_n^{tail}\|_2 \leqslant H_4$ then
        Compute $x_j^{tail}$ the intersection of $\ell^{tail}$ and $[[z_p^{tail}, z_n^{tail}]]$
    else
        Leave $x_j^{tail}$ empty
2: Replace the nonempty intersection points $x_j^{head}$ and $x_j^{tail}$, $j = 1, \ldots, m$, whose distances are less than $H_5$ by their averages;
3: Add average points to the corresponding sets;
4: Merge two sets $F_j$ and $F_k$ if their new heads and/or tails points are averaged;
5: Return new ordered sets $\widehat{F}_j$, $j = 1, \ldots, n$;
```

Algorithm 6 Fixing intersections.

fill-distance of the initial data set. When ordered points on one fault are obtained, the algorithm starts again from one new point on another fault. The distance between this new point to those of the previous faults is more than H_2. Thus, H_2 should be chosen large enough so that the new point falls on another fault and avoids reconstructing a fault twice. In our experiments, the value $H_2 = H_1$ leads to satisfactory results.

In Algorithm 6, the parameter H_3 should be close to $H_1/2$ because from Section 5.3, the minimum distance between two disjoint faults is determined by $H_1/2$. We set $H_3 = H_1/2$. On the other hand, the threshold parameter H_4 should be proportional to H_1 because H_1 is the maximum distance between two consecutive points in each ordered sequence. Here we set $H_4 = 2H_1$. Finally, H_5 should be selected small enough to unite the intersection points that are very close to each other. The choice $H_5 = H_1$ is quite nice in all experiments.

Note that all parameters $H_k, k = 1, \ldots, 5$ are explicitly related to the approximate fill-distance h of the initial data set X via $H_5 = H_4/2 = 2H_3 = H_2 = H_1 = C_H h$. In our experiments, we use $C_H = 6$. The fill distance h is approximated by $h = 1/\sqrt{N}$ where N is the number of initial points in X.

To approximate s^∇ and s^Δ for indicators, in the first PU algorithm, we use variable patch radius ρ_ℓ per any patch center y_ℓ. Using a nearest search algorithm, we select $n = 12$ nearest points to y_ℓ and then adjust the radius ρ_ℓ as the maximum distance between y_ℓ and that surrounding 12 points. In examples, $Y = \{y_1, \ldots, y_{N_c}\}$ is assumed to be a grid set in the domain Ω with spacing distance $h_{\rm cov} = C_{\rm cov}h = 2.5h$. This parameter selection guarantees the inclusion $\Omega \subset \cup_\ell \Omega_\ell$ for both random and Halton point sets in our numerical examples. Moreover, for the second PU algorithm in the narrowing step, we use constant radius $\rho = C_{\rm ovlp}h_{\rm cov}$ where we set $C_{\rm ovlp} = 1.5$. We also set $C_M = 1/4$, $C_G = 1$, $C_L = 1/2$, and $\eta = 4h$ in all examples unless specified otherwise. Finally, the main algorithm of the fault detection method can be written as follows.

```
Data: Function $f : \Omega \to \mathbb{R}^2$, initial set of points $X$ in $\Omega$, patch centers $Y$, $n$ (number of
      points in each patch), $C_{\rm cov}$, and $C_{\rm ovlp}$, $C_L$, $C_G$, $C_M$, and $\eta$.
Result: Fault curves.
1: Set $h = 1/\sqrt{N}$ where $N$ is the number of points in $X$;
2: Generate fault sets $F_\nabla$ and $F_\Delta$ by calling Algorithm 1 with inputs $f$, $X$, $Y$, $n$, $C_L$, $C_G$,
   $C_M$, and $\eta$;
3: Narrow $F_\nabla$ and $F_\Delta$ using Algorithm 4 with $\rho = C_{\rm ovlp}C_{\rm cov}h$;
4: Set $H_1 = 6h$, $H_2 = H_1$, $H_3 = H_1/2$, $H_4 = 2H_1$, $H_5 = H_1$;
5: Obtain the ordered sets using Algorithm 5;
6: Modify the ordered sets in case of intersections by calling Algorithm 6;
7: Compute the parametric cubic spline interpolation to each modified ordered set;
```

Algorithm 7 Fault curve reconstruction.

7 Experimental results

In this section, the results of some experiments are given. The efficiency of the method is confirmed by testing it on various kinds of problems: problems with multiple faults of the same type (ordinary or gradient) and problems with faults of different types with or without intersections. The initial set X is assumed to be a sequence of N random points with uniform distribution on a square domain $\Omega \subset \mathbb{R}^2$. We use $N = 10000$ random points (Fig. 5 left) in our test examples unless otherwise stated. For instance, in some examples, Halton points or a set of varying density random points are also used. The constant-generated weight function (8) is applied for both PU subroutines.

For the final curve reconstruction, we use the `csaps` function of MATLAB with smoothing parameter $p = 0.9999$ to obtain a cubic spline smoothing function on ordered fault points. The case $p = 1$ works as well but we choose a little bit smaller value to have smoother fits. In Example 7.7, we use $p = 1$ to better capture higher curvatures of the solution at fins and flukes of the dolphin.

We suppose that the type of faults (ordinary or gradient) is not known in advance. Thus, both gradient and Laplace indicators are used by default for all examples.

In Example 7.1, we compute the root mean distance of the detected points from a fine set of points on the real fault to measure the closeness of detected points to the exact fault or to measure the error of the final fault reconstruction. Assume that Z is

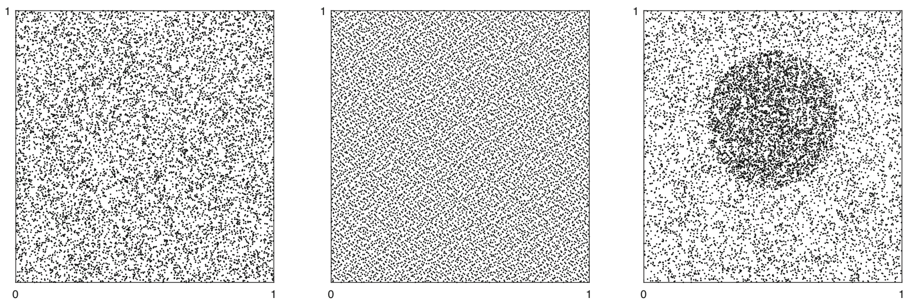

Fig. 5 A set of $N = 10000$ random points with uniform distribution (left), Halton points (middle), and random points with varying density (right)

a set of m points on the exact fault Γ and F is the set of detected points around Γ. We define the root mean distance $\operatorname{dist}(F, Z)$ by [12]

$$\operatorname{dist}(F, Z) := \sqrt{\frac{\sum_{x\in F}(\min_{z\in Z}\|x-z\|_2^2)}{|F|}}, \tag{16}$$

where $|F|$ stands for cardinality of F. In the case of multiple faults, we measure the error for each fault separately. In experiments, we assume that Z is a set of $m = 500$ points on each individual fault.

All algorithms are implemented in MATLAB and executed on a machine with an Intel Core i7 processor, 4.00 GHz, and 16 GB RAM. The code is freely available at GitHub via https://github.com/ddmirzaei/FaultDetection to facilitate the reproduction of the examples presented in this section. The connection between scripts in the GitHub repository and the pseudocodes in the paper is as follows.

mfile	Pseudocode
`FaultDetection.m`	Algorithm 1
`RBF_PU.m`	Algorithm 2
`LagrangeMat.m`	Algorithm 3
`PCR_PU.m`	Algorithm 4
`OrderedSubset.m`	Algorithm 5
`FixingIntersections.m`	Algorithm 6
`RunExample.m`	Algorithm 7

Other subroutines in the repository are called in the above MATLAB functions.

7.1 Example 1

Consider the test function [12]

$$f(x, y) := \begin{cases} |x - 0.4 - 0.1\sin(2\pi y)|, & \text{if } \ x \leqslant 0.7 + 0.2\sin(2\pi y) \\ |x - 0.4 - 0.1\sin(2\pi y)| - 0.2, & \text{otherwise,} \end{cases}$$

where $(x, y) \in [0, 1]^2 =: \Omega$. As is distinguishable from the upper-left side of Fig. 6, this function has an ordinary and a gradient fault. Faults are exactly represented by

$$\begin{aligned} \Gamma_1 &= \{(x, y) \in [0, 1]^2 : x - 0.7 - 0.2\sin(2\pi y) = 0\}, \quad \text{(ordinary fault)} \\ \Gamma_2 &= \{(x, y) \in [0, 1]^2 : x - 0.4 - 0.1\sin(2\pi y) = 0\}. \quad \text{(gradient fault)} \end{aligned}$$

The upper-right panel of Fig. 6 shows the clouds of faults points which are detected by the algorithm out of $N = 10000$ random points in Ω. The lower-left panel shows the narrowed points and the ordered points. Finally, at the lower-right panel, the exact and reconstructed curves using smooth spline interpolation are depicted. We also test the algorithm on variable density random points (Fig. 5 right). The detected fault points and the final reconstructed curves are shown in Fig. 7. The algorithm handles

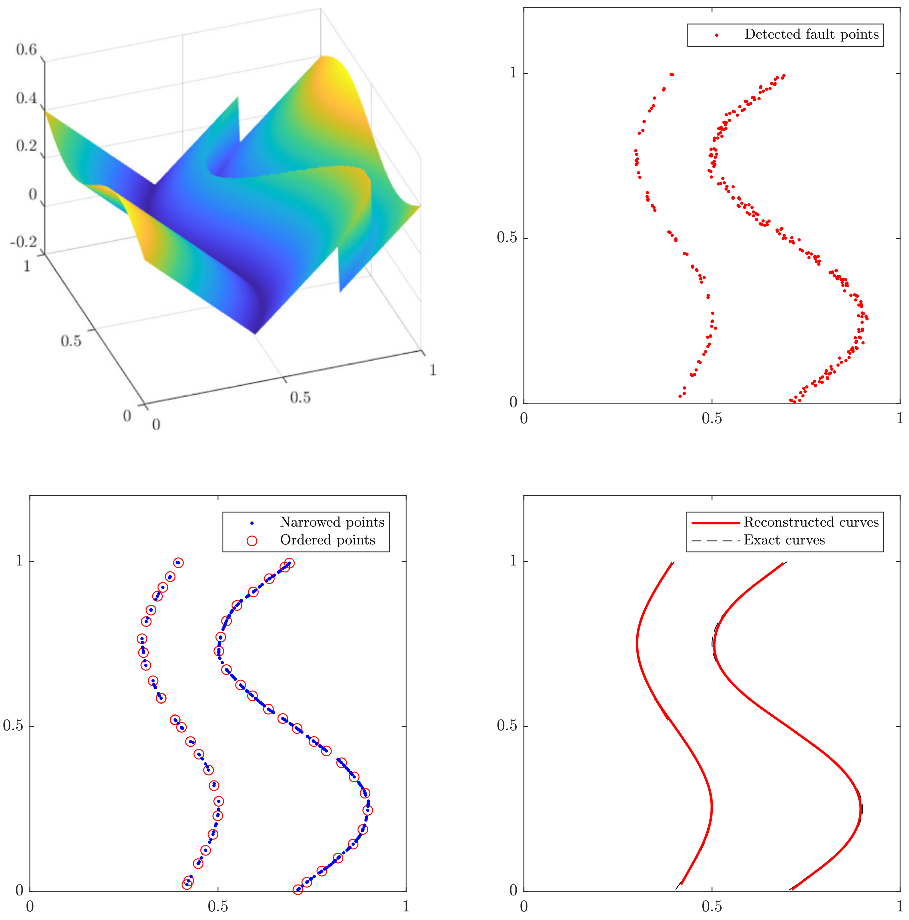

Fig. 6 Example 7.1: the test function (upper left), primary detected clouds (upper right), narrowed and ordered points (lower left), and reconstructed and exact curves (lower right). The primary set consists of $N = 10000$ uniformly distributed random points

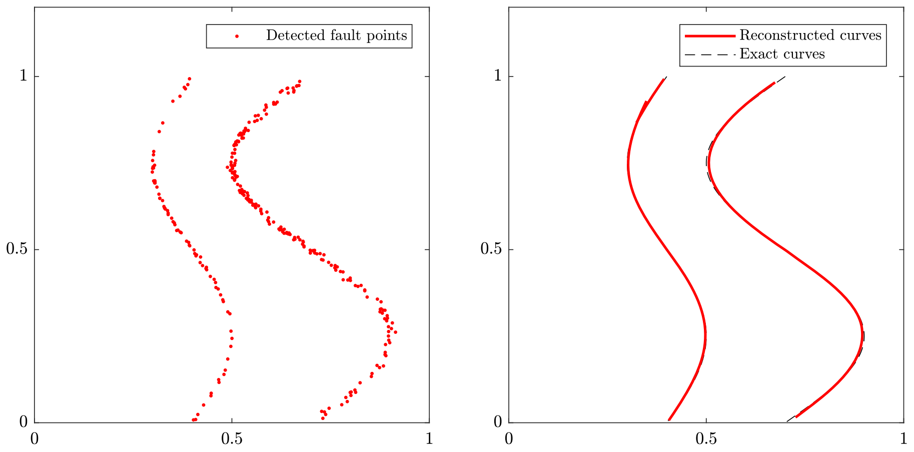

Fig. 7 Example 7.1: detected points out of a set of 10000 scattered points with varying density (left) and reconstructed and exact curves (right)

this case perfectly because patch sizes are selected automatically. The only difference is that we use more patch centers in that part of the region with a higher point density.

To verify the accuracy of the method, we first discretize the exact ordinary and gradient faults either by Z, a set of $m = 500$ equidistant points. Then we compute $\mathrm{dist}(F, Z)$ where F is the set of primary detected fault points around the fault curve. In fact, $\mathrm{dist}(F, Z)$ measures the closeness of the detected points to the exact fault. We also assume that Z' is a set of $m = 500$ equidistant points on the final reconstructed curve (after narrowing and curve fitting) and measure the root mean distance $\mathrm{dist}(Z', Z)$ to estimate the error of the reconstruction. Results are given in Table 1 for both ordinary and gradient faults for $N = 5000, 10000, 20000$ random points with uniform distributions on $[0, 1]^2$. Results are comparable with [12] but seem better than the grid-based algorithm of [11].

Finally, we note that the errors for $N = 10000$ varying density points (Fig. 5 right) are $\mathrm{dist}(F, Z) = 5.2\mathrm{e}-3$ and $\mathrm{dist}(Z', Z) = 3.5\mathrm{e}-3$ for the ordinary fault and $\mathrm{dist}(F, Z) = 2.6\mathrm{e}-3$ and $\mathrm{dist}(Z', Z) = 1.9\mathrm{e}-3$ for the gradient fault.

Table 1 The root mean distance between detected fault points F and points Z on the exact fault, and the distance between Z' and Z where Z' is a set of fine points on the reconstructed curve

	Ordinary		Gradient	
N	$\mathrm{dist}(F, Z)$	$\mathrm{dist}(Z', Z)$	$\mathrm{dist}(F, Z)$	$\mathrm{dist}(Z', Z)$
5000	7.9e − 3	4.5e − 3	7.3e − 3	3.4e − 3
10000	6.7e − 3	2.6e − 3	3.2e − 3	1.4e − 3
20000	4.1e − 3	1.8e − 3	1.5e − 3	1.4e − 3

Initial sets are random points with uniform distribution in $[0, 1]^2$. The case $N = 10000$ is shown on the left side of Fig. 5

7.2 Example 2

Consider the test function [11]

$$f(x,y) := \begin{cases} x^2 + x + \sin(2y), & \text{if } \ y \geqslant (5/3)x^2 - 11/3x + 2.2, \\ 0, & \text{if } \ y < (5/3)x^2 - 11/3x + 2.2 \text{ and } y \leqslant x + 0.01, \\ (x+y)y^2 - 1.2, & \text{otherwise,} \end{cases}$$

where $(x, y) \in [0, 1]^2$. This function has two ordinary faults that intersect each other at a point inside the domain (see the upper left-hand side of Fig. 8). Faults are exactly represented as

$$\begin{aligned} \Gamma_1 &= \{(x, y) \in [0, 1]^2 : y - (5/3)x^2 + 11/3x - 2.2 = 0\}, \\ \Gamma_2 &= \{(x, y) \in [0, 1]^2 : y - x - 0.01 = 0, \quad y \leqslant (5/3)x^2 - 11/3x + 2.2\}. \end{aligned}$$

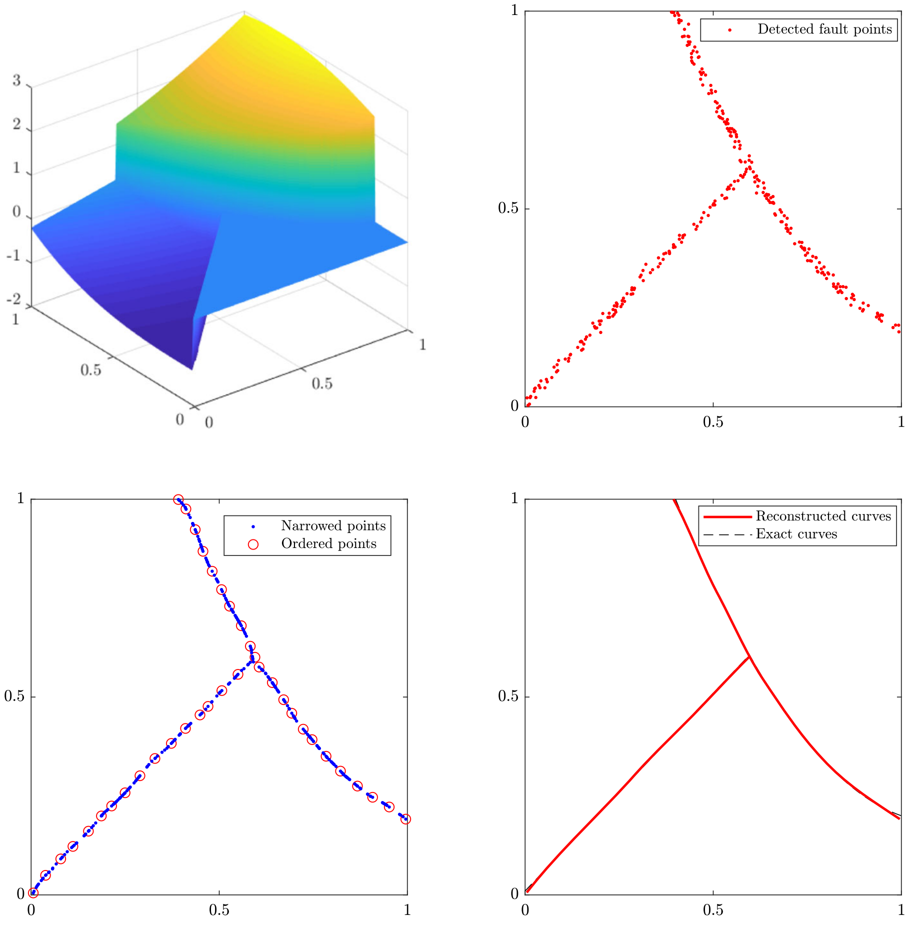

Fig. 8 Example 7.2: the test function (upper left), primary detected clouds (upper right), narrowed and ordered points (lower left), reconstructed and exact curves (lower right)

The upper-right panel of Fig. 8 shows the clouds of fault points detected by the algorithm. The lower-left panel shows the narrowed and the ordered points. Finally, at the lower-right panel, the exact and reconstructed curves using smooth spline interpolation are plotted.

7.3 Example 3

Consider the test function

$$f(x, y) := \begin{cases} 1 + 2\lfloor 7\sqrt{x^2 + y^2} \rfloor, & \text{if } (x - 0.5)^2 + (y - 0.5)^2 < 0.16, \\ 0, & \text{otherwise,} \end{cases}$$

where $(x, y) \in [0, 1]^2$. As shown in the upper-left side of Fig. 9, this function is discontinuous across six curves. In the upper-right side of Fig. 9, the clouds of

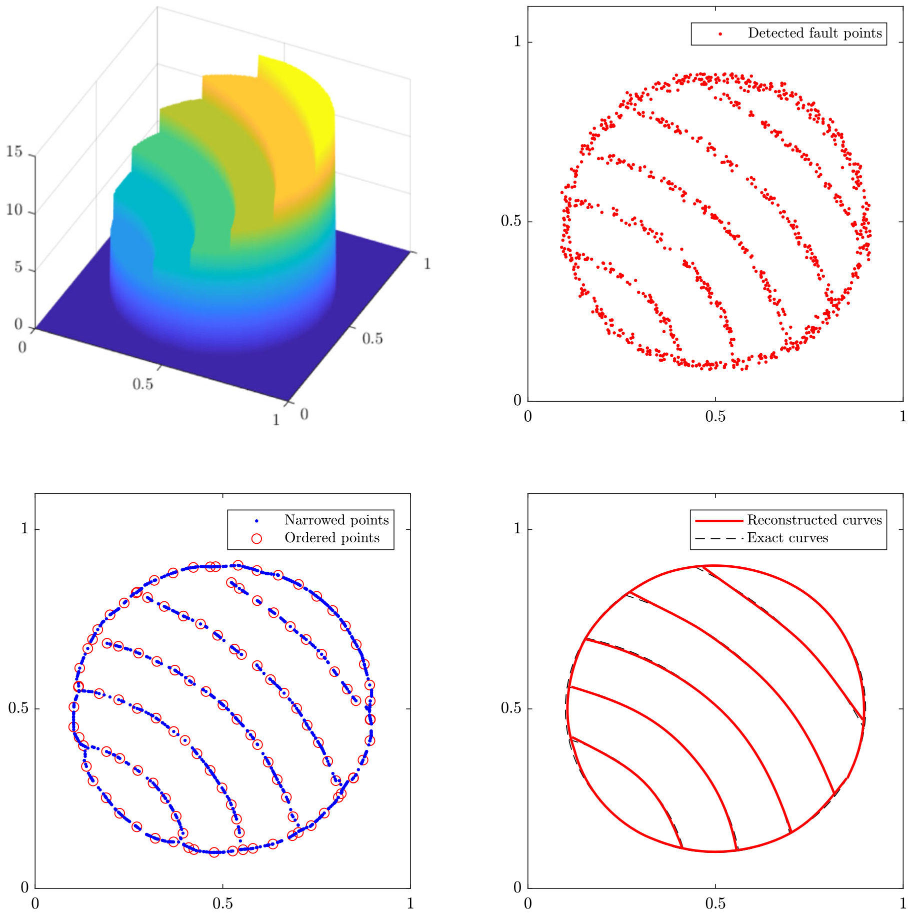

Fig. 9 Example 7.3: the test function (upper left), primary detected clouds (upper right), narrowed and ordered points (lower left), reconstructed and exact curves (lower right)

fault points detected by the algorithm are shown. Narrowed and ordered points are depicted in the lower-left panel while exact and reconstructed curves are shown at the lower-right panel.

7.4 Example 4

In this example, we consider the function [12]

$$f(x, y) = |x - 0.2 - 0.1\sin(2\pi(y - 0.13))| + |x^2 + y^2 - 0.5| - |(x - 1)^2 + y^2 - 0.36|$$

on $[0, 1]^2$ (see Fig. 10). This function has three gradient faults where one of them intersects two others. The given algorithm detects all fault curves and handles the

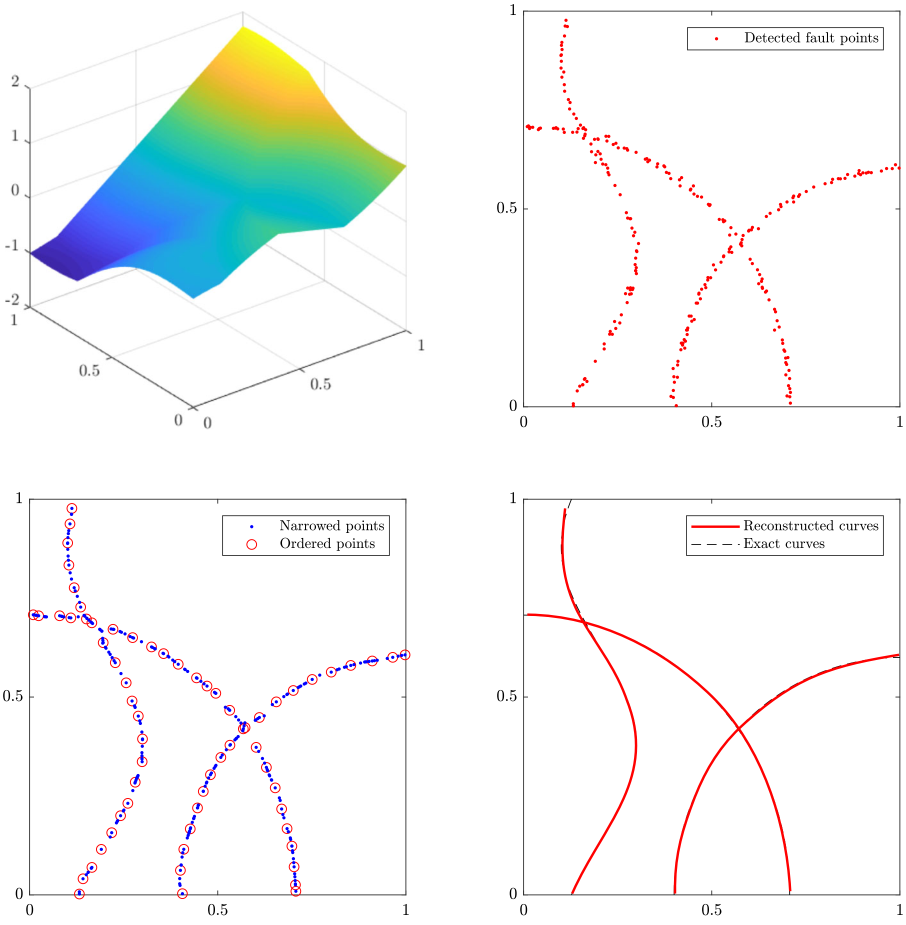

Fig. 10 Example 7.4: the test function (upper left), primary detected clouds (upper right), narrowed points ordered points (lower left), reconstructed and exact curves (lower right)

intersections. For this example, we use the value $C_M = 1$ instead of $1/4$. The primary detected points, the narrowed cloud, the ordered points, and the exact and reconstructed curves are all depicted in Fig. 10.

7.5 Example 5

Consider the test function [12]

$$f(x, y) := \begin{cases} |x^2 + y^2 - 1/2| - x + y^2, & \text{if } x - 0.4 - 0.1\sin(2\pi y) \geq 0 \\ |x^2 + y^2 - 1/2| - x + y^2 + 0.3, & \text{otherwise,} \end{cases}$$

where $(x, y) \in [0, 1]^2$ (see Fig. 11). This function has a gradient and an ordinary faults which intersect to each other with a small angle. In Fig. 11, primary detected clouds, narrowed and ordered points, and exact and reconstructed curves are shown.

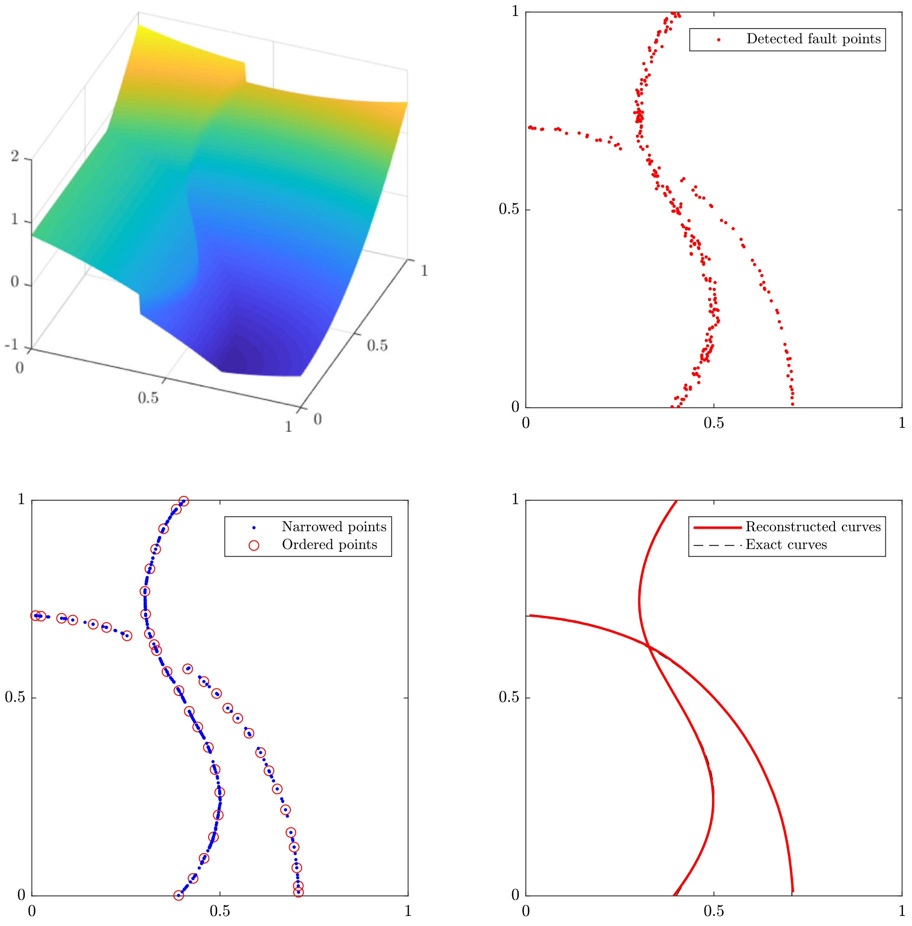

Fig. 11 Example 7.5: the test function (upper left), primary detected clouds (upper right), narrowed and ordered points (lower left), reconstructed and exact curves (lower right)

7.6 Example 6

In this example, we have a tangential intersection. Consider the test function [12]

$$f(x,y) := \begin{cases} \sqrt{4-x^2-y^2}-\sqrt{7/2}, & \text{if } \ x^2+y^2 \le 1/2 \\ \sqrt{4-(x-1)^2-(y-1)^2-\sqrt{7/2}}, & \text{if } \ (x-1)^2+(y-1)^2 \le 1/2 \\ 0, & \text{otherwise,} \end{cases}$$

where $(x, y) \in [0, 1]^2$ (see Fig. 12). This function has two gradient faults which are tangent to each other at the middle of the square. In the upper-right side of Fig. 12,

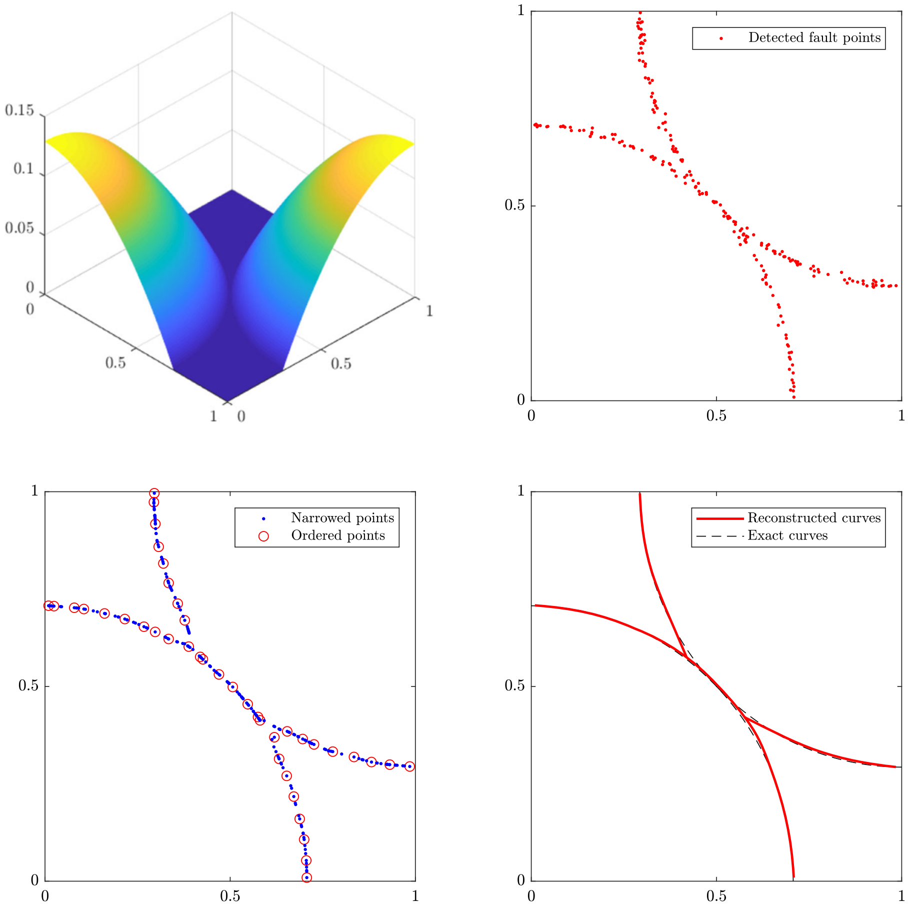

Fig. 12 Example 7.6: the test function (upper left), primary detected clouds (upper right), narrowed and ordered points (lower left), reconstructed and exact curves (lower right)

the clouds of fault points detected by the algorithm are shown. The detected points around the intersection point are not separable for each fault; thus, the algorithm determines two intersection points and a common fault between them (see the narrowed and ordered points on the lower-left panel and the exact and reconstructed curves on the lower-right panel of Fig. 12).

7.7 Example 7

As a toy example, using closed parametric curve $C(t) = x(t)i + y(t)j$ for $t \in [0, 2\pi)$ which represents a plane shape of a *dolphin*, we construct the test function

$$f(x, y) := \begin{cases} 1, & \text{if } \ (x, y) \in \text{interior of } C \\ 0, & \text{otherwise,} \end{cases}$$

for $(x, y) \in [0, 1]^2$. Obviously, this function is discontinuous on C. We use $N = 30000$ Halton points as an initial set X. Results are given in Fig. 13 where the narrowed fault points and the exact and reconstructed curves are illustrated.

8 An application for solving conservation laws

In this section, an application of the presented fault detection method is expressed in the process of solving conservation law equations via the weighted essentially non-oscillatory (WENO) finite volume methods (FVM). A brief summary of solution of conservation law equations by WENO FVM is outlined here. The reader is refereed to [1, 10, 25, 30, 42, 47, 54] for a complete explanation.

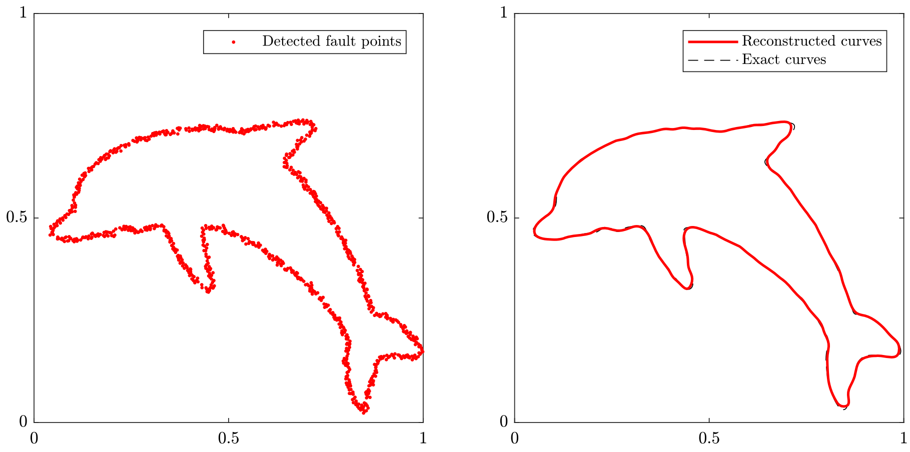

Fig. 13 Example 7.7: narrowed points (right), reconstructed and exact curves (right)

8.1 Spatial disctretization

Consider the following problem of scalar conservation law

$$\frac{\partial u}{\partial t} + \nabla \cdot F(u) = 0, \qquad u\,(0, x) = u_0(x), \tag{17}$$

where $u \equiv u\,(t, x) : I \times \Omega \longrightarrow \mathbb{R}$ is the solution of the problem. $\Omega \subset \mathbb{R}^2$ is an open and bounded computational domain and $I := \left(0, t_f\right]$ is a time interval with final time t_f. $F\,(u) := (f_1\,(u)\,, f_2\,(u))^T$ is called flux function. In order to discretize the problem (17) via FVM, a conforming triangulation $\mathcal{T} = \{T\}_{T \in \mathcal{T}}$ is considered on Ω where T is a triangle (control volume) in this triangulation. The integral form of (17) on each triangle $T \in \mathcal{T}$ at time $t \in I$ is obtained as

$$\frac{d}{dt}\overline{u}_T + \frac{1}{|T|}\int_{\partial T} F(u) \cdot n ds = 0, \qquad \text{for } \; T \in \mathcal{T} \tag{18}$$

where

$$\overline{u}_T \equiv \overline{u}_T(t) := \frac{1}{|T|}\int_T u(t, x)dx \qquad \text{for } \; T \in \mathcal{T}, \; t \in I$$

is the cell average value of u on triangle T at time t. Here, the boundary of T is denoted by ∂T and consists of union of Γ_j for $j = 1, 2, 3$ where Γ_j are edges of triangle T with unit outward normal vectors n_j. Thus (18) can be written as

$$\frac{d}{dt}\overline{u}_T + \frac{1}{|T|}\sum_{j=1}^{3}\int_{\Gamma_j} F(u(t, s)) \cdot n_j \, ds = 0.$$

The line integrals in the above equation can be approximated by a N_G-point Gaussian integration formula as

$$\frac{d}{dt}\overline{u}_T + \frac{1}{|T|}\sum_{j=1}^{3}|\Gamma_j|\sum_{\ell=1}^{N_G}\omega_\ell^{(j)} F(u(t, x_\ell^{(j)})) \cdot n_j = 0$$

where $\omega_\ell^{(j)}$ are Gaussian weights and $x_\ell^{(j)}$ are Gaussian points on edge Γ_j of triangle T. The Lax-Friedrichs numerical flux

$$F(u) \cdot n \approx \widetilde{F}(u_{\text{in}}, u_{\text{out}}; n) = \frac{1}{2}\,(F(u_{\text{in}}) + F(u_{\text{out}})) \cdot n + \frac{\sigma}{2}(u_{\text{in}} - u_{\text{out}}) \tag{19}$$

is used here where $u_{\text{in}}(\cdot, x_\ell^{(j)})$ is the approximate solution at Gaussian point $x_\ell^{(j)}$ of triangle T itself and $u_{\text{out}}(\cdot, x_\ell^{(j)})$ is the approximate solution at the same point but from an adjacent triangle which shares Γ_j as a common edge with T. The coefficient σ is obtained by

$$\sigma = \max_{\min(u_{\text{in}}, u_{\text{out}}) \le u \le \max(u_{\text{in}}, u_{\text{out}})} |F'(u) \cdot n_j|, \qquad F'(u) = \left[\frac{df_1}{du}, \frac{df_2}{du}\right].$$

Therefore, cell average values $\{\overline{u}_T(t)\}_{T\in\mathcal{T}}$ can be updated as

$$\frac{d}{dt}\overline{u}_T(t) = \mathcal{L}_T(\overline{u}_T(t)) \quad \text{for} \quad T \in \mathcal{T}, \tag{20}$$

where

$$\mathcal{L}_T(\overline{u}_T(t)) = -\frac{1}{|T|}\sum_{j=1}^{3}|\Gamma_j|\sum_{\ell=1}^{N_G}\omega_\ell^{(j)}\widetilde{F}(u_{\text{in}}(t,x_\ell^{(j)}), u_{\text{out}}(t,x_\ell^{(j)}); n_j). \tag{21}$$

It is necessary to reconstruct u_{in} and u_{out} from the current cell average values $\{\overline{u}_T(t)\}_{T\in\mathcal{T}}$. There are different methods for reconstruction that will be explained later.

8.2 Time discretization

For hyperbolic equations, an ODE solver which maintains the stability of the problem and avoids oscillations should be employed. Here we use a strong stability preserving Runge-Kutta (SSPRK) methods of order 3 [7, 26, 48]. Consider the time depended system (20). For moving from data $\{\overline{u}_T(t^n)\}_{T\in\mathcal{T}}$ at time t^n to data $\{\overline{u}_T(t^{n+1})\}_{T\in\mathcal{T}}$ with time length Δt, the SSPRK3 method consists of three steps

$$\begin{aligned}
\overline{u}_T^{(1)} &= \overline{u}_T(t^n) + \Delta t\mathcal{L}_T(\overline{u}_T(t^n)),\\
\overline{u}_T^{(2)} &= \frac{3}{4}\overline{u}_T(t^n) + \frac{1}{4}\overline{u}_T^{(1)} + \frac{1}{4}\Delta t\mathcal{L}_T(\overline{u}_T^{(1)}),\\
\overline{u}_T(t^{n+1}) &= \frac{1}{3}\overline{u}_T(t^n) + \frac{2}{3}\overline{u}_T^{(2)} + \frac{2}{3}\Delta t\mathcal{L}_T(\overline{u}_T^{(2)}).
\end{aligned} \tag{22}$$

By applying the CFL condition, Δt is restricted as

$$\Delta t \le \min_{T\in\mathcal{T}}\frac{r_T}{\eta_t^{\max}}$$

where r_T is the radius of the incircle of triangle T and $\eta_t^{\max} = \max|F'(u)\cdot n|$, where maximum is being taken over all Gaussian points on edges of triangle T.

8.3 Reconstruction step

As discussed before, the approximation values u_{in} and u_{out} in (21) should be reconstructed from the cell average values $\{\overline{u}_T(t)\}_{T\in\mathcal{T}}$ in each time step. An efficient particle reconstruction scheme based on polyharmonic spline interpolation is described [32]. For some other reconstruction methods, see, for example, [1, 42].

Assume that $\{x_{c_T}\}_{T\in\mathcal{T}}$ is the set of barycenters of triangles in $\mathcal{T}$. For each reference triangle $T \in \mathcal{T}$, consider a stencil $\mathcal{S} = \{T_1, \ldots, T_n\} \subset \mathcal{T}$, where $T \in \mathcal{S}$. In each triangle $R \in \mathcal{S}$, the cell average value $\overline{u}_R(t)$ is considered as an approximation for u at x_{c_R} at time t. So we are looking for a function s that interpolates u from the given values $u(x_{c_R}, t) \approx \overline{u}_R(t)$ for $R \in \mathcal{S}$, i.e.,

$$s(x_{c_R}) = \overline{u}_R \qquad \text{for all } R \in \mathcal{S}.$$

In polyharmonic spline interpolation, s is written as

$$s(x) = \sum_{R \in \mathcal{S}} \alpha_R \phi(x - x_{c_R}) + \sum_{k=1}^{Q} a_k p_k(x),$$

and by imposing the interpolation condition, the same system as (2) is resulted where f is replaced by the vector of cell average values at stencil $\mathcal{S}$.

In order to avoid the nonphysical oscillations in solution, the WENO reconstruction is frequently used in literature [25, 30, 38, 54]. In WENO schemes, a weighted average of reconstructions from a set of stencils $\{\mathcal{S}_k\}_{k=1}^K$ for which $T \in \mathcal{S}_k$ for all $k = 1, \ldots, K$ is used. The weights are chosen in such way that the oscillations are minimized. We use a WENO reconstruction with an oscillation indicator parameter based on native space norm of the underlying polyharmonic kernel which is fully described in [32]. Details are left and the reader is referred to original sources.

8.4 Combination with the fault detection algorithm

In a WENO reconstruction, the process of selecting stencils $\{\mathcal{S}_k\}_{k=1}^K$ for a $T \in \mathcal{T}$ is an important part [22, 25, 29, 38]. In polyharmonic kernel reconstruction, the size of each stencil $\mathcal{S}_k$ (the number of triangles in stencil $\mathcal{S}_k$) should not be less than Q, the dimension of polynomial space. Three types of stencils for each triangle T are introduced in [1]; centered, forward sector, and backward sector stencils. Three size 7 stencils of each type are displayed in Fig. 14.

It was concluded in [1] that in WENO polyharmonic spline reconstruction with $\phi = \|\cdot\|_2^2 \log(\|\cdot\|_2)$, the use of 7 stencils (1 centered, 3 forward, and 3 backward) all of size 4 is sufficient for smooth solutions, while for solutions with discontinuity or a steep gradient at least 7 stencils (1 centered, 3 forward, and 3 backward), all of size 7 are required. For smooth solutions, we can even get rid of WENO and use a simple central stencil approximation instead. Usually the solution of a conservation law problem has a steep gradient or becomes discontinuous on a curve or a small subregion of global domain Ω. Thus, it is reasonable to detect such fault curves or regions in advance and use higher number of stencils or higher sizes in that regions only.

To follow this strategy, at each time step, $X = \{x_{c_T}\}_{T \in \mathcal{T}}$ is considered as a set of scattered points in Ω and $\{\overline{u}_T(t)\}_{T \in \mathcal{T}}$ as an approximation for solution values at these scattered points. Then, using the proposed fault detection algorithm with a Laplace indicator, a set of fault barycenters $F = F(X, \Delta, \delta_2)$ are detected where δ_2 is defined in (11). As we discussed before, $F(X, \Delta, \delta_2)$ contains points close to both ordinary and gradient discontinuities. A triangle with its barycenter belongs to F is marked as a fault triangle. We use 7 stencils of size 7 for a fault triangle and a

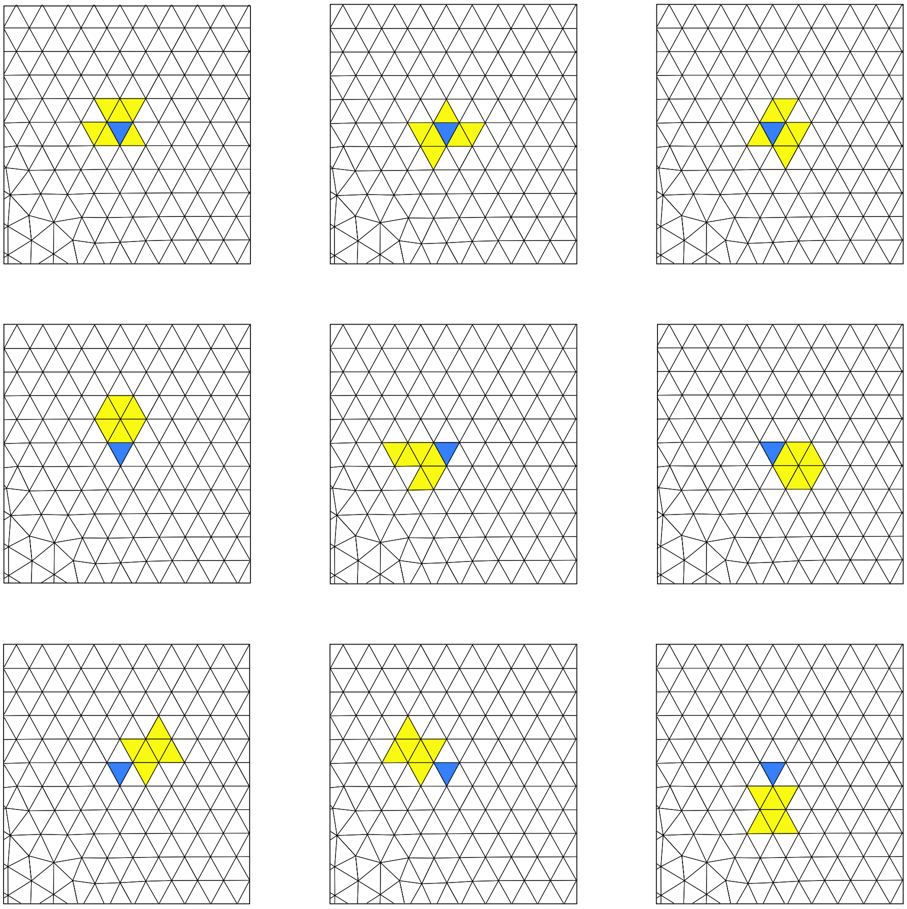

Fig. 14 Centered stencils (up), forward sector stencils (middle), and backward sector stencils (down)

central stencil of size 7 otherwise. This means that the WENO reconstruction is used for that parts of the domain on which the solution has steep gradient and is going to be discontinuous. Other parts are subjected to a simple central stencil reconstruction. The algorithm of this hybrid method is as follows.

Numerical experiments show that this approach leads to an speedup of around 30% while retains the accuracy of solution compared to the original WENO reconstruction (see the next section for an example).

8.5 A numerical example

Consider the nonlinear Burger's equation

$$u_t + uu_{x_1} + uu_{x_2} = 0, \qquad x = (x_1, x_2) \in [-0.5, 0.5]^2$$

```
Data: Time level $t$, stepsize $\Delta t$, triangulation $\mathcal{T}$, barycenters $X = \{x_{c_T}\}_{T\in\mathcal{T}}$,
      current average values $\{\overline{u}_T(t)\}_{T\in\mathcal{T}}$.
Result: $\{\overline{u}_T(t+\Delta t)\}_{T\in\mathcal{T}}$
1: Find the fault barycenters $F = F(X, \Delta, \delta_2)$ using the fault detection Algorithm 1;
2: Assign 7 stencils of size 7 for reconstructing $u$ at edges of triangles with barycenters in $F$;
3: Assign 1 central stencil of size 7 for reconstructing $u$ at edges of triangles with barycenters in $X \setminus F$;
4: for $T \in \mathcal{T}$ do
     Reconstruct $u_{\text{in}}$ and $u_{\text{out}}$ at time $t$ on Gaussian points on edges on $T$ using PHS interpolation;
     Compute numerical fluxes along edges of triangle $T$ by (19);
     Compute $\mathcal{L}_T(\overline{u}_T(t))$ using (21);
     Update the solution via (22) and obtain $\overline{u}_T(t+\Delta t)$;
5: Return $\{\overline{u}_T(t+\Delta t)\}_{T\in\mathcal{T}}$;
```

Algorithm 8 Combination of WENO-FVM and fault triangle detection algorithm.

where $u := u(t,x) : [0,1]\times[-0.5,0.5]^2 \longrightarrow \mathbb{R}$, with initial condition

$$u(0,x) = \begin{cases} \exp\left(\frac{\|x-c_0\|_2^2}{\|x-c_0\|_2^2 - r_0^2}\right), & \text{if } \|x-c_0\|_2 < r_0, \\ 0, & \text{otherwise,} \end{cases} \tag{23}$$

where $r_0 = 0.15$, $c_0 = (-0.2, -0.2)$ and with periodic boundary conditions. The initial condition is shown in Fig. 15. The solution evolves into a very steep gradient when advancing in time.

The MATLAB code of this part is also freely available at GitHub repository https://github.com/ddmirzaei/FaultDetection_Application to facilitate the reproduction of the results. All constants and parameters of the fault detection algorithm are set as before except parameter C_M which is set to be 1 in this experiment. The previous value $C_M = 1/4$ is also works but we use $C_M = 1$ to reduce the number of fault triangles, slightly. Since the exact solution is not available, in Table 2, the errors between the solution of the full WENO reconstruction and the solution of the hybrid method for different values of triangles sizes $h_T = \{\frac{1}{16}, \frac{1}{32}, \frac{1}{64}, \frac{1}{128}\}$ are given. We observe a good agreement between the solutions of both methods which means that the hybrid method retains the accuracy of the full WENO reconstruction method. The next two columns of the table contain the total number of triangles and the average number of fault triangles in all time steps. For this example, and for mesh-sizes $h_T = 1/64, 1/128$, we observe that 4–5% of all triangles are detected as fault triangles. The total run times are given in the last two columns. We observe speedup of about 30% with the new hybrid method.

For a better illustration, numerical solutions using the hybrid method at time levels $t = 0, 0.3$, $t = 0.5$, and $t = 1$ are shown in Fig. 15, and the fault barycenters at time $t = 1$ are shown in Fig. 16.

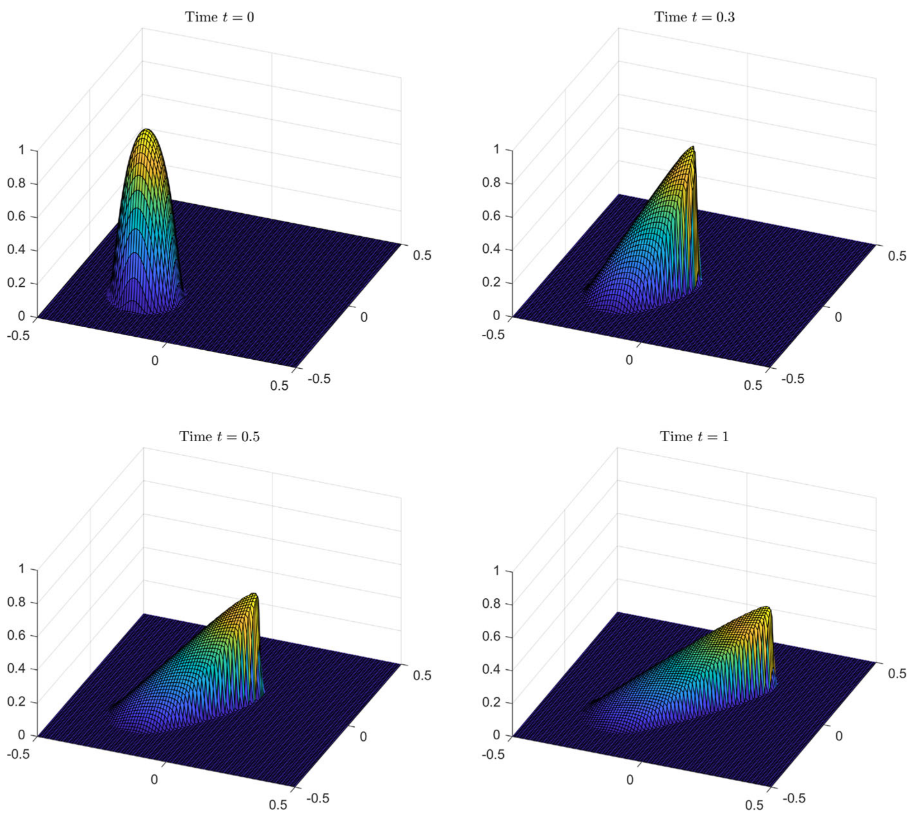

Fig. 15 Numerical solutions of Burger's equation at time levels $t = 0$, $t = 0.3$, $t = 0.5$, and $t = 1$ using the hybrid method

9 Conclusion

A localized scattered data approximation method based on polyharmonic spline interpolation in combination with partition of unity method was proposed to define a gradient and a Laplacian based indicators for detecting a cloud of fault points on or close to discontinuities of a bivariate function. The polyharmonic interpolation was

Table 2 Root mean square errors (RMSE) between the full WENO and the hybrid methods, number of all and detected triangles in the hybrid method, and the comparison of CPU times (in seconds)

h_T	RMSE	N_{total}	N_{fault}	CPU time (WENO)	CPU time (hybrid)
$\frac{1}{16}$	4.37e − 3	778	58	18	15
$\frac{1}{32}$	3.20e − 3	2334	162	132	102
$\frac{1}{64}$	1.06e − 3	9388	492	1001	717
$\frac{1}{128}$	4.05e − 4	37666	1395	7771	5576

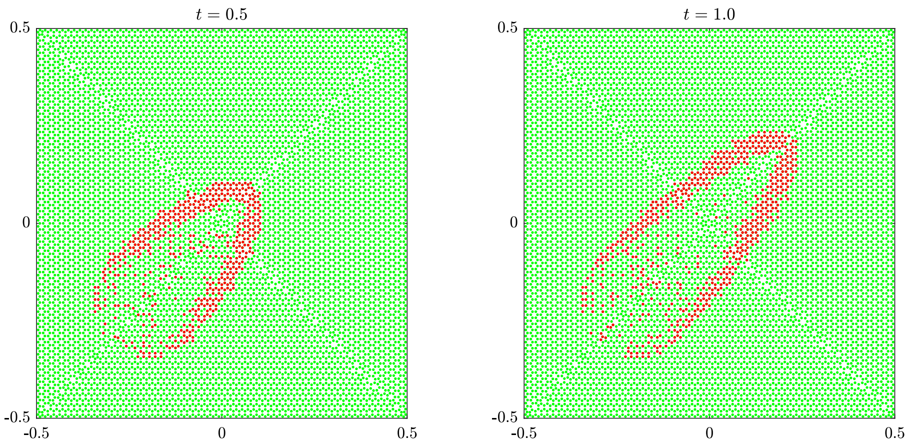

Fig. 16 Fault barycenters (red dots) at time levels $t = 0.5$ and $t = 1$ for $h_T = 1/64$

done on scaled data points to prevent the instability of kernel matrices. To get an accurate reconstruction of the fault, a localized principal component regression was applied to generate a second set of points which are supposed to be closer to the fault curve than the primary detected set. Then an ordered subset of these narrowed points was extracted and a smooth parametric spline interpolation was employed to reconstruct the fault curve. Situations with multiple fault curves and special cases with intersections and multi-branch configurations were addressed. Finally, an application for solving conservation law PDEs was given.

Applications to other areas such as image processing and geosciences, and generalization to 3-variate functions are left for a future study.

Acknowledgements We wish to express our deep gratitude to anonymous reviewers for their helpful comments which improved the quality of the paper.

Funding Open access funding provided by Uppsala University. The work of the second author (Navid Soodbakhsh) was supported by a grant from Iranian National Science Foundation (INSF), No. 98001906.

Data availability The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Aboiyar, T., Georgoulis, E.H., Iske, A.: Adaptive ADER methods using kernel-based polyharmonic spline WENO reconstruction. SIAM J. Sci. Comput. **32**, 3251–3277 (2010)
2. Allasia, G., Besenghi, R., Cavoretto, R.: Adaptive detection and approximation of unknown surface discontinuities from scattered data. Simul. Model. Pract. Theory **17**, 1059–1070 (2009)
3. Arandiga, F., Cohen, A., Donat, R., Dyn, N., Matei, B.: Approximation of piecewise smooth functions and images by edge-adapted (ENO-EA) nonlinear multiresolution techniques. Appl. Comput. Harmon. Anal. **24**(2), 225–250 (2008)
4. Arandiga, F., Cohen, A., Donat, R., Matei, B.: Edge detection insensitive to changes of illumination in the image. Image Vis. Comput. **28**(4), 553–552 (2010)
5. Archibald, R., Gelb, A., Yoon, J.: Polynomial fitting for edge detection in irregularly sampled signals and images. SIAM J. Numer. Anal. **43**(1), 259–279 (2005)
6. Arge, E., Floater, M.: Approximating scattered data with discontinuities. Numer. Algoritm. **8**, 149–166 (1994)
7. Barth, T.J., Deconinck, H.: High order methods for computational physics. Springer, Berlin (1999)
8. Besenghi, R., Allasia, G.: Scattered data near-interpolation with applications to discontinuous surfaces. In: Curve and Surface Fitting, pp. 75–84, Nashville, TN. Vanderbilt University Press (2000)
9. Bhardwaj, S., Mittal, A.: A survey on various edge detector techniques. Procedia Technol. **4**, 220–226 (2012)
10. Boscheri, W.: High order direct Arbitrary-Lagrangian–Eulerian (ALE) finite volume schemes for hyperbolic systems on unstructured meshes. SIAM J. Sci. Comput. **24**, 7510–801 (2017)
11. Bozzini, M., Rossini, M.: The detection and recovery of discontinuity curves from scattered data. J. Comput. Appl. Math. **240**, 148–162 (2013)
12. Bracco, C., Davydov, O., Giannelli, C., Sestini, A.: Fault and gradient fault detection and reconstruction from scattered data. Comput. Aided Geom. Des. **75**, 101786 (2019)
13. Cates, D., Gelb, A.: Detecting derivative discontinuity locations in piecewise continuous functions from fourier spectral data. Numer. Algoritm. **46**, 59–84 (2007)
14. Cavoretto, R., De Marchi, S., De Rossi, A., Santin, G.: Partition of unity interpolation using stable kernel-based techniques. Appl. Numer. Math. **116**, 95–107 (2017)
15. Cavoretto, R., De Rossi, A.: Adaptive meshless refinement schemes for RBF - PUM collocation. Appl. Math. Lett. **90**, 131–138 (2019)
16. Cavoretto, R., De Rossi, A., Perracchione, E.: Efficient computation of partition of unity interpolants through a block-based searching technique. Comput. Math. Appl. **71**, 2568–2584 (2016)
17. Cavoretto, R., De Rossi, A., Perracchione, E.: Optimal selection of local approximants in RBF-PU, interpolation. J. Sci. Comput. **74**, 1–22 (2018)
18. Ahmadi Darani, M.R.: The RBF partition of unity method for solving the Klein-Gordon equation. Engineering with Computers, In press (2020)
19. Davydov, O., Schaback, R.: Error bounds for kernel based numerical differentiation. Numer. Math. **132**, 243–269 (2016)
20. Davydov, O., Schaback, R.: Optimal stencils in Sobolev spaces. IMA J. Numer. Anal. **39**, 398–422 (2019)
21. Drake, K.P., Fuselier, E.J., Wright, G.B.: Implicit surface reconstruction with a curl-free radial basis function partition of unity method. SIAM J. Sci. Comput. **42**, A3018–A3040 (2022)
22. Dumbser, M., Kaser, M.: Arbitrary high order non-oscillatory finite volume schemes on unstructured meshes for linear hyperbolic systems. J. Comput. Phys. **221**, 693–723 (2007)
23. Farazandeh, E., Mirzaei, D.: A rational RBF interpolation with conditionally positive kernels, 47:74. Adv. Comput. Math. **47**, 74 (2021)
24. Fasshauer, G.E.: Meshfree Approximations Methods with Matlab. World Scientific, Singapore (2007)
25. Friedrich, O.: Weighted essentially non-oscillatory schemes for the interpolation of mean values on unstructured grids. J. Comput. Phys. **144**, 194–212 (1998)
26. Gottlieb, S., Shu, C.W.: Total variation diminishing Runge-Kutta schemes. Math. Comput. **67**, 73–85 (1998)
27. Gout, C., Le Guyader, C.: Segmentation of complex geophysical structures with well data. Computional Geosci. **10**, 361–372 (2006)
28. Gutzmer, T., Iske, A.: Detection of discontinuities in scattered data approximation. Numer. Algoritm. **16**, 155–170 (1997)

29. Harten, A., Chakravarthy, S.R.: Multidimensional ENO schemes for general geometries. Tech. Rep., ICASE **221**, 91–76 (1991)
30. Hu, C., Shu, C.W.: Weighted essentially non-oscillatory schemes on triangular meshes. J. Comput. Phys. **150**, 97–127 (1999)
31. Iske, A.: On the approximation order and numerical stability of local Lagrange interpolation by polyharmonic splines . In: International Series of Numerical Mathematics 145, pp. 153–165. Basel, Birkhäuser Verlag (2003)
32. Iske, A.: On the construction of kernel-based adaptive particle methods in numerical flow simulation. In: Notes on Numerical Fluid Mechanics and Multidisciplinary Design (NNFM), pp. 197–221, Berlin, Springer (2013)
33. Jabalameli, M., Mirzaei, D.: A weak-form RBF-generated finite difference method. Comput. Math. Appl. **79**, 2624–2643 (2020)
34. Jeffers, J.: Two case studies in the application of principal component. Appl. Stat. **16**, 225–236 (1967)
35. Jolliffe, I.T. Principal Component Analysis, 2nd. Springer, New York (2002)
36. Jung, J.H., Durante, V.R.: An iterative adaptive multiquadric radial basis function method for the detection of local jump discontinuities. Appl. Numer. Math. **59**(7), 1449–1466 (2009)
37. Jung, J.H., Gottlieb, S., Kim, S.O.: Iterative adaptive RBF methods for detection of edges in two-dimensional functions. Appl. Numer. Math. **61**(1), 77–91 (2011)
38. Kaser, M., Iske, A.: ADER schemes on adaptive triangular meshes for scalar conservation laws. J. Comput. Phys. **205**, 486–508 (2005)
39. Kendall, M.G.: A Course in Multivariate Analysis. Griffin, London (1957)
40. Larsson, E., Shcherbakov, V., Heryudono, A.: A least squares radial basis function partition of unity method for solving PDEs. SIAM J. Sci. Comput. **39**, A2538–A2563 (2017)
41. Lee, I.K.: Curve reconstruction from unorganized points. Comput. Aided Geom. Des. **17**, 161–177 (2000)
42. LeVeque, R.J.: Finite Volume Methods for Hyperbolic Problems. Cambridge University Press, Cambridge (2002)
43. Mirzaei, D.: The direct radial basis function partition of unity (d-RBF-PU) method for solving PDEs. SIAM J. Sci. Comput. **43**, A54–A83 (2021)
44. Romani, L., Rossini, M., Schenone, D.: Edge detection methods based on rbf interpolation. J. Comput. Appl. Math. **349**, 532–547 (2019)
45. Safdari-Vaighani, A., Heryudono, A., Larsson, E.: A radial basis function partition of unity collocation method for convection–diffusion equations arising in financial applications. J. Sci. Comput. **64**(2), 341–367 (2015)
46. Shankar, V.: The overlapped radial basis function-finite difference (RBF-FD) method: a generalization of RBF-FD. J. Comput. Phys. **342**, 211–228 (2017)
47. Shu, C.W.: High order ENO and WENO schemes for computational fluid dynamics. In: High Order Methods for Computational Physics, pp. 439–852, Berlin, Springer (1991)
48. Shu, C.W., Osher, S.: Efficient implementation of essentially non-oscillatory shock capturing schemes. J. Comput. Phys. **77**, 439–471 (1998)
49. Singh, S., Singh, R.: Comparison of various edge detection techniques. In: 2nd International Conference on Computing for Sustainable Global Development, pp. 393–396 (2015)
50. Sober, B., Levin, D.: Manifold approximation by moving least squares projection (MMLS). Constr. Approx. **52**, 433–478 (2020)
51. Strang, G.: Linear Algebra and Learning from Data. Wellesley-Cambridge Press, Cambridge (2019)
52. Wendland, H.: Fast evaluation of radial basis functions: methods based on partition of unity. In: Approximation Theory, X: Wavelets, Splines, and Applications, pp. 473–483. Nashville, TN, Vanderbilt University Press (2002)
53. Wendland, H.: Scattered Data Approximation. Cambridge University Press, Cambridge (2005)
54. Wolf, W.R., Azevedo, J.L.F.: High-order ENO and WENO schemes for unstructured grids. Int. J. Numer. Methods Fluids **55**, 917–943 (2007)
55. Yi, S., Labate, D., Easley, G.R., Krim, H.: A shearlet approach to edge analysis and detection. IEEE Trans. Image Process. **18**(5), 929–941 (2009)