Dolomites Research Notes on Approximation (DRNA)
Vol. 3, (2010), p. 7 – 38

ISSN: 2035-6803

Efficient approximation algorithms. Part I: approximation of unknown fault lines from scattered data

Giampietro Allasia, Renata Besenghi,
Roverto Cavoretto and Alessandra De Rossi
Department of Mathematics "G. Peano", University of Torino (Italy)
e-mail: {giampietro.allasia, renata.besenghi, roberto.cavoretto, alessandra.derossi}@unito.it

web: http://drna.di.univr.it/
email: drna@drna.di.univr.it

Abstract

A method for the detection of fault lines of a surface only known at scattered data is presented. In particular, we use a technique for the detection of fault points, picking out and collecting in a set all the data points close to fault lines. To select these points we use a cardinal radial basis interpolation formula. Then, applying a powerful refinement technique, we discuss different methods for accurate approximation of fault lines, considering procedures based on polygonal line, least squares, and best l_∞ approximations. Moreover, an adaptive method enables us to locally increase (step by step) the number of nodes to be dealt with in the detection process, reducing considerably the entire number of data points to be considered. Numerical results point out the efficiency of our approaches.

Keywords: surface discontinuities, fault lines, detection and approximation methods, radial basis functions, interpolation algorithms, scattered data.

AMS Subject classification[2010]: 65D05, 65D10, 65D17.

1 Introduction

We consider the problem of the detection and approximation of fault lines of a surface only known on a finite number of scattered data. Predicting the location of fault lines and obtaining accurate approximations of them enable the construction of constrained approximation models of the surface overcoming common problems such as over-smoothing.

Some applications require only a rough knowledge of faults, in the sense that their number, position, and behaviour are significant, but accurate approximations of fault lines are not so important. This may be the case, for example, in the oil industry, where fault detection provides useful information on the occurrence of oil reservoirs (see, e.g. [23, 20, 21]).

On the other hand, other applications call for accurate approximations of fault lines, as it happens in the reconstruction of discontinuous surfaces when images are required to be highly reliable. As an example, we may refer to medical images by magnetic resonance in which discontinuity lines may indicate the presence of some pathology (see, e.g. [30]). Also in some geophysical problems, a faithful representation of irregular surfaces by an accurate data fitting process is of great importance (see, e.g. [22]).

In the last two decades several authors have analyzed the fault detecting and approximating problem, as well as the discontinuous surface approximating problem, using different techniques and methodologies (see, for instance, in chronological order, [10, 31, 26, 15, 20, 23, 29, 16, 13, 3, 24, 14, 8, 19]). To get an idea one can see the given references, and, in particular, the monograph by Arcangéli, López de Silanes and Torrens [9].

Each method has its own performance and range of application, but all of them must unavoidably match the quantity of information, namely, the number of data and their distribution. Clearly, none of the existing methods is universally satisfactory, and understanding their limitations is important in order to apply them successfully.

In [3] the authors proposed a method to detect unknown fault lines, starting from a large set of points irregularly distributed in a plane region and the corresponding function values. This procedure is strictly connected with another one they proposed about surface approxima-

tion [14]. The latter requires the preliminary knowledge of fault points and reconstructs the discontinuous surface by means of a parallel algorithm.

Here, at first, we improve the technique of detection proposed in [3] (see [6]). Secondly, we improve the detection scheme, introducing an *adaptive detection process* (see [7]). Finally, we discuss different methods for accurate approximation of fault lines considering polygonal lines, least squares, and best l_∞ approximations [6]. However, we remark that we are here only interested in finding fault lines and not in approximation of discontinuous surfaces.

Hence, the resulting procedure can be divided into two phases:

1. detection of fault lines (to predict as precisely as possible the locations of the fault lines, possibly using the adaptive technique);

2. approximation of fault lines (to get accurate analytic representations of the fault lines).

For more details see [18].

The paper is organized as follows. In Section 2 the technique to detect fault lines discussed in [3] is recalled, adding explanations on the choice of the threshold value σ_0, a crucial parameter. As basic tools for detection of fault points, i.e. points close to faults, we use a reliable cell-based search method and a cardinal radial basis interpolant (see [1, 2]), that is, Shepard's formula possibly with suitable modifications. The output of this procedure is first a set of fault points and then a new set of points, called barycentres, generally closer to the faults than the fault points themselves. Then the barycentres are processed in various steps in order to handle complex fault situations. Section 3 is devoted to describe the adaptive scheme. In Section 4 we point out the different techniques of approximation, outlining in detail the various procedures. Section 5 contains several numerical results which show the effectiveness of our methods. The use of conventional approximation tools, if properly adapted to the topic, does not lead to instability phenomena or undesiderable oscillations which could locally or even globally hinder the approximation. Finally, in Section 6, we make some conclusions and remarks, giving an idea of possible developments.

2 Detection of fault lines

To characterize the data points on or close to the fault lines, named *fault points*, first we consider a procedure based on local data interpolation by cardinal radial basis interpolants (CRBIs), where the difference between any known function value and the related value of the interpolant is computed and compared with a threshold parameter. Then, the set of fault points must be further manipulated to obtain a new set of points, named *barycentres*, which supplies more information on the faults. Finally, after ordering the set of barycentres, we obtain the set of barycentres of barycentres, which can be further refined.

Definition 2.1. *Let $\mathcal{S}_n = \{x_k, k = 1, 2, \ldots, n\}$ be a set of distinct and scattered data points in a domain $D \subset \mathbb{R}^2$, and let $\mathcal{F}_n = \{f_k = f(x_k), k = 1, 2, \ldots, n\}$ be a set of corresponding values of an unknown function $f : D \to \mathbb{R}$, which is discontinuous across a set $\Gamma = \{\Gamma_j, j = 1, 2, \ldots, M\}$ of fault lines*

$$\Gamma_j = \{\gamma_j(t) : t \in [0, 1]\} \subset D,$$

where γ_j are unknown parametric continuous curves. On $D \setminus \Gamma$ the function f is supposed to be sufficiently smooth. The domain D is bounded, closed, simply connected, and contains the convex hull of $\mathcal{S}_n$.

The detection algorithm is divided into five steps:

(I) DETECTION OF FAULT POINTS. Initially, the cell-based search method by Bentley and Friedman [12] (adopted also by Renka [27]) is applied to find the data point set $\mathcal{N}_k$ neighbouring to each point x_k of the data set $\mathcal{S}_n$. This preprocessing phase is a classic *nearest neighbour searching procedure* (NNSP), in which we make a subdivision of the domain D in cells and identify the set $\mathcal{N}_k$ for each x_k. Then, to locate the fault points of $\mathcal{S}_n$ that are close to a fault line, we propose a local interpolation scheme involving CRBIs, as alternative to the scheme based on thin plate splines first suggested by Gutzmer and Iske [23], and then reconsidered and extended by Crampton and Mason [19]. Therefore, considering the global *Shepard's formula* (see, e.g., [4])

$$F(x;\mathcal{S}_n) = \sum_{i=1}^{n} f_i \, \frac{d(x,x_i)^{-p}}{\sum_{j=1}^{n} d(x,x_j)^{-p}}, \qquad F(x_i;\mathcal{S}_n) = f_i, \quad i = 1,2,\ldots,n,$$

where $d(x,x_i)$ is the Euclidean distance in $\mathbb{R}^2$ and $p > 0$, we adapt it to our purposes and, in particular, to the cell structure of the domain D. Thus, we use a local Shepard's formula on each set $\mathcal{N}_k$, $k = 1,2,\ldots,n$, (note that $x_k \notin \mathcal{N}_k$)

$$\begin{aligned} F(x_k;\mathcal{N}_k) &= \sum_{i=1}^{n_k} f_i \, \frac{d(x_k,x_i)^{-2}}{\sum_{j=1}^{n_k} d(x_k,x_j)^{-2}}, \\ F(x_i;\mathcal{N}_k) &= f_i, \quad i = 1,2,\ldots,n_k, \end{aligned} \tag{1}$$

n_k being the number of points of $\mathcal{N}_k$.

Let us consider now the absolute value of the difference $\sigma(x_k)$ between the function value $f(x_k)$ and the value of the interpolant at x_k, i.e.

$$\sigma(x_k) = |f(x_k) - F(x_k;\mathcal{N}_k)|\,. \tag{2}$$

Supposing that the interpolant gives a good approximation to f in D, then $\sigma(x_k)$ gives a measure of smoothness of the function around x_k. Choosing a suitable threshold value $\sigma_0 > 0$, we compare the values $\sigma(x_k)$ and σ_0: if $\sigma(x_k)$ is less than or equal to σ_0 then f is smooth in a neighbourhood of x_k, or else there is a steep variation of f at x_k and accordingly x_k will be marked as a fault point. When this procedure has involved all the data points, we have characterized the *set of fault points*

$$\mathcal{F}(f;\mathcal{S}_n) = \{x_k \in \mathcal{S}_n : \sigma(x_k) > \sigma_0\}, \tag{3}$$

which consists of all the data points which are expected either to belong to the faults or to be close to them.

A crucial point is an optimal choice of the threshold value σ_0. It is, in general, a difficult task, because the finite number of function values is the only available information. We start computing the largest deviation

$$S_{max} = \max\{|f_i - f_j| : i > j, \text{ for all } i, j = 1, 2, \ldots, n\}, \tag{4}$$

because obviously $0 < \sigma_0 < S_{max}$. An appropriate sorting procedure, included in the preprocessing phase, gives the set of all deviations and, at the same time, supplies some additional information on the variation of f.

Then we go on to find an optimal value of σ_0, introducing a variable threshold parameter, namely

$$\sigma_0^{(t)} = \frac{S_{max}}{\sqrt{2^t}}, \qquad \text{for } t = 1, 2, \ldots.$$

Starting up an iterative procedure, we increase the value of t by one unit each time and compare $\sigma(x_k)$ with $\sigma_0^{(t)}$. When the number of the found fault points is considered appropriate with information that we acquire analyzing Shepard's surface and the contour lines, the process successfully ends.

In general, a few iterations suffice to determine an optimal threshold value $\sigma_0^{(t^*)}$ and we assume $\sigma_0 \equiv \sigma_0^{(t^*)}$. Moreover, in some cases, one can shorten the process, starting with values of t greater than 1, as suggested by examination of Shepard's surface.

(II) SHEPARD'S SURFACE AND CONTOUR LINES. To get information on the surface, its faults, and the parameter σ_0, starting from the set $\mathcal{S}_n$ of data points and the corresponding function values f_i, $i = 1, 2, \ldots, n$, we obtain a surface representation by the following local Shepard's formula

$$F(x; \mathcal{I}_x) = \sum_{i=1}^{n_x} f_i \, \frac{d(x, x_i)^{-2}}{\sum_{j=1}^{n_x} d(x, x_j)^{-2}}, \qquad F(x_i; \mathcal{I}_x) = f_i, \quad i = 1, 2, \ldots, n_x, \tag{5}$$

where $\mathcal{I}_x$ is the set of the n_x data points closest to the point x, contained in a ball centred at x. In general, a suitable choice of n_x is 10. This Shepard's surface and the relative contour lines give very useful information about the surface behaviour and the position of possible faults. A different approach is proposed in [31] to obtain additional information about geological surfaces.

(III) CONSTRUCTION OF BARYCENTRES. Given a fault point $x_k \in \mathcal{F}(f; \mathcal{S}_n)$ and the corresponding nearest neighbour set $\mathcal{N}_k$, we define $D_k = \{x \in D : d(x, x_k) \le R_k\}$, where $R_k = \max\{d(x_i, x_k) : x_i \in \mathcal{N}_k\}$. Then we order the $N_k + 1$ points in $\bar{\mathcal{N}}_k = \mathcal{N}_k \cup \{x_k\}$, being $N_k = \text{card}(\mathcal{N}_k)$, so that $f(x_{k1}) \le f(x_{k2}) \le \cdots \le f(x_{kN_k+1})$. The expected jump δ_k of f in the subdomain D_k is evaluated by

$$\delta_k = \max_{1 \le l \le N_k} \Delta f(x_{kl}),$$

where Δ is the forward difference operator. We set l_k the lowest value of the index $l \in \{1, 2, \ldots, N_k\}$ for which δ_k is obtained.

The part $\Pi_k = D_k \cap \Gamma$ of a fault line separates the set $\Delta_k^L = \{y \in \bar{\mathcal{N}}_k : f(y) \le f(x_{kl_k})\}$ of all points of $\bar{\mathcal{N}}_k$ with lower function values from the set $\Delta_k^H = \{y \in \bar{\mathcal{N}}_k : f(y) > f(x_{kl_k})\}$ of

all points of $\bar{\mathcal{N}}_k$ with higher function values. If Δ_k^L or Δ_k^H is the empty set, then we enlarge $\mathcal{N}_k$ and repeat the process, so that $\bar{\mathcal{N}}_k$ contains points lying in the two parts of the subdomain separated by the fault line. Having determined Δ_k^L and Δ_k^H in this way, we calculate the barycentres A_k^L and A_k^H of Δ_k^L and Δ_k^H, respectively. Then we find $A_k = (A_k^L + A_k^H)/2$ and put it in $\mathcal{A}(f;\mathcal{S}_n)$, the *set of the barycentres*.

(IV) Construction and sorting of barycentres of barycentres. This phase consists of two stages:

`Stage 1.` We subdivide the domain D by a regular grid and point out the grid cells containing points of $\mathcal{A}(f;\mathcal{S}_n)$, i.e. barycentres. Since each grid cell contains at least a barycentre, we calculate the barycentre of the points of $\mathcal{A}(f;\mathcal{S}_n)$ in each cell, namely, the barycentre of the barycentres for each cell. In this manner, we obtain a further set $\mathcal{B}(f;\mathcal{S}_n)$, whose points are generally closer to the faults than the barycentres.

`Stage 2.` We order the points of the set $\mathcal{B}(f;\mathcal{S}_n)$, so that each point can be identified by an index. In this way, we obtain the ordered set

$$\mathcal{C}(f;\mathcal{S}_n) = \{C_i, i = 1, 2, \ldots, m\},$$

where m is the number of points of $\mathcal{B}(f;\mathcal{S}_n)$.

The nearest neighbour searching procedure used in the sorting process to build $\mathcal{C}(f;\mathcal{S}_n)$ is as follows:

1. Detection of any point of $\mathcal{B}(f;\mathcal{S}_n)$ nearest to a side of the domain D and assumption of this one as the first probe point.

2. Search of the nearest neighbour point to the actual probe point and assumption of this one as the new probe point, excluding the points already considered.

3. Stopping the process, when all points of $\mathcal{B}(f;\mathcal{S}_n)$ are ordered.

The knowledge of $\mathcal{C}(f;\mathcal{S}_n)$ is fundamental, because it allows us to handle complex situations as the presence of several faults, and intersections or bifurcations of faults. Indeed, the problem of splitting fault lines into their branches is, in general, a hard task for every method. Now, we can connect each point of $\mathcal{C}(f;\mathcal{S}_n)$ with the following one by a straight line segment, thus obtaining a preliminary, generally low accurate, approximation curve. Since each point of $\mathcal{C}(f;\mathcal{S}_n)$ is labelled and the direction of the ordered points is known, we can simply subdivide the set $\mathcal{C}(f;\mathcal{S}_n)$ into a number q of subsets greater or equal to the number of fault lines. Each subset of $\mathcal{C}(f;\mathcal{S}_n)$ is denoted by the symbol $\mathcal{C}_j(f;\mathcal{S}_n)$, for $j = 1, 2, \ldots, q$, and it is still an ordered set.

(V) Iterative refinement. In some cases, it is convenient to consider a further stage of refinement, called *iterative refinement*. It consists of two steps:

1. From the set $\mathcal{C}_j(f;\mathcal{S}_n) = \{C_{i,j}, i = 1, 2, \ldots, m_j, j = 1, 2, \ldots, q\}$, we consider $C_{i-1,j}$, $C_{i,j}$ and $C_{i+1,j}$, for $2 \le i \le m_j - 1$.

2. These three points are considered as the vertices of a triangle; then, we calculate the barycentre $C_{i,j}^{(1)}$, $2 \le i \le m_j - 1$, $j = 1, 2, \ldots, q$, for each of the $m_j - 2$ triangles, holding $C_{1,j}$ and $C_{m_j,j}$ fixed. These barycentres form the set $\mathcal{C}_j^{(1)}(f; \mathcal{S}_n)$, $j = 1, 2, \ldots, q$.

Step 2 can be repeated, so obtaining the barycentres $C_{i,j}^{(k)}$, forming the set $\mathcal{C}_j^{(k)}(f; \mathcal{S}_n)$, for $i = 2, 3, \ldots, m_j - 1$, $j = 1, 2, \ldots, q$, $k = 2, 3, \ldots$ We can see the good performance of this technique in Figure 1 and Figure 2, where we compare the polygonal lines obtained by applying or not the iterative refinement.

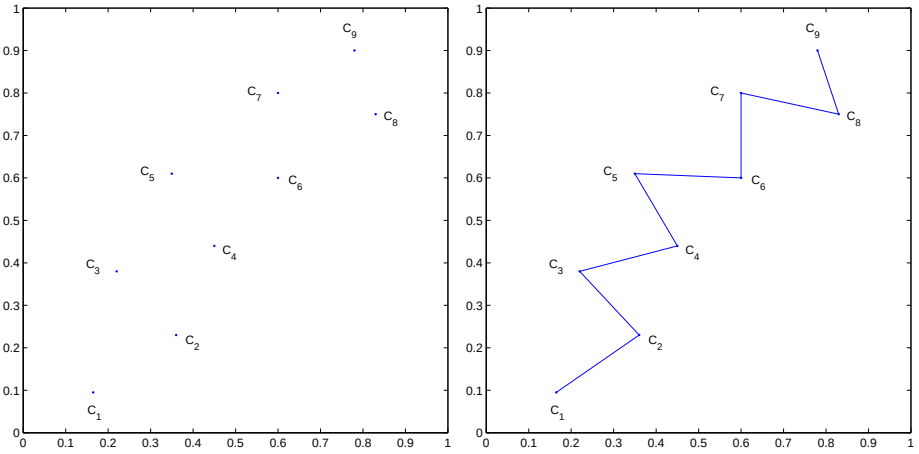

Figure 1: Set $\mathcal{C}(f; \mathcal{S}_n) = \{C_i, i = 1, 2, \ldots, 9\}$ (left) and polygonal line obtained without applying the iterative refinement (right).

This approach, based on repeated averages, is simple but very powerful, because it allows us to construct smooth curves and obtain accurate approximations. In Section 5, we will discuss in detail some particularly difficult cases, in which the presence of bifurcations requires a splitting of fault lines.

3 Adaptive detection of fault lines

The basic idea of adaptability is to start with a few hundreds of data points, locating roughly the position of possible fault lines. Then, as long as the set of the found barycentres of barycentres does not give sufficient information, we repeatedly increase the number of nodes generating new data points within local neighbourhoods (see [7]). The barycentres are considered enough when a comparison of them with the contour lines generated by a Shepard approximation of the faulted surface is satisfying.

Here is the algorithm in detail:

`Stage 1.` Let $\mathcal{S}_n = \{x_k, k = 1, 2, \ldots, n\}$ be a set of distinct and scattered data points or nodes in a domain $D \subset \mathbb{R}^2$, and $\mathcal{F}_n = \{f_k = f(x_k), k = 1, 2, \ldots, n\}$ a set of corresponding data values obtained by an unknown (possible) discontinuous function $f : D \to \mathbb{R}$ (see Definition 2.1).

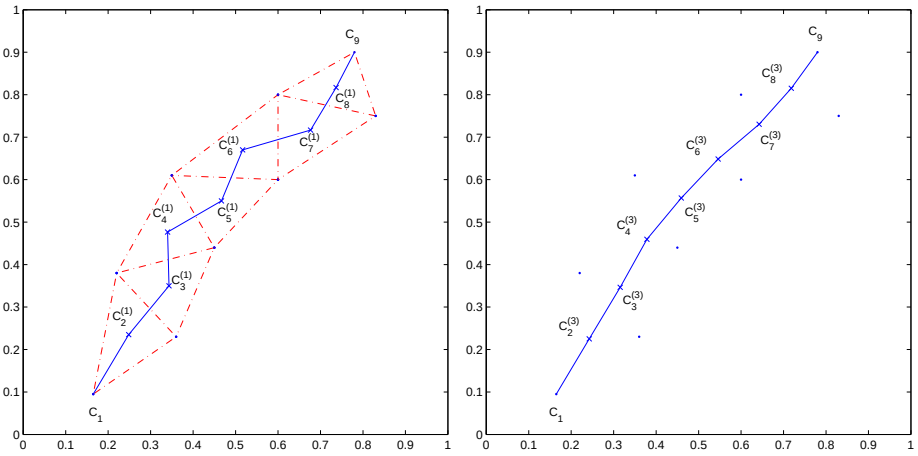

Figure 2: Polygonal line obtained by applying the iterative refinement to $\mathcal{C}(f;\mathcal{S}_n) = \{C_i, i = 1, 2, \ldots, 9\}$ (left) and polygonal line obtained by using the iterative refinement with three iterations (right).

`Stage 2.` Compute the largest deviation (or jump size) S_{max} as in (4) of the unknown function f along each of two directions x and y. In particular, we have

```
1:   S = 0;
2:   quicksort_x(n, x, y, f);
3:   for (i = 1, ..., n)
4:   {
5:     S* = f_{i+1} − f_i;
6:     if (S* > S)
7:       S = S*;
8:   }
9:   S_max = S;
```

This process is repeated for possible updating of S_{max} by setting $quicksort_y(n, x, y, f)$ in `line 2`.

`Stage 3.` Set $\sigma_0 = \epsilon \cdot S_{max}$, with $0 < \epsilon < 1$.

`Stage 4.` Apply a cell-based search method to find for each node x_k of $\mathcal{S}_n$ a suitable neighbourhood $\mathcal{N}_k$ of the data points [12]. This phase consists of making a partition in cells of the domain D and identify the data points closest to a node within each cell.

`Stage 5.` Employ the local version of Shepard's formula (1) on each set $\mathcal{N}_k$, $k = 1, 2, \ldots, n$, and evaluate the difference $\sigma(x_k)$ between the function value $f(x_k)$ and the value of the local interpolant $F(x_k;\mathcal{N}_k)$ given in (2). In particular, chosen a suitable threshold value $\sigma_0 > 0$, we have that

1. if $\sigma(x_k) \leq 0$, then f is smooth in a neighbourhood of x_k;

2. else $\sigma(x_k) > 0$, that is, there is a steep variation of f at x_k, and x_k is marked as a fault point.

`Stage 6.` Locate the set $\mathcal{F}(f;\mathcal{S}_n)$ of the fault points as in (3).

`Stage 7.` Find the set $\mathcal{A}(f;\mathcal{S}_n)$ of barycentres (see Step (III) in Section 2).

`Stage 8.` Find the set $\mathcal{B}(f;\mathcal{S}_n)$ of barycentres of barycentres and its ordered version $\mathcal{C}(f;\mathcal{S}_n)$ (see Step (IV) in Section 2).

`Stage 9.` Establish if the detection stage has been sufficiently accurate:

1. if the faults are well identified, run `Stage 10`;

2. else increase the number of nodes, generating a certain number of "local" scattered data on rectangular neighbourhoods centred at C_i, $i = 1, 2, \ldots, m$, namely the points of the set $\mathcal{C}(f, \mathcal{S}_n)$, and then repeat from `Stage 1` to `Stage 9` on the new enlarged set of nodes.

Iterating the procedure with the enlarged set of nodes as described in Step 2 (see Subsection 5.2), in general we obtain more (and new) fault points discarding eventually false fault points previously considered, more barycentres, and accordingly more barycentres of barycentres.

`Stage 10.` Apply the iterative refinement (see Step (V) in Section 2), subdividing if required the set $\mathcal{C}(f;\mathcal{S}_n)$ into a suitable number q of subsets, which are denoted by $\mathcal{C}_j(f;\mathcal{S}_n)$, for $j = 1, 2, \ldots, q$.

4 Approximation of fault lines

We consider different approaches to approximate fault lines, namely the polygonal line method, the least squares method, and the best l_∞ approximation method.

The polygonal method must be applied in all cases, because it is not only useful in itself, but also preliminary to the other methods.

The minimax approach, in general, assigns too much weight to outliers. The least squares approach gives substantially more weight to a point that is farther to the approximation curve than the rest of data, but will not allow that point to completely dominate the approximation. In our case, the situation just described is little important, because the refinement procedure works out repeated averages.

◇ Polygonal line method. The polygonal line method that we consider represents an improvement of the procedures developed by Gutzmer and Iske [23] and by Allasia, Besenghi and De Rossi [3].

The ordered points of $\mathcal{C}(f;\mathcal{S}_n)$ are connected by straight line segments, obtaining polygonal lines which approximate the fault lines. The performance of the approximation procedures depends, in particular, on the number of points, the cell dimension, and the form of the faults. Indeed, if a fault is centred in the cells, the polygonal method produces good results; otherwise, the polygonal line may appear too irregular and poorly accurate. To yield smoother (and, possibly, more accurate) polygonal lines, we use the iterative refinement and connect the points $C_{i,j}^{(k)}$ by straight line segments.

◇ LEAST SQUARES METHOD. The least squares method can be used to approximate a data set $\{(x_i, y_i), i = 1, 2, \ldots, m\}$ by an algebraic polynomial

$$P_s(x) = a_0 + a_1 x + \cdots + a_{s-1} x^{s-1} + a_s x^s \tag{6}$$

of degree $s < m - 1$. In our problem we choose the constants $a_0, a_1, \ldots, a_s$ to solve the *least squares problem*

$$\min_{a_0, a_1, \ldots, a_s} \sum_{i=1}^{m} \left[y_i - P_s(x_i) \right]^2 ,$$

where $(x_i, y_i) \in \mathcal{C}(f; \mathcal{S}_n)$. Replacing the coefficients $a_0, a_1, \ldots, a_s$ in (6) with the values obtained by the least squares method, we get the polynomial approximating the fault line.

If there is only one fault in D, then the discrete least squares method can be advantageously applied. Otherwise, when we deal with complex situations (several faults, intersections or bifurcations of faults), the least squares method cannot be applied directly, but a suitable subdivision of $\mathcal{C}(f; \mathcal{S}_n)$ is required. Indeed, if the surface shows two or more fault lines, we need to split the set $\mathcal{C}(f; \mathcal{S}_n)$ in a number of subsets greater or equal to the number of fault lines before applying the least squares method. This procedure is suggested also when a fault line is not of open type. Acting in this way, the least squares method yields very good results.

Finally, when a fault line is parallel or nearly parallel to the y-axis, it is convenient to make first a rotation of the coordinate axes and then to apply the least squares method.

◇ BEST l_∞ APPROXIMATION METHOD. The best l_∞ (or Chebychev) approximation method can be considered as a tool for finding polynomial approximations to fault lines starting from the set $\mathcal{C}(f; \mathcal{S}_n)$ or from some further set possibly subjected to iterative refinement. The polynomial in (6) is chosen such as to minimize the largest absolute value of the differences $P_s(x_k) - y_k$ at $m < n$ discrete abscissae x_k, $1 \le k \le m$.

Given the abscissae x_k and the corresponding ordinates y_k, for $k = 1, 2, \ldots, m$, the polynomial in (6) is sought to solve the *minimax problem* (see, e.g., [11])

$$\min_{a_0, a_1, \ldots, a_s} \max_k \left| P_s(x_k) - y_k \right| . \tag{7}$$

Introducing the *residuals* r_k at the abscissae x_k

$$P_s(x_k) - y_k = r_k, \quad k = 1, 2, \ldots, m,$$

(7) becomes

$$\min_{a_0, a_1, \ldots, a_s} \max_k \left| r_k \right| .$$

Among the residuals r_k there exists at least one with the largest absolute value. Let us denote it by $H = \max_k |r_k| > 0$. Hence, we have the inequalities

$$\left| P_s(x_k) - y_k \right| \le H, \quad k = 1, 2, \ldots, m, \tag{8}$$

and the problem in (7) is equivalent to minimize the value H. Now, we rewrite (8) in the form

$$\left| \sum_{j=0}^{s} \left(\frac{a_j}{H} \right) x_k^j - \left(\frac{1}{H} \right) y_k \right| \le 1, \quad k = 1, 2, \ldots, m. \tag{9}$$

With the unknowns

$$\xi_1 = \frac{a_0}{H}, \quad \xi_2 = \frac{a_1}{H}, \quad \xi_3 = \frac{a_2}{H}, \ \ldots, \ \xi_{s+1} = \frac{a_s}{H}, \quad \xi_{s+2} = \frac{1}{H},$$

the conditions (9) read as follows

$$\left|\xi_1 + x_k\xi_2 + x_k^2\xi_3 + \ldots + x_k^s\xi_{s+1} - y_k\xi_{s+2}\right| \leq 1, \quad k = 1, 2, \ldots, m. \tag{10}$$

To avoid expressions with absolute values, we observe that each inequality of (10) is equivalent to two inequalities. Moreover, the variable ξ_{s+2} equals the reciprocal value of the non-negative quantity H which has to be minimized. Thus we obtain the following linear program

$$\text{maximize} \qquad Z = \xi_{s+2},$$

subject to the constraints

$$\begin{aligned} \xi_1 + x_k\xi_2 + x_k^2\xi_3 + \ldots + x_k^s\xi_{s+1} - y_k\xi_{s+2} &\leq \quad 1, \\ \xi_1 + x_k\xi_2 + x_k^2\xi_3 + \ldots + x_k^s\xi_{s+1} - y_k\xi_{s+2} &\geq -1, \end{aligned}$$

for $k = 1, 2, \ldots, m$, and

$$\xi_{s+2} \geq 0.$$

As only ξ_{s+2} has to satisfy a sign condition, the variables $\xi_1, \xi_2, \ldots, \xi_{s+1}$ are free. Since the standard procedure requires that all the variables have non-negativity constrains, any problem containing variables allowed to be negative must be converted to an equivalent problem involving only non-negative variables before the *simplex method* is applied. Each free variable ξ_j, $1 \leq j \leq s+1$, is replaced throughout the model by the difference of two new non-negative variables

$$\xi_j = \xi_j^+ - \xi_j^-, \quad j = 1, 2, \ldots, s+1.$$

Since ξ_j^+ and ξ_j^- can have any non-negative values, the difference $\xi_j^+ - \xi_j^-$ can have any value (positive and negative), so it is a legitimate substitute for ξ_j in the model. Now, applying the simplex method, we determine the coefficients a_0, $a_1, \ldots, a_s$ of (6) and identify an approximating polynomial to the fault line.

The considerations previously developed for the least squares method are effective also for the best l_∞ approximation method. Hence, problems considering several faults, intersections or bifurcations of faults can be successfully solved. Obviously, in particularly difficult cases, the iterative refinement can be used in both the least squares method and the best l_∞ approximation method; this gives a greater smoothness to approximating curves.

We note that solving a minimization problem to find fault line approximation is proposed and carefully investigated by Crampton and Mason [19], who use the simplex method too.

Remarks.

a) Smoothness. The performance of the approximation procedures depends, in particular, on the number of points and the form of the faults. Least squares and best l_∞ approximations give smooth curves, whereas polygonal lines have a simpler formulation. If a polygonal line

appears too irregular and poorly accurate, we can obtain another smoother and, possibly, more accurate polygonal line, using the refinement technique and connecting the barycentres by straight line segments. Iterating a few times the refinement technique, we construct smooth, planar curves rather than polygonal curves. This is a meaningful advantage if compared with the Gutzmer-Iske method [23].

b) Connection errors. When we have to deal with complex surface discontinuities, applying least squares and best l_∞ approximations require a further subdivision of data. This situation can create small errors at points in which two piecewise curves join, but the introduction of the refinement technique considerably reduces the occurrence of these errors of connection. Nevertheless, the decision of how to subdivide the domain can hide several difficulties (see [19]). Obviously, in particularly difficult cases, the refinement technique can be used and iterated in both the least squares method and the best l_∞ approximation method.

5 Numerical results

In this section we propose a detailed and extensive investigation of a few test functions. They are examples of several numerical and graphical results, obtained by computational procedures developed in C/C++, MATLAB, and MAPLE environments.

In the various tests we consider n randomly scattered data points x_i in the square $[0,1] \times [0,1] \subset \mathbb{R}^2$, and the corresponding function values f_i, for $i = 1, 2, \ldots, n$, supposing that these are the only information relative to f at our disposal. All the nodes are generated by means of the MATLAB command `rand`, which gives uniformly distributed random points on the interval (0,1). In fact, in many real applications, one does not know the location of faults on the surface or even whether or not the surface is faulted [9].

The detection and approximation schemes are tested against functions with different kinds of faults, varying the dimension n of the sample generated by a uniform distribution and the threshold value σ_0. The results are obtained by using local Shepard's formula (1), even if other choices of CRBIs could work as well. Although the CRBI choice can appear quite elementary, the use of more advanced mathematical tools do not guarantee remarkable improvements. Moreover, the cell-based search method and the CRBIs allow to partition the domain, to process the data in some stages, and to insert or remove data points. These features are particularly important in surveying phenomena, such as geodetic or geophysical activities, where data are distributed in regions with different characteristics.

In general, it is remarkable that, also taking a data point set with a few thousands or, in the adaptive approach, even a few hundreds scattered data, the method of detection holds its efficiency. In this case a loss of approximation accuracy is unavoidable, but it depends essentially on the reduced information, that is, the number of data points. In fact, the method allows to individualize the position of fault lines all the same, outlining roughly their form.

If the set of obtained fault points is not satisfactory, the initial set of data points can be enlarged, but considering only new data points located in a smaller region including the fault. Then we can repeat the detection process and find a new and larger set of fault points. The adaptiveness of this approach ensures good results, though the number of data points is meaningfully reduced. It will be extensively discussed in Subsection 5.2.

5.1 Approximation of unknown fault lines

Let us now consider some examples to test the standard process described in Section 2.

Example 1

The test function (see [23, 3, 24, 14])

$$f_1(x,y) = \begin{cases} 1 + 2\left\lfloor 3.5\sqrt{x^2+y^2}\right\rfloor, & \text{if } (x-0.5)^2+(y-0.5)^2 < 0.16, \\ 0, & \text{otherwise,} \end{cases} \tag{11}$$

is a surface with discontinuities across the set

$$\begin{aligned} \Gamma &= \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 = \{(x,y) \in \mathbb{R}^2 : (x-0.5)^2+(y-0.5)^2 = 0.16\} \\ &\cup \{(x,y) \in \mathbb{R}^2 : x^2+y^2 = 16/49,\ (x-0.5)^2+(y-0.5)^2 \le 0.16\} \\ &\cup \{(x,y) \in \mathbb{R}^2 : x^2+y^2 = 36/49,\ (x-0.5)^2+(y-0.5)^2 \le 0.16\}. \end{aligned}$$

This surface is always constant in $D \setminus \Gamma$, but presents faults intersected and/or bifurcated. In particular, first Gutzmer and Iske [23], and then Allasia, Besenghi and De Rossi [3] test their detection methods against this function using 2000 scattered data. López de Silanes, Parra and Torrens [24] also detect surface discontinuities using 2800 track data. This kind of data and, more in general, data disposed on a family of lines or curves can be advantageously treated by the method proposed by Allasia [5].

Since other authors considered $n = 2000$ data points, we also used a data set of the same dimension. Obviously, the proposed method works better considering a larger number of data points, e.g. $n = 4000, 10000$, as shown by errors in Table 1 and Table 2.

Starting from the set $\mathcal{S}_n$ of 2000 data points and the function values f_i, $i = 1, 2, \ldots, n$, we use the formula (5) that, in general, is applied to approximate a continuous surface. The obtained surface graphic and the respective contour lines, in Figure 3 (bottom), give us a preliminary, but very useful, information on the surface, in general, and on the dislocation of possible fault lines in particular.

After calculating the largest function oscillation, $S_{max} = 7$, and applying the cell-based search method in the preprocessing phase, we proceed with the calculus of the threshold value $\sigma_0^{(t)}$, $t = 1, 2, \ldots$, that occurs with $t^* = 12$ iterations, namely $\sigma_0 \equiv \sigma_0^{(t^*)} = \sigma_0^{(12)} \approx 0.11$. In practice, one can start with the value $\sigma_0^{(6)} = S_{max}/\sqrt{2^6}$, based on the information given by Shepard's surface.

Now, exploiting the particular cell structure of the domain D constructed in the previous step, we evaluate $F(x_k; \mathcal{N}_k)$ in (1) and compute $\sigma(x_k)$ in (2). We find 345 barycentres (belonging to the set $\mathcal{A}(f; \mathcal{S}_n)$) and 115 barycentres of barycentres (belonging to the set $\mathcal{B}(f; \mathcal{S}_n)$), which are shown in Figure 4 (left) and (right) respectively.

We remark that the phase which requires to split fault lines into different branches is not completely automatic, but a simple user interaction is required: it is to examine the polygonal lines (see Figure 5 (left)).

In general, if we use the polygonal line method in approximation phase, three refinements are needed; otherwise, least squares and best l_∞ approximation methods, which hold in themselves the smoothing property, require only one refinement. Figure 5 (right) presents the fault

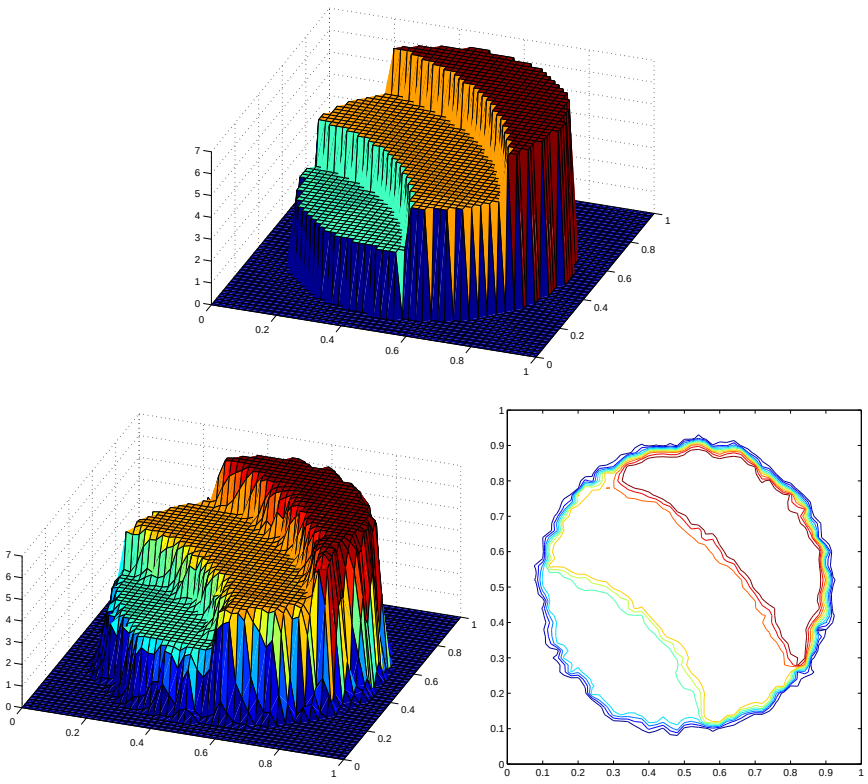

Figure 3: Analytic expression (top), Shepard's surface (bottom, to left) and contour lines (bottom, to right) of f_1.

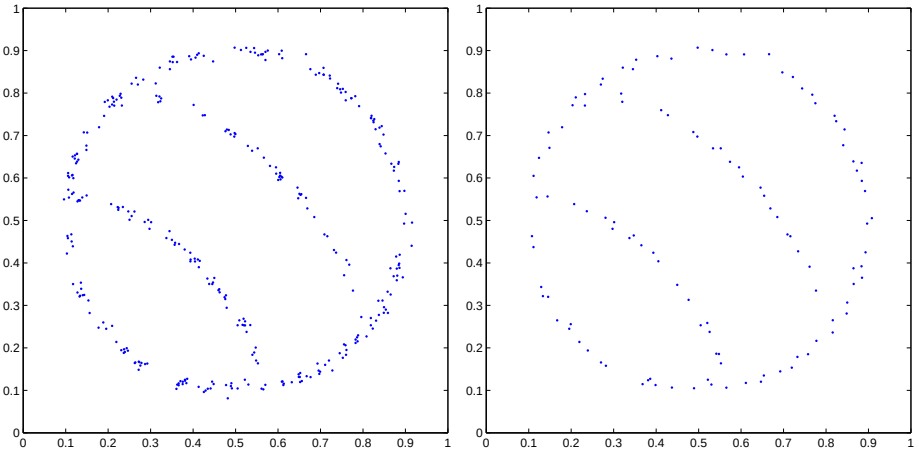

Figure 4: Set $\mathcal{A}(f;\mathcal{S}_n)$ of barycentres (left) and set $\mathcal{B}(f;\mathcal{S}_n)$ of barycentres of barycentres (right) of f_1.

approximation obtained by the polygonal line method, whereas the least squares and best l_∞ approximation methods are shown in Figure 6 (left) and (right), respectively.

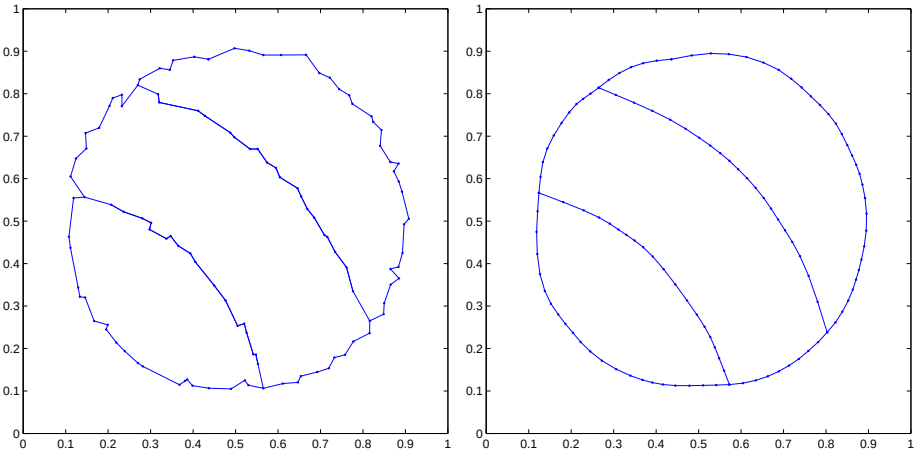

Figure 5: Initial approximation of the fault lines of f_1, obtained by using NNSP and polygonal lines (left), and accurate approximation, obtained by polygonal line method with three iterations (right).

In a similar way, we can handle any case, in which the unknown surface is almost constant in $D \setminus \Gamma$ with one or more faults, and with intersections or bifurcations of faults. Functions of this type are studied by various authors with different techniques and methodologies [26, 28, 13, 3, 24, 14, 19, 30, 25].

Example 2

As second test, we consider the function (see [10])

$$f_2(x,y) = \begin{cases} g(x,y), & \text{if } \ y < \min(1/3x, 0.25), \\ h(x,y), & \text{if } \ \min(1/3x, 0.25) \le y < 2x - 0.5, \\ 6, & \text{otherwise,} \end{cases} \tag{12}$$

with

$$\begin{aligned} g(x,y) &= 1 + 1.4375\left(2 - (x-1)^2\right)\exp(5\,(1-x)\,(4y-1)), \\ h(x,y) &= 1 + \left(2 - (x-1)^2\right)\left(2 - (y-1)^2\right), \end{aligned}$$

and discontinuity set

$$\Gamma = \Gamma_1 \cup \Gamma_2 = \left\{ \left(t, \frac{1}{3}t\right) : 0 \le t \le 0.75 \right\} \cup \left\{ \left(t, 2t - \frac{1}{2}\right) : 0.3 \le t \le 0.75 \right\}.$$

This function gives a discontinuous surface with two faults, which bifurcate at the point (3/10,1/10). The fault line Γ_1 has a jump size that vanishes when x tends to 0.75, while

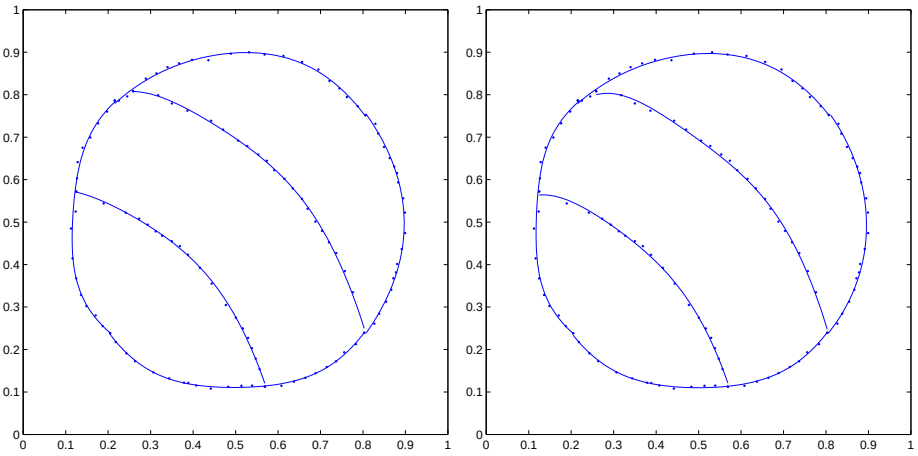

Figure 6: Accurate approximation of fault lines obtained by least squares (left) and best l_∞ approximation methods with one iteration (right) of f_1.

Γ_2 presents a variable jump size on all the domain D. The surface is smooth on $D \setminus \Gamma$, but shows different characteristics, because a part of the function is constant, another is almost constant, and the remainder changes enough quickly. It follows that this surface collects some features, which are difficult to be represented, namely, bifurcated fault lines, rapid surface variations, and changing or vanishing jump sizes.

In [26], Parra, López de Silanes and Torrens evaluate the function f_2 at 10000 points randomly distributed on the domain and then apply their detection method, which determines fault lines using a divergence property of sequences measuring gradient near discontinuities (see also [25] for a different characterization of jump discontinuities and the vertical fault detection).

In the following, we test our method focusing on the case of $n = 2000$ scattered data, but we also report errors obtained by $n = 4000$, 10000 (see Table 1 and Table 2).

In Figure 7, we present the analytic expression of f_2 (top), Shepard's surface (bottom, to left) and the corresponding contour lines (bottom, to right). Such preliminary information highlights the function characteristics, locating not only the two fault lines, but also the presence of rapid variations of the surface.

Thus, since the maximum jump size is $S_{max} \approx 4.98$, the procedure finds the threshold value $\sigma_0^{(t)}$ after 9 iterations, so that $\sigma_0 \equiv \sigma_0^{(9)} \approx 0.22$. This value leads, first, to obtain the set $\mathcal{A}(f;\mathcal{S}_n)$ consisting of 81 barycentres, and then the set $\mathcal{B}(f;\mathcal{S}_n)$ containing 41 points (see Figure 8). To shorten, one can start with $\sigma_0^{(4)} = S_{max}/\sqrt{2^4}$.

Now, subdividing the set $\mathcal{C}(f;\mathcal{S}_n)$ into two subsets $\mathcal{C}_j(f;\mathcal{S}_n)$, $j = 1, 2$, we obtain an initial approximation of the fault lines, in Figure 9 (left). Then, we apply the iterative refinement and get more accurate approximation curves, as shown in Figure 9 (right) and Figure 10.

Example 3

Finally, let us consider the discontinuity set Γ, which consists in the polygonal line of ver-

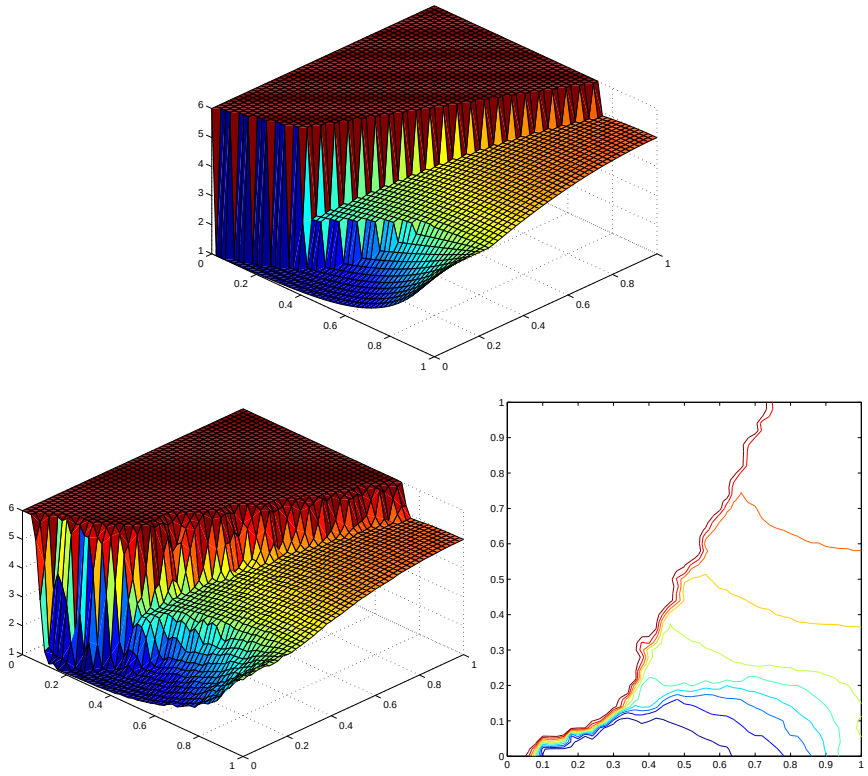

Figure 7: Analytic expression (top), Shepard's surface (bottom, to left) and contour lines (bottom, to right) of f_2.

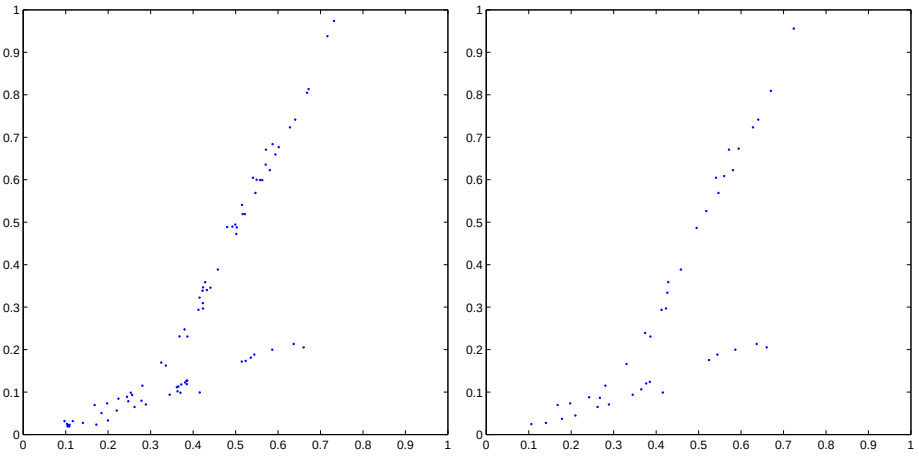

Figure 8: Set $\mathcal{A}(f;\mathcal{S}_n)$ of barycentres (left) and set $\mathcal{B}(f;\mathcal{S}_n)$ of barycentres of barycentres (right) of f_2.

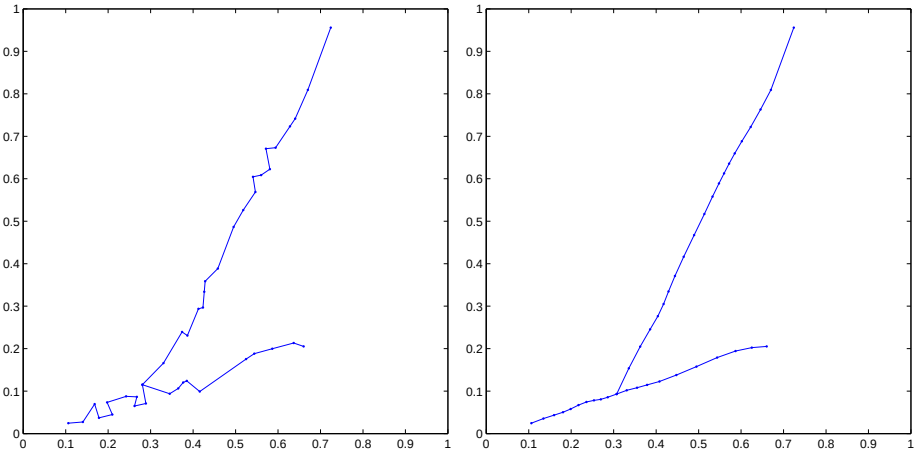

Figure 9: Initial approximation of the fault lines of f_2, obtained by using NNSP and polygonal lines (left), and accurate approximation, obtained by polygonal line method with three iterations (right).

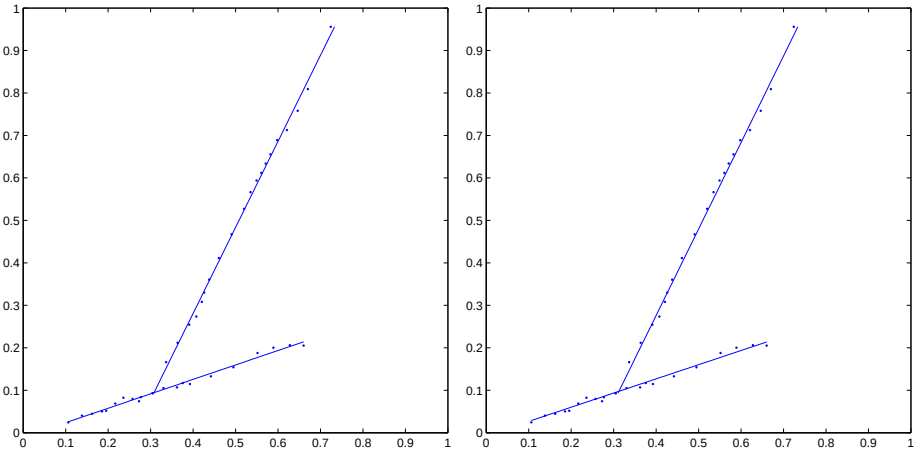

Figure 10: Accurate approximation of fault lines obtained by least squares (left) and best l_∞ approximation methods with one iteration (right) of f_2.

tices $(0, 0.2)$, $(0.2, 0.2)$, $(0.35, 0.225)$, $(0.42, 0.3)$, $(0.48, 0.4)$, $(0.49, 0.53)$, $(0.5, 0.65)$, $(0.65, 0.725)$, $(0.8, 0.75)$, $(1, 0.8)$, and let f be the function defined by

$$f_3(x,y) = \frac{1}{\left[\left(3x-\frac{7}{2}\right)^2 + \left(3y-\frac{7}{2}\right)^2\right]} + g(x,y),$$

where g denotes the function given by

$$g(x,y) = \begin{cases} 0.35 + \exp\left[-\left(3x-\frac{3}{2}\right)^2 - \left(3y-\frac{3}{2}\right)^2\right], & \text{if } (x,y) \in \omega, \\ 0, & \text{otherwise,} \end{cases}$$

and ω is the part of the domain D located above the fault line (according to the y-axis direction). This test function is considered by Gout et al. [22] (see also [17]).

Using a scattered data sample of dimension $n = 2000$, we test our method proposed in the paper and obtain the results as follows:

(a) From Shepard's formula (5) and the relative contour lines (see Figure 11 (bottom)), we have basic information on the behaviour and the orientation of the fault line.

(b) After obtaining the maximum surface variation $S_{max} \approx 2.06$, we find the threshold value $\sigma_0 \equiv \sigma_0^{(6)} \approx 0.26$. The detection phase identifies 36 points belonging to the set $\mathcal{A}(f; \mathcal{S}_n)$, while the set $\mathcal{B}(f; \mathcal{S}_n)$ contains 22 points, represented in Figure 12 (left).

(c) Constructing the set $\mathcal{C}(f; \mathcal{S}_n)$, we approximate the fault line by the polygonal line method applying once the iterative refinement as shown in Figure 12 (right).

Analogous results are obtained considering the least squares and best l_∞ approximation methods.

Functions with features similar to those described in the last two tests can be found in [26, 29, 24, 9]. Furthermore, recovering functions with discontinuities sampled on cartesian grid are studied in [15, 16].

For the first two test functions, we compute the maximum absolute error (MAE) and the root mean square error (RMSE), evaluating each fault line at 100 equispaced points in $0 \le x \le 1$. If a fault line is parallel or nearly parallel to the $y-$axis, it is convenient first to rotate the coordinate axes and then to compute the errors. The results are reported in Table 1 and Table 2. For simplicity, we compare only least squares and best l_∞ approximation curves, because the polygonal line method requires an evaluation of all linear pieces.

5.2 Adaptive approximation of unknown fault lines

In the adaptive detection algorithm a crucial point is the optimal choice of the threshold value σ_0. It is, in general, a difficult task, because the finite number of data is the only available information. Several tests showed that the procedure works well when $1/2^5 \le \epsilon \le 1/2^3$, and it ends successfully if the function f is smooth enough on $D \setminus \Gamma$. If the function f shows very steep variations and the procedure accordingly does not end successfully, we can repeatedly apply the process by using increasing values of $\epsilon > 1/2^3$. This reduces the sensitivity of the procedure, thus avoiding the detection of false fault points.

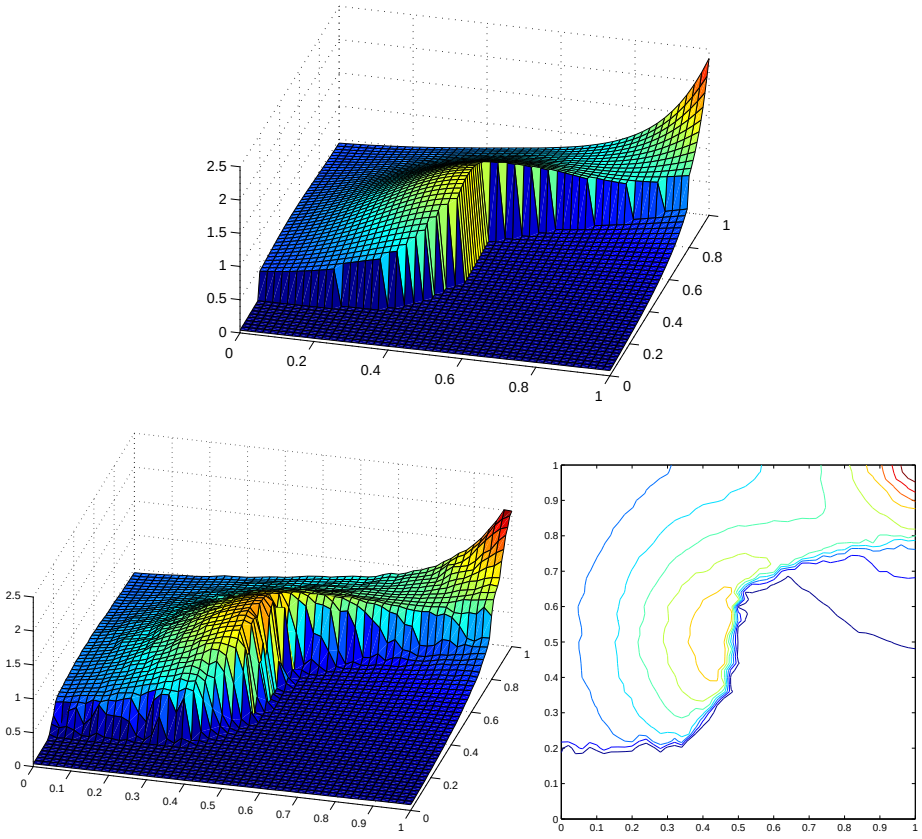

Figure 11: Analytic expression (top), Shepard's surface (bottom, to left) and contour lines (bottom, to right) of f_3.

f/n	2000	4000	10000
f_1	6.83495E − 2	2.83227E − 2	2.04529E − 2
	8.57487E − 3	3.63514E − 3	2.70869E − 3
f_2	1.06565E − 2	7.34524E − 3	4.41158E − 3
	7.97225E − 3	5.40742E − 3	3.79520E − 3

Table 1: MAEs and RMSEs resulting from least squares method.

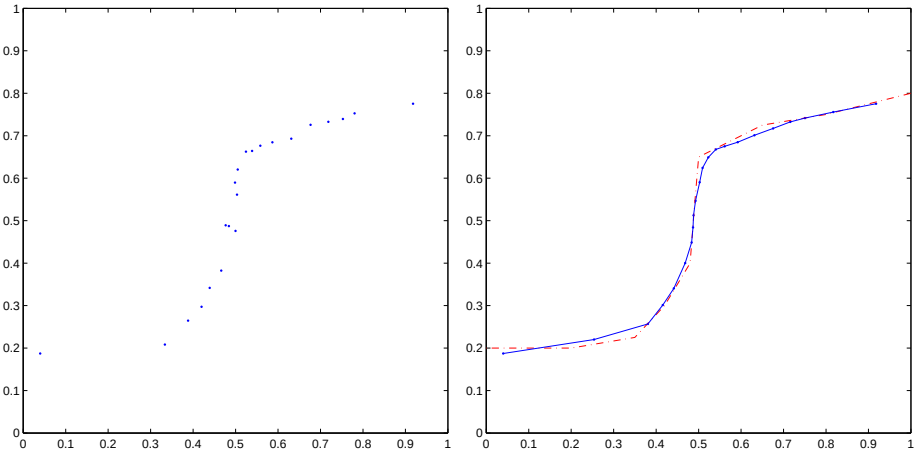

Figure 12: Set $\mathcal{B}(f;\mathcal{S}_n)$ of barycentres of barycentres (left) and comparison between the fault line of f_3 (dash-dot line) and the fault line approximation obtained by polygonal line method with one iteration (solid line) (right).

f/n	2000	4000	10000
f_1	6.89141E − 2 8.91904E − 3	2.63607E − 2 4.57793E − 3	2.08777E − 2 3.47217E − 3
f_2	1.34226E − 2 8.33712E − 3	1.06295E − 2 5.65426E − 3	9.24255E − 3 5.68766E − 3

Table 2: MAEs and RMSEs resulting from best l_∞ approximation method.

We sketch the results obtained by our method considering five significant test examples. Choosing the optimal value of ϵ, numerous tests pointed out that a few thousand data points are sufficient to produce an accurate approximation.

Now, for convenience we briefly summarize the adaptive approach to accurately detect the surface discontinuities. It consists of the following steps:

`Step 1.` Starting from a few hundreds of data points (here, we take $n = 250$) in the unit domain D, we apply the detection algorithm, locating roughly the position of possible fault lines. The process determines, first, the set $\mathcal{F}(f, \mathcal{S}_n)$ of fault points, then, the set $\mathcal{A}(f, \mathcal{S}_n)$ of barycentres and, finally, the set $\mathcal{B}(f, \mathcal{S}_n)$ of barycentres of barycentres and its ordered form $\mathcal{C}(f, \mathcal{S}_n)$.

`Step 2.` We increase the number of nodes, generating a certain number of "local" scattered data on rectangular neighbourhoods centred at C_i, $i = 1, 2, \ldots, m$, the points of the set $\mathcal{C}(f, \mathcal{S}_n)$ (in general, 15–20 nodes for each neighbourhood are sufficient).

`Step 3.` From the enlarged set of nodes we obtain more fault points, discarding eventually false fault points previously considered, and accordingly we obtain more barycentres and barycentres of barycentres.

`Step 4.` Stop, if the faults are well identified, else go back to `Step 2`.

In practice, a suitable choice of the (local) neighbourhood size in `Step 2` is essential. It is advisable to change the rectangular neighbourhood size, first considering larger neighbourhoods (here, side length and width are taken between 0.12 and 0.16 in the unit square) and then, in subsequent adaptive detection phases, reducing their sizes by about one half.

Note that the process allows us to evaluate only the latest generated data points, exploiting the previous evaluations due to Shepard's formula. Obviously, if we have some information about the position of fault lines, we can locate from the beginning a reduced searching region.

In the Figures we plot the test functions, the corresponding fault line approximations and a picture of the adaptive detection process. An error analysis is shown as well. It gives an idea of the good quality of our approach.

Example 4

The test function (see [13, 3])

$$f_4(x, y) = \begin{cases} y - x + 1, & \text{if } 0 \le x < 0.5, \\ 0, & \text{if } 0.5 \le x < 0.6, \\ 0.3, & \text{if } x \ge 0.6, \end{cases}$$

is a surface with discontinuities in $x = 0.5$ and $x = 0.6$, i.e.

$$\Gamma = \Gamma_1 \cup \Gamma_2 = \{(0.5, t) : 0 \le t \le 1\} \cup \{(0.6, t) : 0 \le t \le 1\}.$$

Figure 13 (left) shows that, outside the discontinuities, the function $f_4(x, y)$ is constant for $x > 0.5$, but it changes quickly enough for $x \le 0.5$, producing a variation of the jump size in Γ_1. Moreover, Figure 13 (right) provides the graphical representation of the least squares curves from the 75 points of the set $\mathcal{C}(f; \mathcal{S}_n)$. In Figure 14 we show step by step the adaptive

detection procedure, representing (bottom, left to right) the subsequent enlarged sets of data points and (top, left to right) related sets $\mathcal{C}(f;\mathcal{S}_n)$. More precisely, in Figure 14 **(A)** (see `Step 1`), we plot the points of the set $\mathcal{C}(f;\mathcal{S}_n)$ obtained by applying the detection procedure from 250 scattered data (not represented). Figure 14 **(C)** and **(E)** (see `Step 3`) refer to new sets $\mathcal{C}(f;\mathcal{S}_n)$ of points obtained by increasing the number of nodes on or close to possible fault lines, as shown in Figure 14 **(B)** and **(D)** respectively (see `Step 2`). Then, Figure 14 **(F)** (see `Step 2`) contains the enlarged set of data points from which we get the final set $\mathcal{C}(f;\mathcal{S}_n)$. The related approximation lines (see Figure 13 (right)) are obtained by subdividing at first $\mathcal{C}(f;\mathcal{S}_n)$ into two subsets, to which the refinement technique is applied, and then by applying the least squares method to the resulting subsets. The six plots in Figure 14 must be considered in alphabetical order.

In the detection phase, to obtain an accurate approximation of the faults, we employ 3479 data points (this number is purely indicative and may considerably vary). In some cases, nevertheless, if a rough knowledge of faults (e.g. position, behaviour, etc.) is sufficient, the number of the considered points can be considerably reduced (say, to about a quarter). In doing so, we may reduce the number of steps in the adaptive detection process, but obviously we do not attain the accuracy obtained in this article.

All these remarks hold also for the other test functions (see Figure 16, 18, 20, 22).

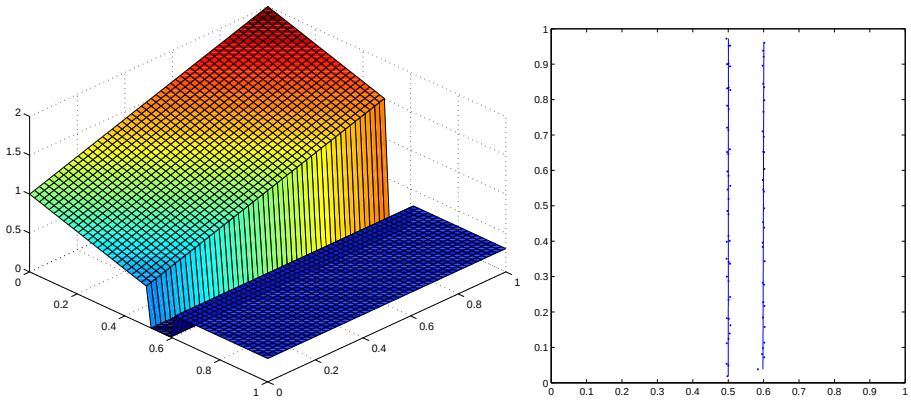

Figure 13: Test function f_4 (left) and least squares approximation curves (right).

Example 5

The test function (see [24])

$$f_5(x,y) = \begin{cases} 0, & \text{if } x > 0.4, y > 0.4, y < x + 0.2, \\ (x-1)^2 + (y-1)^2, & \text{otherwise,} \end{cases}$$

is a surface with discontinuities across the set

$$\begin{aligned} \Gamma = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 &= \{(t, 0.4) : 0.4 \le t \le 1\} \cup \{(0.4, t) : 0.4 \le t \le 0.6\} \\ &\cup \{(t, t + 0.2) : 0.4 \le t \le 0.8\}, \end{aligned}$$

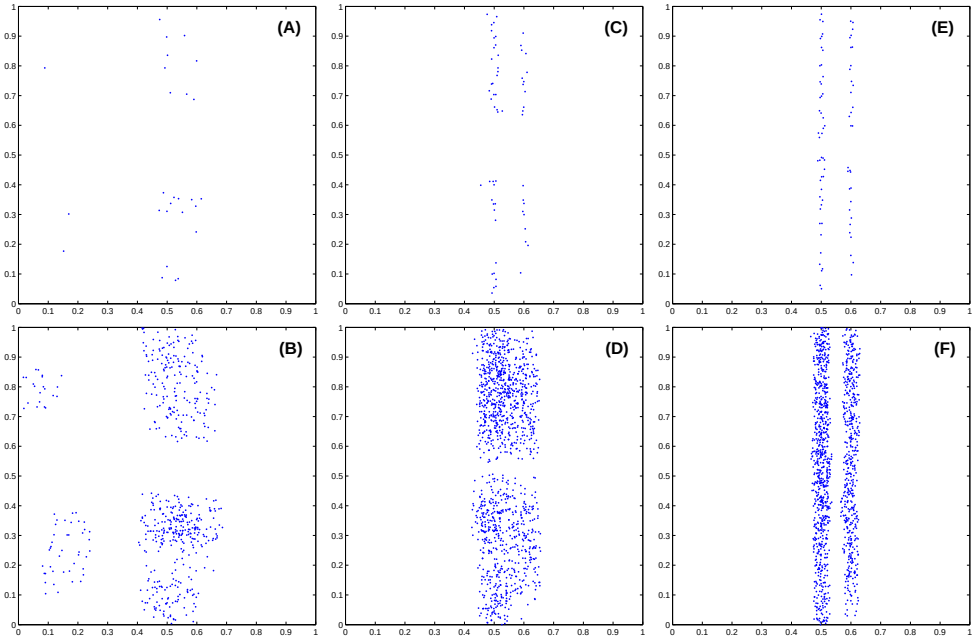

Figure 14: Subsequent enlarged sets of data points (bottom, left to right) and related sets $\mathcal{C}(f;\mathcal{S}_n)$ (top, left to right) in adaptive detection procedure for f_4.

as shown in Figure 15 (left). Moreover, we can observe that the jump size of fault lines varies in Γ_1, Γ_2 and Γ_3.

Figure 15 (right) provides the graphical representation of the best l_∞ approximation curves from the 45 points belonging to the set $\mathcal{C}(f;\mathcal{S}_n)$. Figure 16 presents, instead, a picture of the adaptive detection procedure. The fault line is obtained from the set $\mathcal{C}(f;\mathcal{S}_n)$, subdividing the original set into three subsets.

The adaptive detection process ends successfully using 2099 scattered data.

Example 6

In this test case, we still consider the function f_1 given in (11), which is displayed in Figure 17 (left), while the graph is obtained by the polygonal line method from the ordered 108 points of $\mathcal{C}(f;\mathcal{S}_n)$ (right). Figure 18 presents the results of the adaptive detection procedure. Also in this case, we consider a subdivision of the set $\mathcal{C}(f;\mathcal{S}_n)$. Finally, to obtain a greater smoothness, we apply the refinement technique three times.

In this test, to obtain the result of Figure 17 (right), 5185 nodes are considered in input.

Example 7

The next test considers the function (see [26, 3, 24, 25]),

$$f_6(x,y) = \begin{cases} (2-(x-0.8)^2)(2-(y-0.8)^2), & \text{if } \ y > l(x), \\ (1-(x-0.5)^2)(1-(y-0.2)^2), & \text{otherwise,} \end{cases}$$

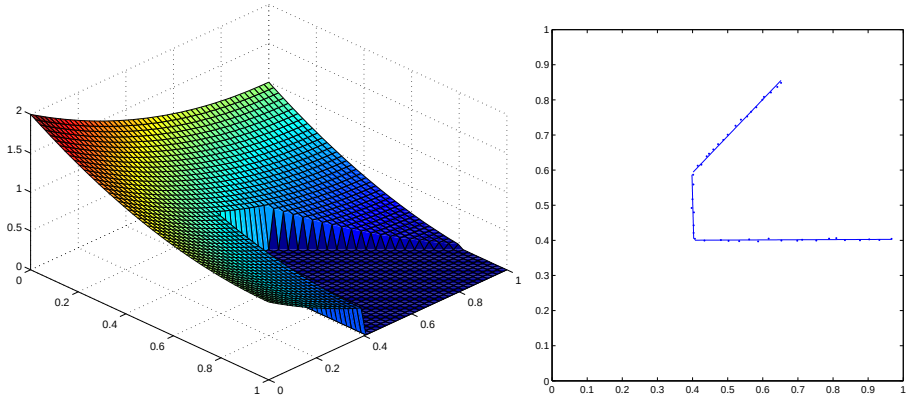

Figure 15: Test function f_5 (left) and best l_∞ approximation curves (right).

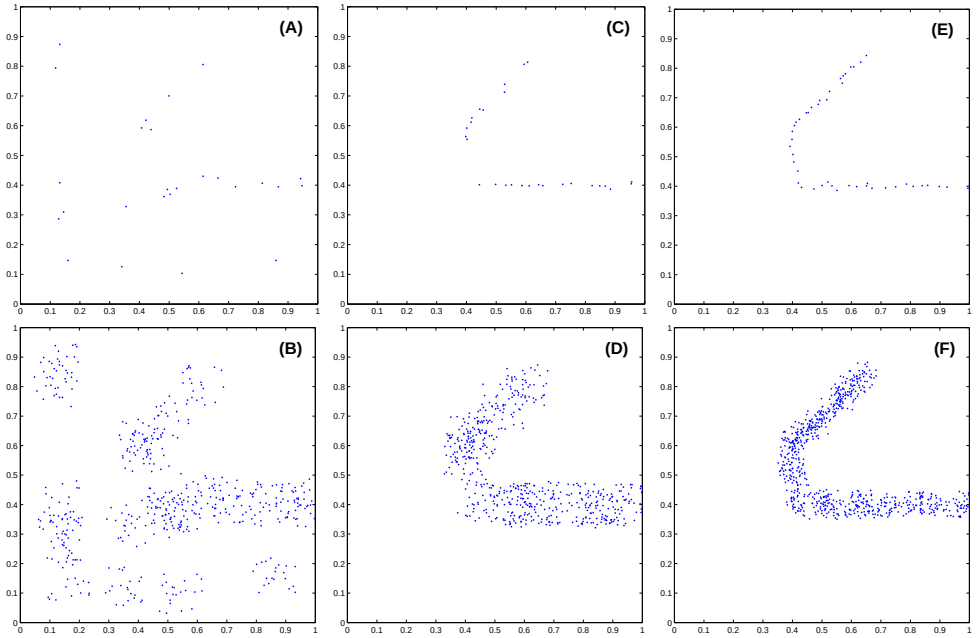

Figure 16: Subsequent enlarged sets of data points (bottom, left to right) and related sets $\mathcal{C}(f;\mathcal{S}_n)$ (top, left to right) in adaptive detection procedure for f_5.

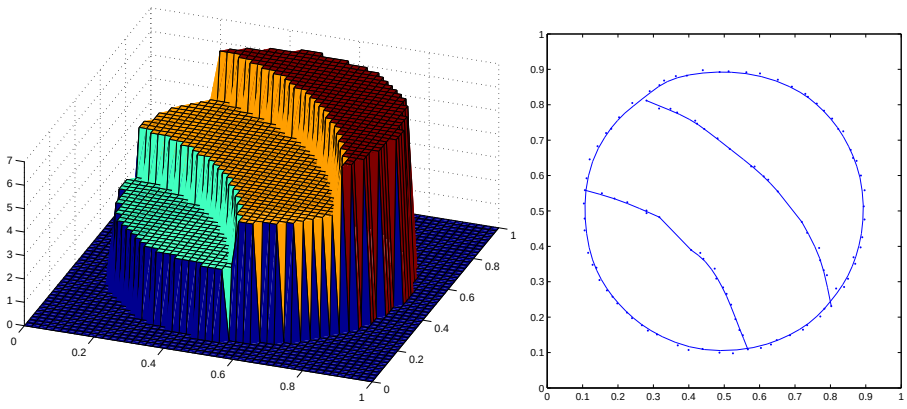

Figure 17: Test function f_1 (left) and polygonal line approximation curves (right).

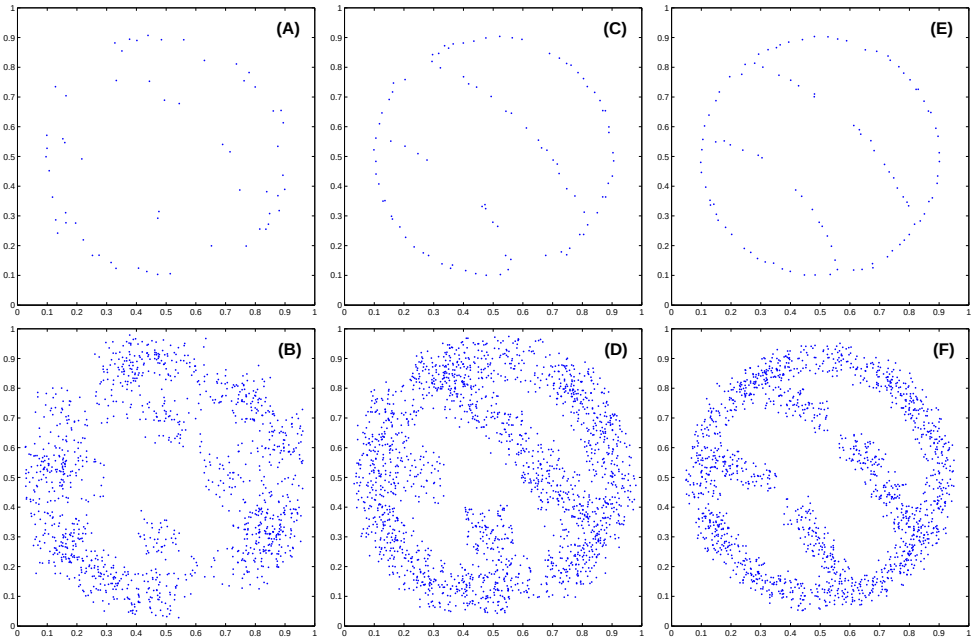

Figure 18: Subsequent enlarged sets of data points (bottom, left to right) and related sets $\mathcal{C}(f;\mathcal{S}_n)$ (top, left to right) in adaptive detection procedure for f_1.

with $l(x) = 0.5 + 0.2\sin(5\pi x)/3$. The fault set is given by

$$\Gamma = \{(t, l(t)) : 0 \le t \le 1\}$$

and its jump size is almost constant on all the domain.

Figure 19 shows the test function f_6 (left) and the fault approximation obtained by the least squares curve from the 49 points of the set $\mathcal{C}(f;\mathcal{S}_n)$ (right), while Figure 20 shows step by step the adaptive detection procedure. Splitting up $\mathcal{C}(f;\mathcal{S}_n)$ in subsets does not produce considerable advantages.

To end the adaptive process, the number of the employed points is 2705.

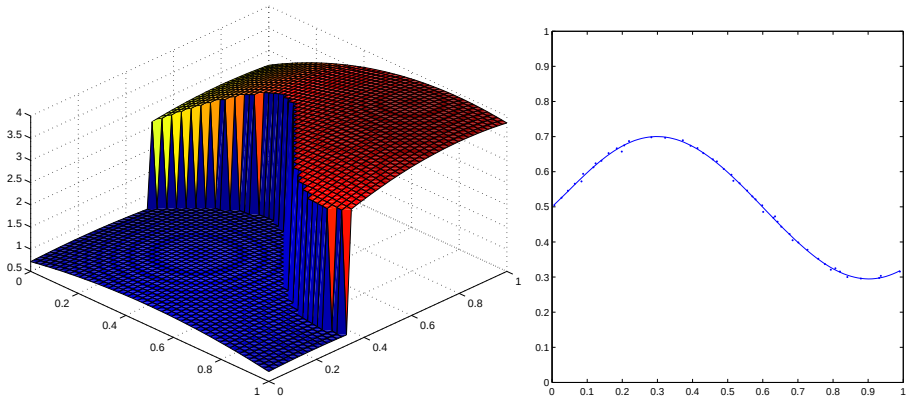

Figure 19: Test function f_6 (left) and least squares approximation curve (right).

Example 8

Last, we take the function f_2 in (12), which is represented in Figure 21. The best l_∞ approximation curves are derived from the 46 points belonging to $\mathcal{C}(f;\mathcal{S}_n)$. Figure 22 shows the output of the adaptive detection process. All these results are obtained, subdividing the $\mathcal{C}(f;\mathcal{S}_n)$ set into two subsets.

The entire detection procedure is completed using 2263 data points.

For each test function, we compute the maximum absolute error (MAE), the maximum relative error (MRE) and the root mean square error (RMSE), evaluating each fault line at 100 equispaced points in $0 \le x \le 1$, as in the previous case. If a fault line is parallel or nearly parallel to the y-axis, we first rotate the coordinate axes and then compute the errors. For simplicity, though polygonal lines could be also expressed analytically as piecewise linear polynomials, we compare only least squares and best l_∞ approximation curves, as it is reported in Table 3 and Table 4.

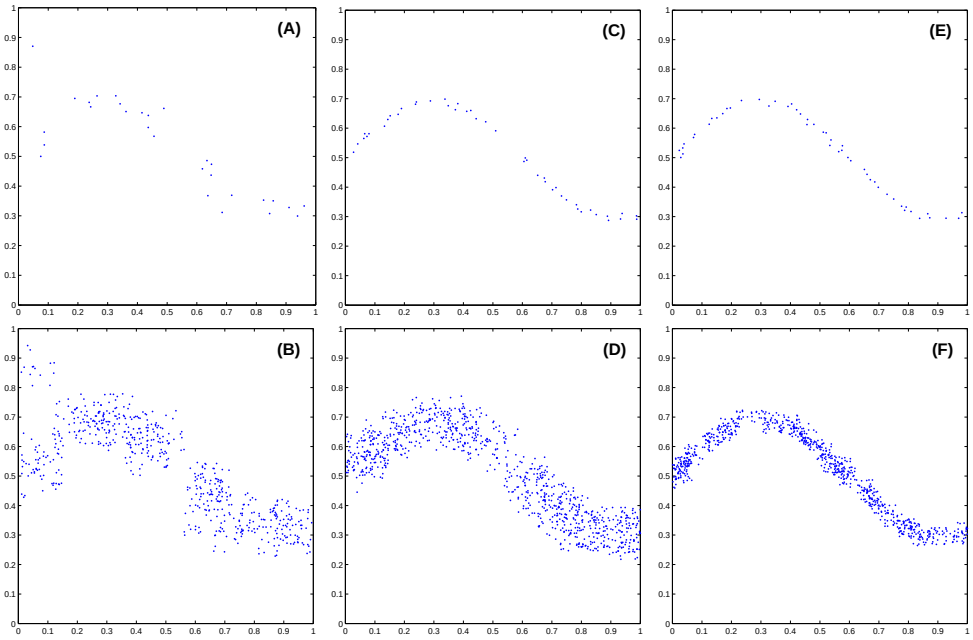

Figure 20: Subsequent enlarged sets of data points (bottom, left to right) and related sets $\mathcal{C}(f;\mathcal{S}_n)$ (top, left to right) in adaptive detection procedure for f_6.

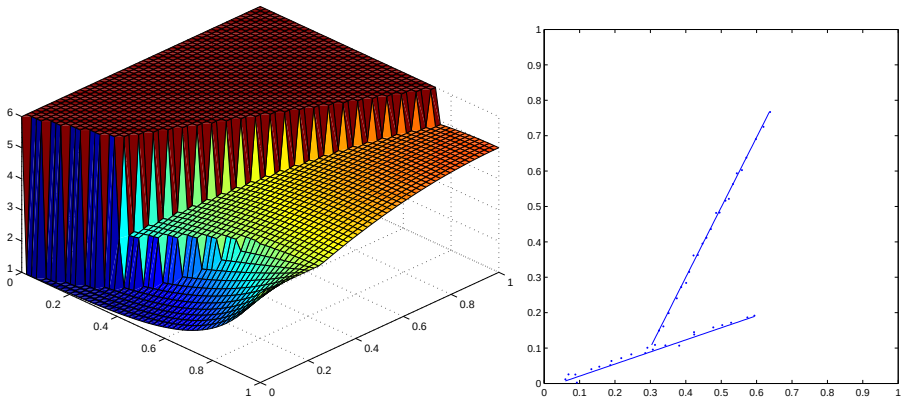

Figure 21: Test function f_2 (left) and best l_∞ approximation curves (right).

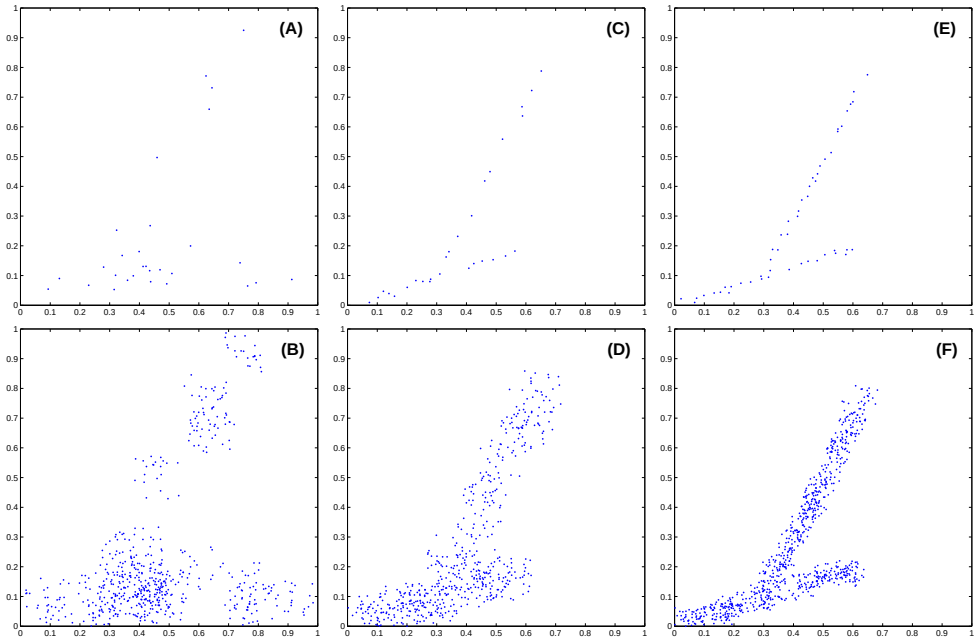

Figure 22: Subsequent enlarged sets of data points (bottom, left to right) and related sets $\mathcal{C}(f;\mathcal{S}_n)$ (top, left to right) in adaptive detection procedure for f_2.

Test Function	MAE	MRE	RMSE
f_1	$4.84865 \cdot 10^{-2}$	$4.56550 \cdot 10^{-1}$	$5.86301 \cdot 10^{-3}$
f_2	$6.30713 \cdot 10^{-3}$	$3.16528 \cdot 10^{-1}$	$3.88514 \cdot 10^{-3}$
f_4	$1.87486 \cdot 10^{-3}$	$3.12477 \cdot 10^{-3}$	$7.29411 \cdot 10^{-4}$
f_5	$3.97532 \cdot 10^{-3}$	$9.93829 \cdot 10^{-3}$	$1.97662 \cdot 10^{-3}$
f_6	$5.35214 \cdot 10^{-3}$	$1.77711 \cdot 10^{-2}$	$2.44892 \cdot 10^{-3}$

Table 3: Errors obtained by the least squares method.

Test Function	MAE	MRE	RMSE
f_1	$5.75570 \cdot 10^{-2}$	$5.08761 \cdot 10^{-1}$	$7.57615 \cdot 10^{-3}$
f_2	$1.29125 \cdot 10^{-2}$	$6.48021 \cdot 10^{-1}$	$7.74472 \cdot 10^{-3}$
f_4	$7.39352 \cdot 10^{-3}$	$1.23225 \cdot 10^{-2}$	$3.02462 \cdot 10^{-3}$
f_5	$7.41478 \cdot 10^{-3}$	$1.23230 \cdot 10^{-2}$	$2.57417 \cdot 10^{-3}$
f_6	$1.67485 \cdot 10^{-2}$	$5.20277 \cdot 10^{-2}$	$5.71694 \cdot 10^{-3}$

Table 4: Errors obtained by the best l_∞ approximation method.

6 Conclusions

In this paper, we presented and analyzed a technique for the detection and the adaptive detection of unknown surface discontinuities in order to obtain an accurate approximation of fault lines from scattered data. The approach we proposed gives good results. In particular, the adaptive technique allows us to reduce the cost, preserving at the same time a good accuracy. Indeed, in some applications, the search of information (data) is very expensive, so that a reduction of cost represents a remarkable advantage.

The detection algorithm, under a reasonable choice of the threshold parameter σ_0, yields the set $\mathcal{A}(f;\mathcal{S}_n)$ of barycentres. This phase allows us to understand the form and the number of faults. The approach is similar but alternative to that proposed in [23], where a local approximation scheme based on thin plate splines combined with a triangulation method is used. Then, from the set $\mathcal{A}(f;\mathcal{S}_n)$ we obtain the set $\mathcal{B}(f;\mathcal{S}_n)$ of barycentres of barycentres, and its ordered version $\mathcal{C}(f;\mathcal{S}_n)$. This last set is necessary to handle complex situations, such as several faults, close type faults and intersections or bifurcations of faults. In these cases $\mathcal{C}(f;\mathcal{S}_n)$ is manually subdivided into a number of subsets greater or equal to the number of discontinuities. Moreover, the set $\mathcal{C}(f;\mathcal{S}_n)$ is basic to apply the proposed approximation methods, namely, polygonal line, least squares, and best l_∞ approximation methods.

For automaticity and simplicity the polygonal line method turns out to be more suitable in almost all applications, even if least squares or best l_∞ approximations produce good or even better results. Moreover, the iterative refinement allows us to construct smooth, planar curves rather than polygonal curves.

When the fault line is a multivalued function, applying least squares and best l_∞ approximations requires to subdivide the data. This situation can create small errors at points in which two pieces of curve join, but the introduction of the iterative refinement reduces considerably the occurrence of these errors of connection. Nevertheless, the decision of how partitioning the domain can hide several difficulties (see [19]).

Finally, the application of this method pointed out at least two problems. If the considered surface shows very rapid variations outside the faults, then the detection algorithm may identify false fault points. Another drawback occurs when the jump size varies fastly or vanishes; here the optimal choice of a global parameter σ_0 is difficult and sometimes it is impossible to detect

correctly the fault points. To solve these problems, we could split up the domain in a uniform grid and find a (local) threshold parameter for each grid cell. This approach should allow us to detect only the actual points of discontinuity, but it is yet an open problem and is still under investigation.

Acknowledgements

The authors are very grateful to the anonymous referee for his/her detailed and valuable comments which helped to greatly improve the paper.

References

[1] G. Allasia, A class of interpolating positive linear operators: theoretical and computational aspects, in: S. P. Singh (Ed.), *Approximation Theory, Wavelets and Applications*, Kluwer Acad. Publ., Dordrecht, 1995, pp. 1–36.

[2] G. Allasia, Cardinal basis interpolation on multivariate scattered data, *Nonlinear Anal. Forum* **6** (2001), 1–13.

[3] G. Allasia, R. Besenghi, A. De Rossi, A scattered data approximation scheme for the detection of fault lines, in: T. Lyche, L. L. Schumaker (Eds.), *Mathematical Methods for Curves and Surfaces*, Vanderbilt Univ. Press, Nashville, TN, 2001, pp. 25–34.

[4] G. Allasia, Simultaneous interpolation and approximation by a class of multivariate positive operators, *Numer. Algorithms* **34** (2003), 147–158.

[5] G. Allasia, Recursive and parallel algorithms for approximating surface data on a family of lines or curves, in: P. Ciarlini et al. (Eds.), *Advanced Mathematical and Computational Tools in Metrology VI*, World Scientific, NJ, 2004, pp. 137–148.

[6] G. Allasia, R. Besenghi, R. Cavoretto, Accurate approximation of unknown fault lines from scattered data, *Mem. Accad. Sci. Torino Cl. Sci. Fis. Mat. Natur.* **33** (2009), 3–26.

[7] G. Allasia, R. Besenghi, R. Cavoretto, Adaptive detection and approximation of unknown surface discontinuities from scattered data, *Simulat. Modell. Pract. Theory* **17** (2009), 1059–1070.

[8] R. Arcangéli, R. Manzanilla, J. J. Torrens, Approximation spline de surfaces de type explicite comportant des failles, *RAIRO Modél. Math. Anal. Numér.* **31** (1997), 643–676.

[9] R. Arcangéli, M. C. López de Silanes, J. J. Torrens, *Multidimensional Minimizing Splines, Theory and Applications*, Kluwer Acad. Publ., Boston, MA, 2004.

[10] E. Arge, M. Floater, Approximating scattered data with discontinuities, *Numer. Algorithms* **8** (1994), 149–166.

[11] I. Barrodale, A. Young, Algorithms for best L_1 and L_∞ linear approximation on a discrete set, *Numer. Math.* **8** (1966), 295–306.

[12] J. L. Bentley, J. H. Friedman, Data structures for range searching, *Comput. Surveys* **11** (1979), 397–409.

[13] R. Besenghi, G. Allasia, Scattered data near-interpolation with application to discontinuous surfaces, in: A. Cohen, C. Rabut, L. L. Schumaker (Eds.), *Curves and Surfaces Fitting*, Vanderbilt Univ. Press, Nashville, TN, 2000, pp. 75–84.

[14] R. Besenghi, M. Costanzo, A. De Rossi, A parallel algorithm for modelling faulted surfaces from scattered data, *Int. J. Comput. Numer. Anal. Appl.* **3** (2003), 419–438.

[15] M. Bozzini, L. Lenarduzzi, Recovering a function with discontinuities from correlated data, in: F. Fontanella, K. Jetter, P.-J. Laurent (Eds.), *Advanced Topics in Multivariate Approximation*, World Scientific, Singapore, 1996, pp. 1–16.

[16] M. Bozzini, M. Rossini, Approximation surfaces with discontinuities, *Math. Comput. Modelling* **31** (2000), 193–213.

[17] M. Bozzini, M. Rossini, Detection of faults and gradient faults from scattered data with noise, in: J. Vigo-Aguiar et al. (Eds.), *Proceedings of the International Conference on Computational and Mathematical Methods in Science and Engineering*, vol. 1, 2009, pp. 189–200.

[18] R. Cavoretto, *Meshfree Approximation Methods, Algorithms and Applications*, PhD Thesis, University of Turin, 2010.

[19] A. Crampton, J. C. Mason, Detecting and approximating fault lines from randomly scattered data, *Numer. Algorithms* **39** (2005), 115–130.

[20] N. P. Fremming, Ø. Hjelle, C. Tarrou, Surface modelling from scattered geological data, in: M. Dæhlem, A. Tveito (Eds.), *Numerical Methods and Software Tools in Industrial Mathematics*, Birkhäuser, Boston, 1997, pp. 301–315.

[21] C. Gout, C. Le Guyader, Segmentation of complex geophysical structures with well data, *Comput. Geosci.* **10** (2006), 361–372.

[22] C. Gout, C. Le Guyader, L. Romani, A. G. Saint-Guirons, Approximation of surfaces with fault(s) and/or rapidly varying data, using a segmentation process, D^m-splines and the finite element method, *Numer. Algorithms* **48** (2008), 67–92.

[23] T. Gutzmer, A. Iske, Detection of discontinuities in scattered data approximation, *Numer. Algorithms* **16** (1997), 155–170.

[24] M. C. López de Silanes, M. C. Parra, J. J. Torrens, Vertical and oblique fault detection in explicit surfaces, *J. Comput. Appl. Math.* **140** (2002), 559–585.

[25] M. C. López de Silanes, M. C. Parra, J. J. Torrens, On a new characterization of finite jump discontinuities and its application to vertical fault detection, *Math. Comput. Simul.* **77** (2008), 247–256.

[26] M. C. Parra, M. C. López de Silanes, J. J. Torrens, Vertical fault detection from scattered data, *J. Comput. Appl. Math.* **73** (1996), 225–239.

[27] R. J. Renka, Multivariate interpolation of large sets of scattered data, *ACM Trans. Math. Software* **14** (1988), 139–148.

[28] M. Rossini, Irregularity detection from noisy data in one and two dimensions, *Numer. Algorithms* **16** (1997), 283–301.

[29] M. Rossini, 2D-discontinuity detection from scattered data, *Computing* **61** (1998), 215–234.

[30] M. Rossini, Detecting discontinuities in two-dimensional signals sampled on a grid, *J. Numer. Anal. Ind. Appl. Math.* **1** (2007), 1–13.

[31] J. Springer, Modeling of geological surfaces using finite elements, in: P.-J. Laurent, A. Le Méhauté, L. L. Schumaker (Eds.), *Wavelets, Images and Surfaces Fitting*, A K Peters, Wellesley, MA, 1994, pp. 467–474.