Scattered and track data interpolation using an efficient strip searching procedure

Scattered and track data interpolation using an efficient strip searching procedure

G. Allasia, R. Besenghi, R. Cavoretto, A. De Rossi*

Department of Mathematics, University of Turin, via Carlo Alberto 10, I–10123 Torino, Italy.

Abstract

A new local algorithm for bivariate interpolation of large sets of scattered and track data is presented. The method, which changes partially depending on the kind of data, is based on the partition of the interpolation domain in a suitable number of parallel strips, and, starting from these, on the construction for any data point of a square neighbourhood containing a convenient number of data points. Then, the well-known modified Shepard's formula for surface interpolation is applied with some effective improvements. The proposed algorithm is very fast, owing to the optimal nearest neighbour searching, and achieves a good accuracy. Computational cost and storage requirements are analyzed. Moreover, the efficiency and reliability of the algorithm are shown by several numerical tests, also performed by Renka's algorithm for a comparison.

Key words: continuous surface modelling, interpolation algorithms, radial basis functions, scattered and track data, Shepard's formulas.
2010 MSC: 65D05, 65D15, 65D17.

1. Introduction

We consider the problem of interpolating a continuous function $f : \mathbb{R}^m \to \mathbb{R}$, $(m \geq 1)$, defined on a finite set $\mathcal{S}_n = \{x_i, i = 1, 2, \ldots, n\}$ of *data points* or *nodes*, which are situated on a domain $D \subset \mathbb{R}^m$. It consists of finding a real multivariate function $F \in C^1(D)$ such that, given the x_i and the corresponding function values f_i, the interpolation conditions $F(x_i) = f_i$, $(i = 1, 2, \ldots, n)$, are satisfied.

In particular, we are interested to consider the interpolation of large scattered data sets, a problem which requires efficient and accurate algorithms. The subject has applications on several areas of the applied sciences, e.g. geophysics or meteorology. In 1988 Renka [27] proposed an optimized implementation of the modified version of Shepard's method, which is still now one of the most powerful tools. Then in 2002, Lazzaro and Montefusco [26] presented a modification of the local Renka's algorithm, in which least squares approximants are replaced by nodal functions based on *radial basis functions* (RBFs), thus improving accuracy.

As a matter of fact, in several applied problems the function values are known along a number of lines or curves, as in the case of ocean-depth measurements from a survey ship or meteorological measurements from an aircraft or an orbiting satellite. These data are affected by measurement errors and, generally, taken near to rather than precisely on straight or curved tracks, owing to the effects of disturbing agents, such as wind and waves. Several methods (see, e.g., [7, 8, 12, 16] and references therein) have been proposed to solve the considered problem by using different approximation techniques (approximation or interpolation, as opposed to approximation) and tools (tensor-product splines, least squares, radial basis functions, Chebyshev polynomials of the first or second kind, etc.). Nevertheless, the non-uniformity and anisotropy in the distribution of this kind of data (often called *track data*), that is, nodes are very close to each other along the tracks, but widely separated between tracks, cause several difficulties for traditional scattered data approximation methods. On the contrary, the solution method and the related algorithm we propose in this paper enable to deal with this setting in a very efficient way.

*Corresponding author.
Email addresses: giampietro.allasia@unito.it (G. Allasia), renata.besenghi@unito.it (R. Besenghi), roberto.cavoretto@unito.it (R. Cavoretto), alessandra.derossi@unito.it (A. De Rossi)

In previous works [4, 6] we gave two different approaches for approximating surface data disposed on a family of (straight) lines or curves on a plane domain. The most interesting case occurs when the lines or curves are parallel. It is possible that some or all the nodes are not collocated exactly on the lines or curves but close to them, or that the lines or curves are not parallel in a proper sense but roughly parallel. Although there is a data structure, it is not required that the node distribution on each line or curve has a special regularity, that is, the nodes can be irregularly spaced and in different positions on each line or curve. The schemes we presented approximate the data by means of interpolation or near-interpolation operators, both based on *cardinal radial basis functions* (CRBFs), whose properties are widely discussed in [1, 2]. In particular, the scheme in [4] has been widely tested in [13], where some interesting devices are presented.

Thus, if we suppose to move the parallel lines or curves as close together as the nodes on different lines or curves, then the track data structure vanishes and the node distribution appears quite irregular on the whole domain D. Conversely, if the nodes are scattered, we can think of partitioning the domain into a convenient number of parallel strips, bounded by parallel lines or curves. Then, we can consider the midlines of the strips as a set of parallel lines or curves, each one having a certain number of nodes on or close to it. Following this idea, we start considering an interpolation scheme for track data and then extend it, in a simple and straightforward way, to interpolation of general sets of scattered data. The outcome is an efficient interpolation algorithm, called *strip algorithm*, for modelling continuous surfaces. The particular strip structure gives some advantages, because it allows us to optimize the searching procedure of nodes and guarantees a high parallelism. In the strip algorithm, first, we partition the domain D into a finite number of parallel strips, ordering all the nodes on each strip with respect to a given direction, which is the same for all strips. Then, we consider a *strip searching procedure* that establishes the minimal number of strips to be examined, in order to localize a convenient set of neighbour nodes for each strip point (i.e. a node lying on a strip). Finally, we approximate the unknown function f using a modified Shepard's interpolant F, which uses moving least squares approximants or radial basis functions as local functions. Numerical results show that the strip algorithm is more efficient than Renka's one (in particular with regard to the execution CPU times), and, at the same time, is comparable in accuracy.

The paper is organized as follows. In Section 2 we recall some properties of CRBIs, referring particularly to Shepard's formula. Section 3 is devoted to briefly remind the well-known local interpolation method, namely the modified Shepard's method, and to consider two ways of constructing nodal functions, that is, the least squares method and the radial basis functions. In Section 4 we describe the strip algorithm, dwelling on the details that allow the procedure to be accurate and computationally efficient. The analysis is divided in four parts: the strip searching method, the strip algorithm, the computational complexity of the proposed algorithm, and some numerical results considering the scattered data interpolation. Section 5 follows a similar subdivision but now track data interpolation is studied. In particular, numerical comparisons with Renka's algorithm are presented in both cases. In Section 6 a subdivision procedure for Shepard's type interpolants is pointed out, and we sketch a parallel version of the algorithm, discussing some practical aspects. Section 7 deals with final remarks and future work. Finally, in Appendix A, a brief recall of Renka's algorithm is given for greater convenience of the reader and in order to make clearer the comparison with the strip algorithm.

2. Cardinal radial basis interpolation

Scattered data multivariate approximation and, in particular, radial basis interpolation, are widely studied subjects. In the literature many results can be found, either about theoretical properties of radial basis functions or about their computational aspects (see [11, 24, 33, 18]). A different approach enables to obtain cardinal radial basis interpolation operators, which do not require solving large linear systems (possibly, badly conditioned) and enjoy interesting properties; in particular, they are weighted arithmetic means of the function values [1, 2].

Given a set $\mathcal{S}_n = \{x_i, i = 1, 2, \ldots, n\}$ of distinct nodes, arbitrarily distributed in a domain $D \subset \mathbb{R}^m$, $(m \geq 1)$, and a set $\mathcal{F}_n = \{f_i, i = 1, 2, \ldots, n\}$ of associated real values, to solve the classical interpolation problem, one can consider basis functions which depend on the nodes and, moreover, are cardinal. A cardinal radial basis interpolant is obtained selecting continuous cardinal functions $g_k : D \to \mathbb{R}$, $(k = 1, 2, \ldots, n)$, such that

$$g_k(x) \geq 0, \qquad \sum_{k=1}^{n} g_k(x) = 1, \qquad g_k(x_i) = \delta_{ki}, \qquad i = 1, 2, \ldots, n,$$

where δ_{ki} is the Kronecker delta, and setting up the interpolant F (or $F(x; f, \mathcal{S}_n)$, if appropriate) in the form

$$F(x) = \sum_{k=1}^{n} f_k \, g_k(x).$$

We focus on a wide class of interpolation operators, specified by a constructive procedure suggested by Cheney [14] (see also [15]).

Definition 2.1. *Let $\alpha(x, y)$, with $x, y \in D$, be a continuous real function such that*

$$\alpha(x, y) > 0, \qquad \text{if } x \neq y; \qquad \alpha(x, x) = 0; \qquad \forall x, y \in D. \tag{1}$$

Define the functions g_j by

$$g_j(x) = \frac{\prod_{k=1, k\neq j}^{n} \alpha(x, x_k)}{\sum_{j=1}^{n} \prod_{k=1, k\neq j}^{n} \alpha(x, x_k)},$$

and the interpolant F by

$$F(x) = \sum_{j=1}^{n} f_j \, g_j(x) = \sum_{j=1}^{n} f_j \, \frac{\prod_{k=1, k\neq j}^{n} \alpha(x, x_k)}{\sum_{j=1}^{n} \prod_{k=1, k\neq j}^{n} \alpha(x, x_k)}, \tag{2}$$

or, equivalently, by

$$F(x) = \sum_{j=1}^{n} f_j \, \frac{1/\alpha(x, x_j)}{\sum_{j=1}^{n} 1/\alpha(x, x_j)}, \qquad F(x_i) = f(x_i), \qquad i = 1, 2, \ldots, n. \tag{3}$$

Among all possible choices for the function $\alpha(x, y)$, it is natural to identify $\alpha(x, y)$ with a function of a distance $d(x, y)$ defined on $D \subset \mathbb{R}^m$, i.e.,

$$\alpha(x, y) = \phi(d(x, y)).$$

Since the distance is a nonnegative real number, it is often convenient to consider the univariate function $\phi : \mathbb{R}_{\geq 0} \to \mathbb{R}$ associated to $\phi(d(x, y))$. If $d(x, y)$ is the Euclidean metric $\|x - y\|_2$, then the interpolant F becomes independent of Euclidean transformations of the data set. As a significant example of distance-weighted interpolation, we consider *Shepard's formula* [32], whose *barycentric form* is

$$S(x) = \sum_{j=1}^{n} f_j \, \frac{\|x - x_j\|_2^{-p}}{\sum_{j=1}^{n} \|x - x_j\|_2^{-p}}, \qquad S(x_i) = f_i, \qquad i = 1, 2, \ldots, n, \tag{4}$$

where $p > 0$. The case $p = 2k$, ($k \in \mathbb{N}$), is remarkable, because the basis functions are infinitely differentiable (see, for example, [9, 23]). The original formula (4) can be rewritten in the equivalent *product form*

$$S(x) = \sum_{j=1}^{n} f_j \, \frac{\prod_{k=1, k\neq j}^{n} \|x - x_k\|_2^{p}}{\sum_{j=1}^{n} \prod_{k=1, k\neq j}^{n} \|x - x_k\|_2^{p}},$$

which is more suitable in order to discuss analytic properties. Shepard's formula is not only invariant under translations and rotations, but also scale invariant.

3. Localizing interpolation schemes

The classical Shepard's formula has two crucial drawbacks, namely the occurrence of flat spots at the nodes (i.e., the partial derivatives vanish there) and the dependence of the operator on all the nodes. To avoid these shortcomings, a modified version of Shepard's method has been developed by Franke and Nielson [21], and then improved by Renka [27]. An interesting modification has been suggested by Lazzaro and Montefusco [26].

Here we consider the following definition of the modified Shepard's method.

Definition 3.1. *Given a set $\mathcal{S}_n = \{x_i, i = 1, 2, \ldots, n\}$ of distinct nodes, arbitrarily distributed in a domain $D \subset \mathbb{R}^m$, with the corresponding set $\mathcal{F}_n = \{f_i, i = 1, 2, \ldots, n\}$ of associated values of an unknown real function $f : D \to \mathbb{R}$, the modified Shepard's method $\tilde{F} : D \to \mathbb{R}$ takes the form*

$$\tilde{F}(x) = \sum_{j=1}^{n} L_j(x)\bar{W}_j(x). \tag{5}$$

The nodal functions $L_j(x)$, $(j = 1, 2, \ldots, n)$, are local approximants to f at x_j, constructed on the n_L nodes closest to x_j and satisfying the interpolation conditions $L_j(x_j) = f_j$. The weight functions $\bar{W}_j(x)$, $(j = 1, 2, \ldots, n)$, are given by

$$\bar{W}_j(x) = \frac{W_j(x)}{\sum_{k=1}^{n} W_k(x)}, \qquad j = 1, 2, \ldots, n,$$

where

$$W_j(x) = \tau(x, x_j)/\alpha(x, x_j),$$

$\tau(x, x_j)$ being a nonnegative localizing function, and $\alpha(x, x_j) = \|x - x_j\|_2^2$.

As regard to the choice of nodal functions we consider two possible ways. The former is obtained by solving the *least squares problem* at the node x_j using weights with reduced compact support, that is,

$$\min_{a_j} \sum_{i=1, i\neq j}^{n_L} \left[L_j(x_i) - f_i\right]^2 w_i(x_j),$$

where L_j is a quadratic m-variate polynomial with coefficients $a_j = (a_{j1}, a_{j2}, \ldots, a_{jh})^T$, h is less than the number n_L of nodes of the considered neighbourhood of x_j, and $w_i(x_j) = \tau(x_j, x_i)/\alpha(x_j, x_i)$ is a nonnegative weight function.

The latter is the most used interpolant on scattered data (see [11, 33]), and has the form

$$L_j(x) = \sum_{i=1}^{n_L} a_i\varphi(\|x - x_i\|_2) + \sum_{k=1}^{U} b_k\pi_k(x), \tag{6}$$

where the radial basis functions $\varphi(\|x - x_i\|_2)$ depend on the n_L nodes of the considered neighbourhood of x_j, and the space $\mathcal{P}^m_{v-1}$ spanned by the $(v-1)$-degree polynomials $\pi_k(x)$ has a dimension $U = (m + v - 1)!/(m!(v - 1)!)$ which must be lower than n_L. It is required that L_j satisfies the interpolation conditions

$$L_j(x_j) = f_j, \qquad j = 1, 2, \ldots, n_L,$$

and the side conditions

$$\sum_{i=1}^{n_L} a_i\pi_k(x_i) = 0, \qquad \text{for } k = 1, 2, \ldots, U.$$

Hence, to compute the coefficients $a = (a_1, a_2, \ldots, a_{n_L})^T$ and $b = (b_1, b_2, \ldots, b_U)^T$ in (6), it is required to solve uniquely the system of linear equations

$$\begin{aligned} Ka + Pb &= f, \\ P^T a &= 0, \end{aligned}$$

where $K = \{\varphi(\|x_j - x_i\|_2)\}$ is a $n_L \times n_L$ matrix, $P = \{\pi_k(x_j)\}$ is a $n_L \times U$ matrix, f denotes the column vector of the k-th coordinate of the function values f_j corresponding to the x_j, and the equation $P^T a = 0$ represents the boundary conditions.

The most popular choices for φ are

$$\begin{aligned} \varphi(r) &= r^{2v-m}\log r, \quad 2v - m \in 2\mathbb{N}, && \text{(generalized thin plate spline)} \\ \varphi(r) &= \mathrm{e}^{-\beta r^2}, && \text{(Gaussian)} \\ \varphi(r) &= (r^2 + \gamma^2)^{v/2}, && \text{(generalized multiquadric)} \end{aligned}$$

where β, γ are positive constants, v is an integer number and $r = \|x - x_i\|_2$. The Gaussian and the inverse multiquadric (IMQ), which occurs for $v < 0$ in the generalized multiquadric function, are positive definite functions, and this guarantees the existence of a unique solution of the considered system. Otherwise, the thin plate spline (TPS) and the multiquadric (MQ), i.e. for $v > 0$ in the generalized multiquadric function, are conditionally positive definite functions of order v, which require the addition of a polynomial term of order $v - 1$ together with side conditions in order to obtain an invertible interpolation matrix.

Since Shepard's interpolant in (4) depends on all the data, when the number of data is very large, the evaluation becomes proportionately longer and, eventually, the method will become inefficient or impractical. So for the weights in (5) we propose the use of different localizing functions $\tau(x)$. A first simple but efficient localizing function with compact support, often called *step function*, is given by

$$\tau_1(x, x_j) = \begin{cases} 1, & \text{if } x \in C(x_j; s), \\ 0, & \text{otherwise,} \end{cases}$$

where $C(x_j; s)$ is a hypercube of centre at x_j and side s.

Otherwise, several authors have suggested localizing schemes in such a way as to obtain a weighting function which is zero outside some disk of suitable radius centred at each node (see [21]). Hence, to localize $F(x)$ one can multiply the functions $1/\alpha(x, x_j)$ in (3) by the so-called *mollifying functions* $\tau_2(x, x_j), (j = 1, 2, \ldots, n)$, which are nonnegative, have local supports in some appropriate sense, and satisfy $\tau_2(x_j, x_j) = 1$, as for example the truncated power function

$$\tau_2(x, x_j) = (1 - \|x - x_j\|_2^2 / r_j)_+, \tag{7}$$

where r_j is the radius of the circle of support at the node x_j.

An interesting alternative to (7) is offered by the function

$$\tau_3(t) = \begin{cases} -2^{(3q)}t^3 + 3 \cdot 2^{(2q)}t^2 - 3 \cdot 2^q t + 1, & \text{if } 0 \le t \le 1/2^q, \\ 0, & \text{if } t > 1/2^q, \end{cases}$$

where $t = \|x - x_j\|_2^2$. In fact, we have $\tau_3(0) = 1$ and $\tau_3(1/2^q) = 0$; the function is convex and its tangent plane at $t = 1/2^q$ is horizontal; the localizing effect increases with q. Localizing functions like $\tau_3(t)$, possibly with different orders of continuity, may represent an alternative choice to the families of localizing functions based on cutoff functions (see, for example, [31]).

Among possible choices it is also convenient to localize the method considering a factor $\tau_4(x, x_j)$ rapidly decreasing with distance. The formulas obtained in this way maintain, in general, the analytical and computational properties of the corresponding original ones. The use of the exponential-type weight function

$$\tau_4(x, x_j) = \exp\left(-\gamma \|x - x_j\|_2^2\right), \qquad \gamma \ge 0,$$

yields much more accurate results, and the increased computational effort can generally be tolerated.

4. Scattered data interpolation algorithm

In this section, we consider the problem of approximating a function $f : D \to \mathbb{R}$, $D = [0, 1] \times [0, 1] \subset \mathbb{R}^2$, only known on a set $\mathcal{S}_n = \{(x_i, y_i), i = 1, 2, \ldots, n\}$ of distinct and scattered nodes, which has $\mathcal{F}_n = \{f_i, i = 1, 2, \ldots, n\}$ as the set of corresponding function values. The method and the relative algorithm could be extended in a straightforward way to more general domains D. Our aim is to describe a local interpolation algorithm, called strip algorithm, which is accurate and, at the same time, computationally efficient if compared with those known in the literature. Therefore, we propose a comparison between the strip algorithm and Renka's algorithm [27] (see Appendix A), which is currently considered as a standard procedure.

4.1. Strip searching method

In this subsection we extensively explain our searching method, dwelling on the details that allow the procedure to be accurate and computationally efficient.

The main idea is to construct a suitable family of q strips of equal width (with possible exception of one of them) on the domain D, and parallel to x-axis, so that the set $\mathcal{S}_n$ of nodes is partitioned by the strip structure into q subsets $\mathcal{S}_{n_k}$, whose elements are $(x_{k1}, y_{k1}), (x_{k2}, y_{k2}), \ldots, (x_{kn_k}, y_{kn_k})$, $(k = 1, 2, \ldots, q)$. Then, the nodes of each $\mathcal{S}_{n_k}$ are ordered with respect to the positive direction of the x-axis, by applying a classical sorting procedure. This phase will be described in detail in *Algorithm 1* in Subsection 4.2 (see `Stage 4`).

Let us consider now the construction of local neighbourhoods (in general, rectangular or, more simply, square) for all nodes. These neighbourhoods are to be used in the computation of coefficients in the least squares or RBF nodal functions $L_j(x, y)$, $(j = 1, 2, \ldots, n)$. The choice of each local (square) neighbourhood size is carried out automatically in relation to the sample dimension n, the parameter value n_L, and the positive integer k_1, as it is detailed by (9) in `Stage 2`. To localize the nodes closest to each strip point, we establish the minimal number of strips to be examined, because a node belonging to a strip can be closer to nodes in nearby strips than to those in the same strip. The strategy to define the minimal number of strips is given directly by the neighbourhood half-size δ_y^L (see `Stage 2` and `Stage 3`).

In practice, the search of the nearby nodes is limited at most to three strips: the strip on which the considered node lies, the previous and the next strips (see *Algorithm 2* in `Stage 5` for more details). To satisfy these requests, we explain, first, a way for choosing a suitable size of the neighbourhood, and then a rule for finding which and how many strips to examine.

(1) The size of local square neighbourhoods is found so that, supposing a uniform distribution of nodes on the domain D, each neighbourhood has a prefixed number of nodes. The condition is satisfied, by taking into account the sample dimension n, the parameter n_L (or n_W), and the positive integer k_1 (or k_2). In particular, the rule (9) (or (10) in `Stage 8`) estimates for $k_1 = 1$ (or $k_2 = 1$), at least, $4n_L$ (or $4n_W$) nodes for each inner neighbourhood. If a node lies on or close to the boundary, the number of nodes in its neighbourhood may be considerably reduced, because only a little part of the neighbourhood intersects the domain D (see Fig. 1). However, the approach we propose is completely automatic, since the procedure locates the minimal positive integer k_1 (or k_2) meeting the requirement of having a sufficient number of nodes on each neighbourhood. This implies that the algorithm works successfully even if the distribution of nodes is not uniform.

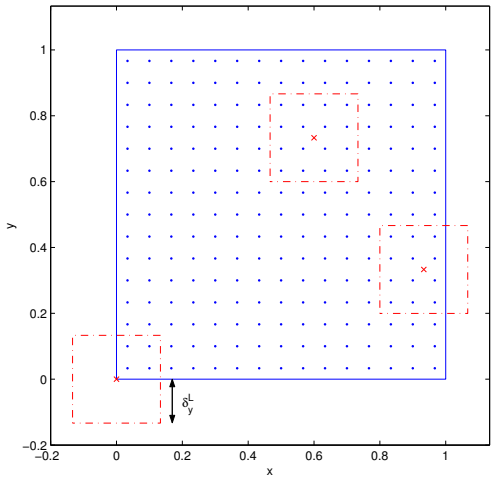

Figure 1: Example of square neighbourhoods with $k_1 = 1$, $n_L = 4$, $n = 225$.

(2) To identify the strips to be examined in the searching procedure, we adopt the following rule which is composed of three steps:

1. The width δ_s of a strip is chosen equal to the half-size δ_y^L of the square neighbourhood, i.e. $\delta_s \equiv \delta_y^L$, and the ratio between these quantities is denoted by

$$i^* = \frac{\delta_y^L}{\delta_s}.$$

2. The value i^* provides the number j^* of strips to be examined for each node by the rule

$$j^* = 2i^* + 1, \tag{8}$$

which obviously here gives $j^* = 3$.

3. For each strip s_k, $(k = 1, 2, \ldots, q)$, a strip searching procedure is considered, examining the nodes from strip $s_k - i^*$ to strip $s_k + i^*$ (see `Stage 5`).

Note that for the nodes of the "first" and "last" strips, in general, we need to reduce the total number of strips to be examined, because if $s_k - i^* < 1$ or $s_k + i^* > q$ it will assign $s_k - i^* = 1$ and strip $s_k + i^* = q$, respectively.

4.2. Strip algorithm

The strip algorithm can be described as follows.

INPUT: n, number of data; $\mathcal{S}_n = \{(x_i, y_i), i = 1, 2, \ldots, n\}$, set of data points; $\mathcal{F}_n = \{f_i, i = 1, 2, \ldots, n\}$, set of data values; s, number of grid points; $\mathcal{G}_s = \{(x_i, y_i), i = 1, 2, \ldots, s\}$, set of grid points; n_L and n_W, localization parameters.

OUTPUT: $\mathcal{A}_s = \{F_i \equiv F(x_i, y_i), i = 1, 2, \ldots, s\}$, set of approximated values.

`Stage 1.` The nodes in the domain D are ordered with respect to a common direction (e.g. the y-axis), by applying a *quicksort$_y$ procedure*.

`Stage 2.` For each node (x_i, y_i), $(i = 1, 2, \ldots, n)$, a local (square) neighbourhood is constructed, whose half-size depends on the sample dimension n, the considered value n_L, and the positive integer k_1, i.e.

$$\delta_x^L = \delta_y^L = \sqrt{k_1 \frac{n_L}{n}}, \qquad k_1 = 1, 2, \ldots \tag{9}$$

`Stage 3.` After the number of strips to be considered is found taking

$$q = \left\lceil \frac{1}{\delta_y^L} \right\rceil,$$

the strips are numbered from 1 to q.

`Stage 4.` The set $\mathcal{S}_n$ is partitioned into q subsets $\mathcal{S}_{n_k}, (k = 1, 2, \ldots, q)$, so that the nodes of $\mathcal{S}_{n_k}$ belong to the k-th strip. Moreover, all the nodes of $\mathcal{S}_{n_k}$, i.e., $(x_{k1}, y_{k1}), (x_{k2}, y_{k2}), \ldots, (x_{kn_k}, y_{kn_k})$, are ordered with respect to a common direction (e.g. the x-axis) on all strips by a *quicksort$_x$ procedure*, and at the same time counted. The number of nodes in the k-th strip is stored in n_k (see *Algorithm 1*).

`Stage 5.` After defining which and how many strips are to be examined, a strip searching procedure is applied for each node of $\mathcal{S}_{n_k}$, $(k = 1, 2, \ldots, q)$, to determine all nodes belonging to a (local) neighbourhood. The number of nodes of the neighbourhood centred at (x_i, y_i) is counted and stored in m_i, $(i = 1, 2, \ldots, n)$, (see *Algorithm 2*). Here we check whether the number of nodes in each neighbourhood is greater or equal to n_L, and if this condition is not satisfied we return back to `Stage 2`.

`Stage 6.` All the nodes belonging to a square neighbourhood centred at (x_i, y_i), $(i = 1, 2, \ldots, n)$, are first ordered by applying a based-distance sorting process, that is a *quicksort$_d$ procedure*, and then reduced to n_L.

`Stage 7.` Taking only the n_L nodes closest to the centre (x_i, y_i), $(i = 1, 2, \ldots, n)$, of the neighbourhood, a local interpolant $L_j(x, y)$, $(j = 1, 2, \ldots, n)$, is found for each node.

Algorithm 1: sorting procedure.

STEP 1 Set $count = 0$.
STEP 2 For $k = 1, 2, \ldots, q$ do
STEP 3 Set $n_k = 0$;
$i = count + 1$.
STEP 4 While $(y_i \leq k \cdot \delta_s \wedge i \leq n)$
set $n_k = n_k + 1$;
$count = count + 1$;
$i = i + 1$.
STEP 5 Set $begin\_strip_k = count - n_k + 1$;
$end\_strip_k = count$.
STEP 6 Compute $quicksort_x(n_k, x, y, f)$.
STEP 7 OUTPUT(n_k, x, y, f).

Algorithm 2: strip searching procedure.

STEP 1 For $k = 1, 2, \ldots, q$ do
STEP 2 Set $begin = k - i^*$;
$end = k + i^*$.
STEP 3 If $begin < 1$
then set $begin = 1$;
If $end > q$
then set $end = q$.
STEP 4 For $h = begin\_strip_k, \ldots, end\_strip_k$ do
STEP 5 Set $m_h = 0$.
STEP 6 For $i = begin, \ldots, end$ do
STEP 7 For $j = begin\_strip_i, \ldots, end\_strip_i$ do
STEP 8 If $(x_j, y_j) \in I_h(\delta_x^L, \delta_y^L)$
then set $m_h = m_h + 1$;
$STORE_{h,m_h}(x_j, y_j, f_j)$.
STEP 9 OUTPUT$((x, y, f) \in I_h(\delta_x^L, \delta_y^L))$.

`Stage 8.` For each grid point $(x, y) \in \mathcal{G}_s$, a square neighbourhood is constructed, whose half-size depends on the sample dimension n, the parameter value n_W, and the (positive integer) number k_2, that is,

$$\delta_x^W = \delta_y^W = \sqrt{k_2 \frac{n_W}{n}}, \qquad k_2 = 1, 2, \ldots \tag{10}$$

`Stage 9.` A searching procedure is applied to determine all nodes belonging to a (local) neighbourhood of centre (x, y) and half-side δ_x^W. Then, we check if the number of nodes in each neighbourhood is greater or equal to n_W; if the condition is not satisfied we return back to `Stage 8`.

`Stage 10.` The nodes of each neighbourhood are first ordered by applying a based-distance sorting procedure ($quicksort_d$), and then reduced to n_W.

`Stage 11.` Considering only the n_W nodes closest to the grid point (x, y), it is found a local weight function $\bar{W}_j(x, y)$, $(j = 1, 2, \ldots, n)$.

`Stage 12.` Applying the modified Shepard's formula (5), the surface can be approximated at any grid point $(x, y) \in \mathcal{G}_s$.

4.3. Computational complexity

The computational complexity of the strip algorithm is characterized by the employment of the standard sorting routine `quicksort`, which requires on average a time complexity $O(M \log M)$, where M is the number of nodes to be sorted. More precisely, we have a preprocessing phase (i.e., for building the data structure), in which the computational cost has order: $O(n \log n)$ for the first sorting of all n nodes (see `Stage 1`); $O(m_i \log m_i)$, $i = 1, 2, \ldots, n$, to sort the nodes in the $i-$ th square neighbourhood (`Stage 6`) and, since $m_i \geq n_L$, for all neighbourhoods we have $\sum_{i=1}^{n} O(m_i \log m_i) \geq n \cdot O(n_L \log n_L)$. Moreover, the corresponding n linear systems of dimension n_L are solved in order to compute the coefficients of the local interpolants, thus requiring $O(n \cdot n_L^3/6)$ and $O(n \cdot 5^3/6)$ arithmetic operations for computing RBF interpolants and least squares approximants, respectively (see `Stage 7`). Conversely, in the evaluation phase we need of a computational cost of order $s \cdot O(n_W \log n_W)$ (see `Stage 9`) and $O(n_W \cdot n_L)$ for each evaluation point, that is $s \cdot O(n_W \cdot n_L)$.

We remark that when the data structure is built, no search time is required, since all points are stored in an ordered sequence. In particular, we point out that in our algorithm the number of nodes needed in each neighbourhood is prescribed (n_L and n_W in the two phases), that is the data structure is built in such a way that exactly n_L and n_W nodes belong to each neighbourhood.

Finally, in the strip algorithm we used $3 \cdot n \cdot n_L$ and $3 \cdot s \cdot n_W$ storage locations in the building of the data structure for the localization of nodal functions and Shepard's interpolant, respectively.

4.4. Numerical results

In this subsection we summarize the extensive and detailed investigation we performed to test and verify the proposed algorithm, especially in view of comparison with Renka's one. In order to obtain numerical validation of the strip algorithm we implemented our procedure in *C/C++* language and used *MATLAB* environment to draw some pictures. All the numerical results were obtained on a Pentium IV computer (2.8 GHz).

In the various tests we considered n randomly scattered nodes (x_i, y_i) in the square $[0, 1] \times [0, 1] \subset \mathbb{R}^2$, and the corresponding function values f_i, for $i = 1, 2, \ldots, n$. The pseudorandom nodes were obtained by using the *MATLAB* command `rand`, which generates uniformly distributed random numbers on the interval (0,1). Since the strip and Renka's algorithms are designed to interpolate to large scattered data sets, in an accurate and efficient way, we considered sets of dimension $n = 1000, 2000, 4000, 8000, 16000$. However, it is remarkable that, in general, also reducing considerably (e.g., to a few thousand nodes) the dimension n of the scattered data set, the proposed algorithm 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 nodes. In Fig. 2 the set of $n = 1000$ nodes is plotted, as an example.

We choose from the literature some well-tried test functions, in order to verify the performance of our algorithm: Franke's test functions f_1 (see [19, 20, 30, 26]), f_2 and f_3 (see [30, 26] and [30], respectively), and Nielson's test function f_4 (see [22]). The analitic expressions of such functions are:

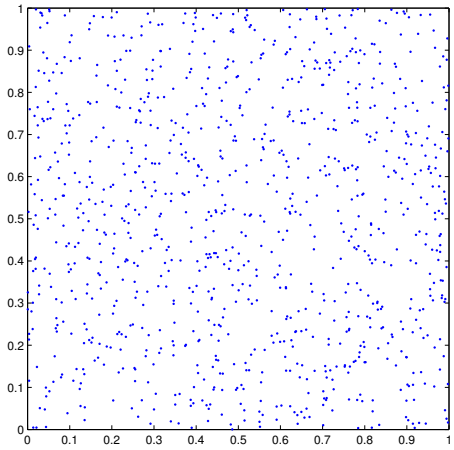

Figure 2: Plot of a scattered data point set ($n = 1000$).

$$f_1(x,y) = \frac{3}{4}\exp\left[-\frac{(9x-2)^2+(9y-2)^2}{4}\right] + \frac{3}{4}\exp\left[-\frac{(9x+1)^2}{49} - \frac{9y+1}{10}\right]$$
$$+\frac{1}{2}\exp\left[-\frac{(9x-7)^2+(9y-3)^2}{4}\right] - \frac{1}{5}\exp\left[-(9x-4)^2-(9y-7)^2\right],$$

$$f_2(x,y) = 2\cos(10x)\sin(10y) + \sin(10xy),$$

$$f_3(x,y) = \exp\left[-\frac{(5-10x)^2}{2}\right] + 0.75\exp\left[-\frac{(5-10y)^2}{2}\right] + 0.75\exp\left[-\frac{(5-10x)^2}{2}\right]\exp\left[-\frac{(5-10y)^2}{2}\right],$$

$$f_4(x,y) = \frac{1}{2}y\cos^4\left[4\left(x^2+y-1\right)\right].$$

The graphs of the test functions are presented in Fig. 3 and 4.

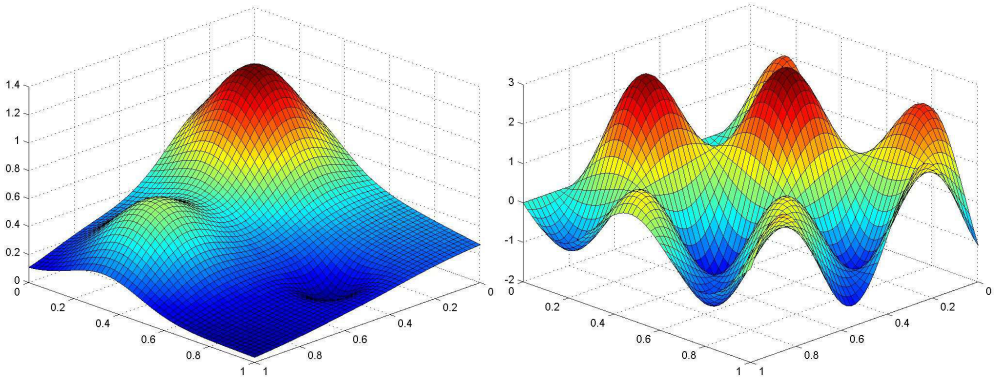

Figure 3: Test functions $f_1(x,y)$ and $f_2(x,y)$.

Renka's algorithm has been cleaned by all instructions which are unnecessary to the interpolant evaluation (as for example the evaluation of the interpolant derivatives), thus obtaining an algorithm comparable with the strip one. The comparison was performed using in the strip algorithm the localizing function τ_3. We extensively tested the choice of the localizing parameters n_L and n_W, finding good results for $n_L = 13$ and $n_W = 8$. Other choices are possible, since

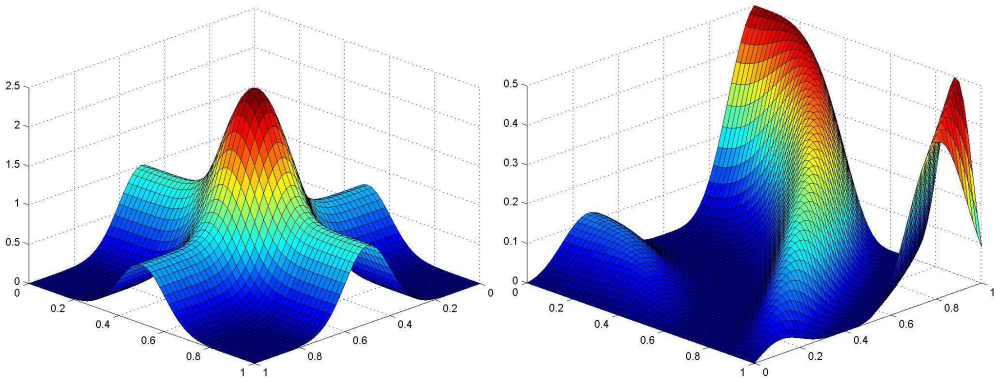

Figure 4: Test functions $f_3(x, y)$ and $f_4(x, y)$.

they depend on the behaviour of the test function, the node distribution and the dimension of the scattered data point set.

The Root Mean Square Errors (RMSEs) and the Maximum Absolute Errors (MAEs) were computed by evaluating the interpolants on $s = 51 \times 51$ grid points. In Tables 1, 2, 3, and 4 we summarized the results of the numerical experiments performed on the four test functions.

n	1000	2000	4000	8000	16000
Renka's algorithm	7.3734E – 4 1.6781E – 2	1.9363E – 4 3.3557E – 3	5.8998E – 5 7.5528E – 4	2.7634E – 5 5.4467E – 4	8.8848E – 6 1.8245E – 4
Strip algorithm	7.3027E – 4 1.3089E – 2	2.5886E – 4 4.3607E – 3	6.5430E – 5 6.1721E – 4	2.7545E – 5 4.9754E – 4	9.3833E – 6 1.7589E – 4

Table 1: RMSEs and MAEs for the function f_1.

n	1000	2000	4000	8000	16000
Renka's algorithm	7.3097E – 3 4.9833E – 2	3.3367E – 3 3.4978E – 2	1.6086E – 3 3.3807E – 2	4.7726E – 4 6.4474E – 3	2.1094E – 4 5.6340E – 3
Strip algorithm	1.0284E – 2 1.7549E – 1	3.2741E – 3 4.5686E – 2	1.5193E – 3 3.1865E – 2	5.0732E – 4 1.0625E – 2	2.2921E – 4 6.7940E – 3

Table 2: RMSEs and MAEs for the function f_2.

It appears that the two methods are comparable in accuracy. This is not astonishing, because the methods are very similar, both being modifications of Shepard's method in which nodal functions are given by least squares approximants. The slight differences we found in errors are given by the different choices of the nearest neighbours: Renka's algorithm works with circular neighbourhoods, while the strip one with square neighbourhoods. Moreover, the strip algorithm uses τ_3 for the weights, while Renka's algorithm employs different localizing functions.

In order to improve accuracy we also considered in the modified Shepard's formula nodal functions constructed by radial basis functions. Errors obtained with such interpolation scheme are listed in Tables 5, 6, 7, and 8. The improvement is considerable, since the errors go down of one or two order of magnitude. This result is given by the better convergence order achieved by radial basis functions in comparison with least squares approximants. The

n	1000	2000	4000	8000	16000
Renka's algorithm	2.5745E – 3 4.1001E – 2	9.0061E – 4 1.0284E – 2	4.5809E – 4 1.3655E – 2	3.2302E – 4 1.4885E – 2	4.0059E – 5 4.9019E – 4
Strip algorithm	2.4492E – 3 3.6862E – 2	9.7451E – 4 1.5808E – 2	2.8984E – 4 4.7287E – 3	1.2813E – 4 2.5127E – 3	3.5490E – 5 5.0930E – 4

Table 3: RMSEs and MAEs for the function f_3.

n	1000	2000	4000	8000	16000
Renka's algorithm	2.1357E – 3 4.1188E – 2	8.1067E – 4 1.2285E – 2	3.0900E – 4 5.8845E – 3	1.1585E – 4 2.7125E – 3	4.3877E – 5 9.9668E – 4
Strip algorithm	4.1101E – 3 1.5478E – 1	8.3175E – 4 1.6464E – 2	3.0855E – 4 6.1582E – 3	1.1788E – 4 1.8115E – 3	6.3401E – 5 2.1948E – 3

Table 4: RMSEs and MAEs for the function f_4.

values of the shape parameters in RBFs were chosen to be $v = 2, \beta = 10$, and $\gamma^2 = 0.1$, and we defined interpolants so that positive definiteness was guaranteed.

RBF / n	1000	2000	4000	8000	16000
TPS	1.6251E – 3 2.4967E – 2	7.2136E – 4 6.8620E – 3	4.1299E – 4 9.4118E – 3	2.0214E – 4 3.6454E – 3	1.0336E – 4 1.3058E – 3
Gaussian	1.6005E – 4 3.7529E – 3	2.3769E – 5 4.1933E – 4	7.5109E – 6 1.2335E – 4	1.8931E – 6 3.4183E – 5	4.9257E – 7 9.1839E – 6
MQ	1.0290E – 4 2.1801E – 3	2.2912E – 5 4.5492E – 4	5.6799E – 6 9.8712E – 5	1.6311E – 6 3.4742E – 5	4.6664E – 7 8.7795E – 6
IMQ	9.0126E – 5 1.1556E – 3	2.6664E – 5 4.2676E – 4	8.5316E – 6 2.4795E – 4	1.9708E – 6 4.5244E – 5	5.6279E – 7 1.3166E – 5

Table 5: RMSEs and MAEs obtained by the strip algorithm with RBFs as nodal functions for the function f_1.

As we already pointed out, the strip algorithm organizes nodes and performs the nearest neighbour procedure in a way particularly suited for the track data interpolation. However, we found that optimal results are obtained by the strip algorithm also when it is applied to scattered data. In particular the execution times of the strip algorithm turned out lower than those of Renka's algorithm, and this can be explained by the smaller computational effort required by the former. RMSEs and execution times are shown in Table 9 for Renka's, strip and IMQ strip algorithms. The plot in Fig. 5 compares results obtained by setting $n_L = 13$ and $n_W = 10$, chosen via trial and error. For the strip algorithm we used τ_1 as localizing function in the weights. Finally, we note that the execution time is only partially (see Table 10) influenced by the number of evaluations at the grid points.

5. Track data interpolation algorithm

Let $\mathcal{S}_n = \{(x_i, y_i), i = 1, 2, \ldots, n\}$ be a set of track data points in a domain $D = [0, 1] \times [0, 1] \subset \mathbb{R}^2$, and $\mathcal{F}_n = \{f_i, i = 1, 2, \ldots, n\}$ a set of associated values of an unknown function $f : D \to \mathbb{R}$. For track data points here we intend

RBF / n	1000	2000	4000	8000	16000
TPS	2.0644E – 2 3.0006E – 1	1.1248E – 2 1.6491E – 1	5.9736E – 3 8.3108E – 2	2.7319E – 3 3.0512E – 2	1.4534E – 3 2.5735E – 2
Gaussian	1.1426E – 3 2.5813E – 2	2.8395E – 4 5.0265E – 3	7.0030E – 5 1.3189E – 3	1.5861E – 5 3.8814E – 4	4.7930E – 6 7.8783E – 5
MQ	1.4829E – 3 3.2994E – 2	2.9807E – 4 4.5790E – 3	7.5994E – 5 1.6226E – 3	1.4639E – 5 1.9987E – 4	8.3343E – 6 2.6484E – 4
IMQ	1.5250E – 3 2.7541E – 2	3.2177E – 4 5.5880E – 3	8.1428E – 5 1.9071E – 3	1.6359E – 5 2.2348E – 4	7.4154E – 6 2.3591E – 4

Table 6: RMSEs and MAEs obtained by the strip algorithm with RBFs as nodal functions for the function f_2.

RBF / n	1000	2000	4000	8000	16000
TPS	5.2636E – 3 6.8544E – 2	2.2818E – 3 3.1592E – 2	1.0894E – 3 1.3144E – 2	6.5611E – 4 1.1422E – 2	2.6804E – 4 3.8364E – 3
Gaussian	4.3050E – 4 6.3983E – 3	1.0003E – 4 1.4928E – 3	2.5752E – 5 2.6237E – 4	6.7225E – 6 8.1370E – 5	1.7877E – 6 3.5539E – 5
M	3.2219E – 4 3.4521E – 3	9.0324E – 5 1.2618E – 3	2.2263E – 5 3.5140E – 4	6.9405E – 6 1.5780E – 4	2.6136E – 6 1.0098E – 4
IMQ	3.2810E – 4 3.5300E – 3	9.1078E – 5 1.0568E – 3	2.3711E – 5 4.9285E – 4	7.9225E – 6 2.2975E – 4	2.2765E – 6 8.6457E – 5

Table 7: RMSEs and MAEs obtained by the strip algorithm with RBFs as nodal functions for the function f_3.

RBF / n	1000	2000	4000	8000	16000
TPS	3.6520E – 3 7.8715E – 2	1.6672E – 3 3.6660E – 2	8.9783E – 4 1.9060E – 2	4.4469E – 4 7.0091E – 3	2.5006E – 4 5.9320E – 3
Gaussian	1.3327E – 3 4.7980E – 2	2.1284E – 4 6.6150E – 3	4.4078E – 5 1.0495E – 3	1.5738E – 5 3.4930E – 4	7.8294E – 6 2.3427E – 4
MQ	1.4504E – 3 5.3337E – 2	1.7017E – 4 4.7333E – 3	3.6401E – 5 8.4462E – 4	1.1848E – 5 2.4779E – 4	5.0412E – 6 1.5073E – 4
IMQ	1.2840E – 3 4.6630E – 2	1.5253E – 4 3.8991E – 3	3.4068E – 5 6.9967E – 4	1.0756E – 5 2.0219E – 4	4.2042E – 6 1.3245E – 4

Table 8: RMSEs and MAEs obtained by the strip algorithm with RBFs as nodal functions for the function f_4.

n	Renka's algorithm		Strip algorithm		Strip algorithm IMQ	
	RMSE	t_{sec}	RMSE	t_{sec}	RMSE	t_{sec}
1000	7.2619E – 4	1.157	8.1573E – 4	0.313	8.8547E – 5	1.000
2000	1.8668E – 4	1.548	2.8286E – 4	0.390	2.7122E – 5	1.172
4000	5.6301E – 5	2.346	7.0714E – 5	0.594	8.7839E – 6	1.438
8000	2.5499E – 5	3.957	3.0269E – 5	1.281	1.9411E – 6	1.985
16000	8.3375E – 6	7.226	1.0297E – 5	2.500	6.6912E – 7	3.781

Table 9: RMSEs and execution times (in seconds) obtained by Renka's algorithm and the strip algorithm using the localizing function τ_1 with $n_L = 13$ and $n_W = 10$ for f_1 (scattered data).

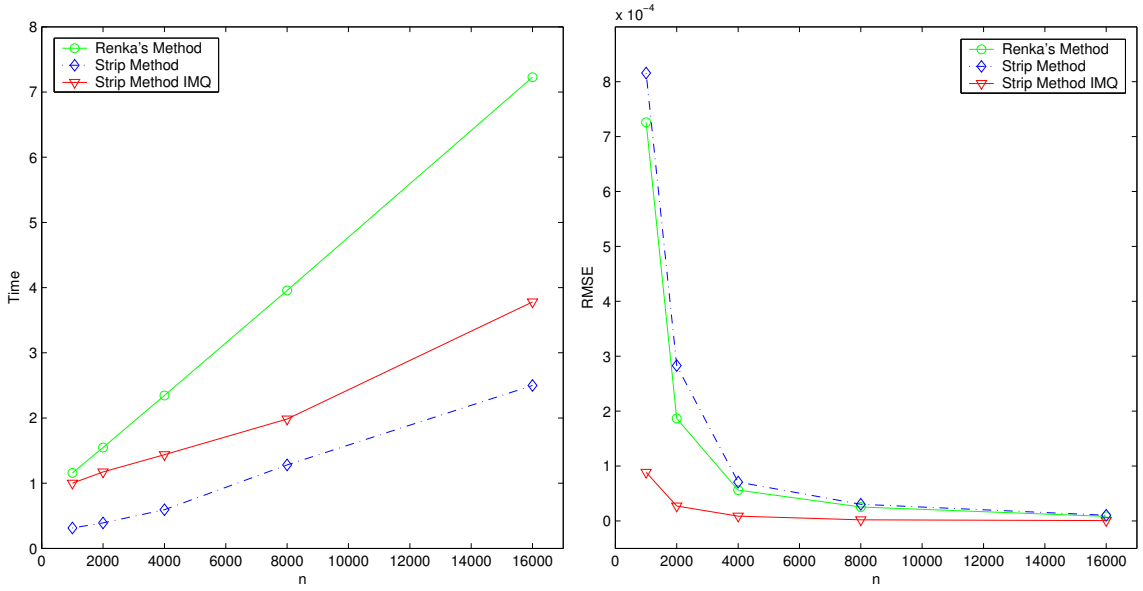

Figure 5: Execution times (left) and RMSEs (right) obtained by Renka's algorithm and the strip algorithm using the localizing function τ_1 with $n_L = 13$ and $n_W = 10$ for f_1 (scattered data).

that nodes may be irregularly spaced and collocated on each line or curve in different positions. Moreover, a feature of this kind of data is that two adjacent nodes along a given line or curve are much closer together than nodes on different lines or curves. A few papers were devoted to the study of approximating schemes for track data (see, e.g., [7, 8, 12, 16]. We suppose that the only available information is represented by the data, that is, no model for the underlying physical phenomenon is considered, as it is done in [25].

In this section we describe the strip searching method and the relative algorithm for track data, focusing only on the parts which differ from those relative to the scattered data interpolation.

5.1. Strip searching method for track data

This process uses a different strategy to construct the strip structure. In the strip searching method for scattered data the strip size derives from the neighbourhood half-size to optimize the searching procedure of the nearby nodes. Conversely, the strip searching method for track data depends on the number of tracks. Therefore, in general, the ratio δ_y^L/δ_s is not equal to one, and accordingly the search of the nearest nodes involves more than three strips.

To find the strips to be examined in the searching procedure of nodes, we consider the following computational rule that consists of three steps:

Grid points	t_{sec} – Renka's algorithm	t_{sec} – Strip algorithm
$11 \times 11 = 121$	6.484	1.516
$21 \times 21 = 441$	6.640	1.656
$31 \times 31 = 961$	6.671	1.875
$41 \times 41 = 1681$	7.091	2.141
$51 \times 51 = 2601$	7.226	2.500

Table 10: Execution times (in seconds) obtained by Renka's algorithm and the strip algorithm for interpolating $n = 16000$ scattered data by varying the number of grid points.

1. Computation of the ratio between the semi-size δ_y^L of square neighbourhood and the strip size δ_s, namely

$$k^* = \frac{\delta_y^L}{\delta_s} = q\delta_y^L = i,$$

   where q indicates the number of strips in the domain D.
2. Taking the smallest integer greater than i, i.e. $i^* = \lceil i \rceil$, $i \in \mathbb{R}^+$, we find from (8) the number j^* of strips to be examined for each node.
3. Referring to the strip s_k, $(k = 1, 2, \ldots, q)$, a strip searching procedure is applied, to examine the nodes from strip $s_k - i^*$ to strip $s_k + i^*$.

As we have previously seen, in general, we need to reduce the total number of strips to be examined for the nodes of the "first"and "last"strips. Thus, if $s_k - i^* < 1$ or $s_k + i^* > q$ it will be convenient to set $s_k - i^* = 1$ and strip $s_k + i^* = q$, respectively.

5.2. Strip algorithm for track data

The strip algorithm for track data differs from that for scattered data only in some details. These allow to optimize the searching procedure of the nearby nodes, and accordingly to minimize the computational cost. Hence, as regard to the algorithm described in Subsection 4.2, the following change is required:

`Stage 3.` After determining the strip size

$$\delta_s = \frac{1}{q}, \tag{11}$$

the strips are numbered from 1 to q.

5.3. Numerical results

In order to test the strip algorithm for track data, we generated some track data point sets of dimensions $n = 1000$, 2000, 4000, 8000, 16000. An example is given in Fig. 6, which shows a set of 1000 track data points.

In the comparison with Renka's algorithm, the strip algorithm for track data make use of τ_3 as localizing function, while the localizing parameters we used are $n_L = 13$ and $n_W = 8$ for both algorithms.

Tables 11, 12, 13, and 14 show the errors obtained by running Renka's algorithm and the strip algorithm on the four test functions presented in Subsection 4.3. Errors are comparable when a least squares approximant as nodal function is used, while the strip algorithm achieves better accuracy if inverse multiquadric is employed.

In Table 15 RMSEs and execution times for the three algorithms are listed, and in Fig. 7 are plotted. For the strip algorithm we used τ_1 as localizing function in the weights. We choose the localizing parameters as $n_L = 13$ and $n_W = 10$. Note that the execution times of the strip algorithm are much lower than those obtained using the Renka's algorithm. The reason is that the data structure employed in the strip algorithm is suitable for a very fast and efficient nearest neighbour search.

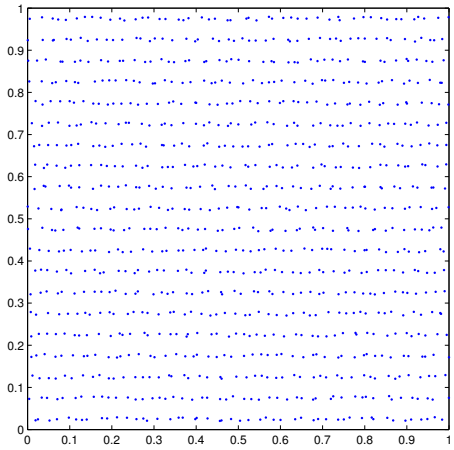

Figure 6: Plot of a track data point set (n = 1000).

n	1000	2000	4000	8000	16000
Renka's algorithm	3.6136E – 4 5.3868E – 3	1.0363E – 4 1.2528E – 3	4.7247E – 5 7.6312E – 4	1.5379E – 5 2.0570E – 4	5.4384E – 6 9.9957E – 5
Strip algorithm	4.5837E – 4 5.8467E – 3	1.3966E – 4 1.3175E – 3	4.8517E – 5 5.2584E – 4	1.5662E – 5 1.8865E – 4	5.9763E – 6 7.1485E – 5
Strip algorithm IMQ	6.0195E – 5 8.5119E – 4	1.9160E – 5 5.4292E – 4	5.3223E – 6 1.3898E – 4	1.3245E – 6 1.8309E – 5	2.6896E – 7 4.7704E – 6

Table 11: RMSEs and MAEs obtained by Renka's algorithm and the strip algorithm either with least squares or inverse multiquadric function as nodal function for the function f_1.

n	1000	2000	4000	8000	16000
Renka's algorithm	4.8687E – 3 4.9131E – 2	1.6902E – 3 2.2466E – 2	9.3309E – 4 1.4329E – 2	2.4749E – 4 3.9565E – 3	8.5918E – 5 1.5992E – 3
Strip algorithm	6.6890E – 3 1.1157E – 1	2.4470E – 3 6.5274E – 2	1.0451E – 3 1.6349E – 2	2.8643E – 4 5.9115E – 3	9.9225E – 5 1.7236E – 3
Strip algorithm IMQ	6.9901E – 4 7.2506E – 3	1.8903E – 4 2.9117E – 3	4.9248E – 5 1.1772E – 3	1.5553E – 5 4.8850E – 4	2.3111E – 6 2.8893E – 5

Table 12: RMSEs and MAEs obtained by Renka's algorithm and the strip algorithm either with least squares or inverse multiquadric function as nodal function for the function f_2.

n	1000	2000	4000	8000	16000
Renka's algorithm	2.4595E – 3 2.4941E – 2	6.3990E – 4 8.9470E – 3	5.2097E – 4 1.1832E – 2	9.3484E – 5 1.2963E – 3	3.8863E – 5 5.8299E – 4
Strip algorithm	2.4415E – 3 3.1233E – 2	6.0235E – 4 7.1965E – 3	4.3469E – 4 8.9015E – 3	6.6895E – 5 7.3296E – 4	3.8891E – 5 7.4650E – 4
Strip algorithm IMQ	2.0806E – 4 2.2564E – 3	4.5709E – 5 4.7039E – 4	1.2407E – 5 8.4037E – 5	3.8863E – 6 3.7584E – 5	7.3300E – 7 8.4982E – 6

Table 13: RMSEs and MAEs obtained by Renka's algorithm and the strip algorithm either with least squares or inverse multiquadric function as nodal function for the function f_3.

n	1000	2000	4000	8000	16000
Renka's algorithm	1.8620E – 3 4.4156E – 2	8.5732E – 4 2.5727E – 2	2.8142E – 4 6.7194E – 3	9.1837E – 5 2.2599E – 3	2.8901E – 5 7.6237E – 4
Strip algorithm	3.4486E – 3 9.3219E – 2	6.5429E – 4 2.0524E – 2	2.7522E – 4 5.6436E – 3	1.0017E – 4 3.4568E – 3	3.5563E – 5 8.5366E – 4
Strip algorithm IMQ	3.0229E – 4 5.2565E – 3	6.8442E – 5 1.4817E – 3	1.6652E – 5 2.7202E – 3	1.3570E – 5 5.7315E – 4	1.6744E – 6 7.1047E – 5

Table 14: RMSEs and MAEs obtained by Renka's algorithm and the strip algorithm either with least squares or inverse multiquadric function as nodal function for the function f_4.

	Renka's algorithm		Strip algorithm		Strip algorithm IMQ	
n	RMSE	t_{sec}	RMSE	t_{sec}	RMSE	t_{sec}
1000	3.4343E – 4	1.235	4.4996E – 4	0.219	6.0801E – 5	0.484
2000	1.0231E – 4	1.751	1.4960E – 4	0.281	1.9333E – 5	0.594
4000	4.5699E – 5	2.706	4.6879E – 5	0.422	5.5826E – 6	0.797
8000	1.4988E – 5	4.864	1.6810E – 5	0.719	1.3595E – 6	1.266
16000	5.2563E – 6	8.759	5.7719E – 6	1.313	2.7681E – 7	2.157

Table 15: RMSEs and execution times (in seconds) obtained by Renka's algorithm and the strip algorithm using the localizing function τ_1 with $n_L = 13$ and $n_W = 10$ for f_1 (track data).

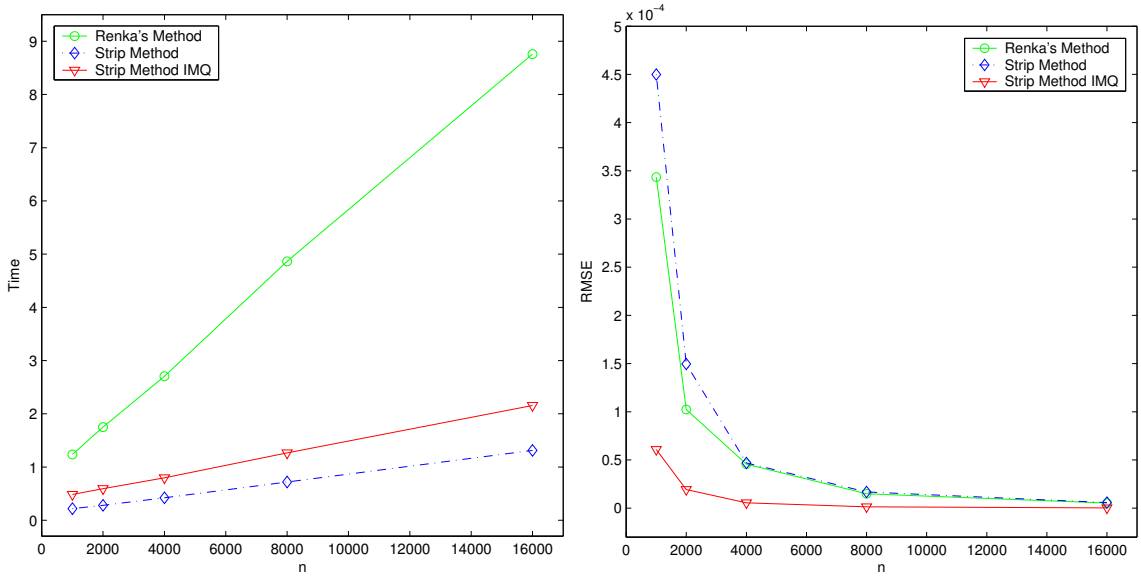

Figure 7: Execution times (left) and RMSEs (right) obtained by Renka's algorithm and the strip algorithm using the localizing function τ_1 with $n_L = 13$ and $n_W = 10$ for f_1 (track data).

6. Parallel algorithm

Shepard's type interpolants enjoy many useful properties, which can be successfully applied, such as a subdivision technique, multistage and iterative procedures (see, e.g., [1, 2, 3]). They are particularly suitable for parallel computation, achieving the maximum speed-up (see [10, 5]). Moreover, they have a wide field of applications. We recall here the most interesting property for our purposes:

Property 6.1 (Subdivision Technique). *Let us make a partition of the set $\mathcal{S}_n$ of nodes on the domain $D \subset \mathbb{R}^2$ into q subsets $\mathcal{S}_{n_j}$, so that the j-th subset, $(j = 1, 2, \ldots, q)$, consists of the nodes $x_{j1}, x_{j2}, \ldots, x_{jn_j}$, with $n_1 + n_2 + \cdots + n_q = n$, and the values $f_{jk_j}, (j = 1, 2, \ldots, q; k_j = 1, 2, \ldots, n_j)$, correspond to the nodes x_{jk_j}. The indexing of the nodes in the subsets may not depend on the indexing in the set, provided the one-to-one correspondence is saved. Then, being $\mathcal{S}_n = \mathcal{S}_{n_1} \cup \mathcal{S}_{n_2} \cdots \cup \mathcal{S}_{n_q}$ and $\mathcal{S}_{n_p} \cap \mathcal{S}_{n_r} = \emptyset$ for $p \neq r$, the Shepard's interpolant can be rewritten in the form*

$$F(x; f, \mathcal{S}_n) = \sum_{j=1}^{q} F(x; f, \mathcal{S}_{n_j}) \, \frac{A_j}{\sum_{j=1}^{q} A_j}, \tag{12}$$

where

$$A_j = \sum_{k_j=1}^{n_j} \frac{1}{\alpha(x, x_{jk_j})}.$$

The strip algorithm discussed above, both for scattered and track data, is very suitable for parallel computations, on account of the data structures. In this section we sketch the related parallel algorithm, which will be completely explained and analyzed in a forthcoming paper.

We suppose to have $q = kp$ strips, where p is the number of available processors. Let the first k strips be assigned to the first processor, the second k strips to the second processor, and so on up to the last processor. There is shown in [5] that under condition of well-balanced workload the speed-up factor of the parallel computation is approximately equal to the number of processors. The proposed parallel algorithm consists of three stages:

`Stage 1. Data distribution.` When the node set $\mathcal{S}_n$ has been partitioned into q strips and the nodes in each strip have been ordered by the master processor, the data organized in k strips are sent to slave processors for the

localization phase (see `Stage 5` of sequential algorithm). This data distribution allows the load balancing and the reduction of communication among processors. Each slave processor has to storage approximately $k \cdot n/p$ nodes.

`Stage 2. Local interpolation problem resolution.` On each slave processor h, $(h = 1, 2, \ldots, p)$, after finding the n_L nodes nearest to x_k, a local interpolant is constructed for x_k, $(k = 1, 2, \ldots, n_h)$, where n_h is the number of nodes assigned to the h-th processor.

`Stage 3. Evaluation phase.` In this stage each slave processor computes the partial sum

$$\sum_{k=1}^{n_h} L_k(x)\bar{W}_k(x), \qquad h = 1, 2, \ldots, p,$$

where x is here an evaluation point. Then the master processor collects the partial results, and using the Property 6.1 accumulates them.

7. Final remarks

In this paper we presented a new efficient interpolation algorithm, which works well also when the number of scattered or track data is very large, and achieves a good accuracy. An optimized implementation of the modified version of Shepard's method is mainly obtained by means of a strip nearest neighbour searching procedure. Moreover, the algorithm is flexible (indeed, different choices for weights and local approximants are allowable), and completely automatic since it works succesfully even when the distribution of nodes is not uniform. Numerical tests have shown that the strip algorithm is comparable with Renka's algorithm with regard to accuracy, and is generally better for execution time. The strip algorithm is easily and efficiently parallelizable, but this topic deserves careful investigation. Some other extensions are possible and, at the moment, under investigation: first, the modelling of discontinuous surfaces only known on scattered or track data; then, the extension to the spherical setting, for which a modified Shepard's method using zonal basis functions was proposed [17]. Moreover, we remark that a well-known drawback of local interpolation methods is the need of a manual determination of the local parameters n_L and n_W; hence an automatic choice would be desirable.

Finally, as research and work in progress we will expect to refine the algorithm, presented here in a first version, by adopting suitable data structures like the kd-trees. However, we note that the current version allows to dynamically handle data sets, increasing or decreasing their sizes according to advisability through the Property 6.1 presented in Section 6 (see [4] for further details).

A. Appendix: Renka's algorithm

Among local methods the modified version of Shepard's method given in [21, 27] shows the characteristics of accuracy and efficiency. Renka in [28, 29] gave an optimized implementation of the method in Fortran language, which is still now one of the most powerful tools of the available mathematical software for multivariate interpolation of large scattered data sets. Here we briefly recall this implementation in the 2D case, since the scope is to compare it with the strip algorithm.

The Renka's algorithm can be divided in two parts: a preprocessing phase and an evaluation phase.

(a) In the preprocessing phase the $5n$ coefficients of the nodal functions and the n squared radii R_W associated with the weights $\bar{W}_k$, are computed and stored. In order to determine the ordered sequence of nearest neighbours of each node, a cell-based search method is applied, which requires to store nodal indexes in a data structure (see *Algorithm 3*). It starts partitioning the smallest rectangle $[XMIN, XMAX] \times [YMIN, YMAX]$ containing the nodes into an $NR \times NR$ uniform grid of cells, while the indexes of the nodes contained in each cells are stored in arrays. The cell number is NR^2 and the average number of nodes per cell is $C = n/NR^2$. A recommended choice for C is $C = 3$, therefore the number of cells is at most $n/3$. The cells dimensions are $DX = (XMAX - XMIN)/NR$ and $DY = (YMAX - YMIN)/NR$. The data structure consists of two arrays containing nodal indexes: $LCELL$ is an $NR \times NR$ array, where $LCELL(I, J)$ contains the index of the node with smallest index in cell (I, J), or 0 if the cell is empty; $LNEXT$ is an array of length n with $LNEXT(K) = L$, where L is the next node in the cell containing K, or $L = K$ if K is the node with largest index in the cell.

Algorithm 3.

STEP 1 For $I = 1, 2, \ldots, NR$ do
STEP 2 For $J = 1, 2, \ldots, NR$ do
$LCELL(I, J) = 0.$
STEP 3 For $K = n, n-1, \ldots, 1$ do
STEP 4 Set $I = \min\{NR, \lfloor (X(K) - XMIN)/DX \rfloor + 1\}$;
$J = \min\{NR, \lfloor (Y(K) - YMIN)/DY \rfloor + 1\}$;
$L = LCELL(I, J)$.
STEP 5 If $L = 0$
then set $LNEXT(K) = K$;
else set $LNEXT(K) = L$.
STEP 6 $LCELL(I, J) = K$.

A nearest neighbour searching is performed so many times as to determine the sequence of n_L nearest neighbours to some node K, excluding from the search the nodes already in the sequence. A node L to be excluded is marked by setting $LNEXT(L)$ to $-LNEXT(L)$. The search begins in the cell containing K and proceeds outward in rectangular layers until all cells that contain nodes within distance R from K have been searched, where R is the distance from K to the first unmarked node encountered (node L such that $LNEXT(L) > 0$). Thus, once R has been determined, the region to be searched is characterized by $(I, J) \in [IMIN, IMAX] \times [JMIN, JMAX]$, where $IMIN = \max\{1, \lfloor (X(K) - R - XMIN)/DX \rfloor + 1\}$ and $IMAX = \min\{NR, \lfloor (X(K) + R\ -XMIN)/DX \rfloor + 1\}$, with similar expressions for $JMIN$ and $JMAX$. When the n_L nearest neighbours of each node are determined, a least squares system is solved for the coefficients of the nodal functions, which are then stored.

(b) The evaluation routines require a loop through the set of nodes whose radii R_W include a node (XP, YP). The cells that must be searched are those intersected by a disk centred at (XP, YP) and having radius R, where R is the largest of the R_W values, and is computed in the preprocessing phase. The region to be searched is defined as above with (XP, YP) in place of $(X(K), Y(K))$ in the expressions for $IMIN$, $IMAX$ and $JMIN$, $JMAX$, respectively.

The two previously described procedures require inner loops on nodes in a cell. *Algorithm 4* describes these loops.

Algorithm 4.

STEP 1 Set $LN = LCELL(I, J)$.
STEP 2 If $LN = 0$
then break.
STEP 3 While $(LN < L)$
set $L = LN$;
$LN = LNEXT(L)$.

Acknowledgements

The authors are very grateful to Prof. Pier Carlo Giolito (Department of Computer Science, University of Turin) for his helpful suggestions and comments.

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, 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, Simultaneous interpolation and approximation by a class of multivariate positive operators, Numer. Algorithms 34 (2003) 147–158.
[4] G. Allasia, Recursive and parallel algorithms for approximating surface data on a family of lines or curves, in: P. Ciarlini, M.G. Cox, F. Pavese, G.B. Rossi (Eds.), Advanced Mathematical and Computational Tools in Metrology VI, World Scientific, NJ, 2004, pp. 137–148.
[5] G. Allasia, P. Giolito, Fast evaluation of cardinal radial basis interpolants, in: A. Le Mèhautè, C. Rabut, L.L. Schumaker (Eds.), Surface Fitting and Multiresolution Methods, Vanderbilt Univ. Press, Nashville, TN, 1997, pp. 1–8.
[6] G. Allasia, R. Besenghi, M. Costanzo, Approximation to surface data on parallel lines or curves by a near-interpolation operator, Int. J. Comput. Numer. Anal. Appl. 5 (2004) 317–337.
[7] I.J. Anderson, M.G. Cox, J.C. Mason, Tensor-product spline interpolation to data on or near a family of lines, Numer. Algorithms 5 (1993) 193–204.
[8] D. Apprato, C. Gout, D. Komatitsch, A new method for C^k-surface approximation from a set of curves, with application to ship track data in the Marianas trench, Math. Geol. 34 (2002) 831–843.
[9] R.E. Barnhill, R.P. Dube, F.F. Little, Properties of Shepard's surfaces, Rocky Mountain J. Math. 13 (1983) 365–382.
[10] 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.
[11] M.D. Buhmann, Radial Basis Functions: Theory and Implementation, Cambridge Monogr. Appl. Comput. Math., vol. 12, Cambridge Univ. Press, Cambridge, 2003.
[12] R.E. Carlson, T.A. Foley, Interpolation of track data with radial basis methods, Comput. Math. Appl. 12 (1992) 27–34.
[13] R. Cavoretto, Un metodo per l'approssimazione di curve e superfici note su famiglie di rette o curve parallele, Tesi di Laurea Specialistica, Università degli Studi di Torino, 2006.
[14] E.W. Cheney, Multivariate approximation theory: selected topics, CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 51, SIAM, Philadelphia, 1986.
[15] E.W. Cheney, W. Light, A Course in Approximation Theory, Brooks Cole, Pacific Grove, CA, 1999.
[16] A. Crampton, J.C. Mason, Surface approximation of curved data using separable radial basis functions, in: P. Ciarlini, M.G. Cox, E. Filipe, F. Pavese, D. Richter (Eds.), Advanced Mathematical and Computational Tools in Metrology V, World Scientific, Singapore, 2000, pp. 119–126.
[17] A. De Rossi, Spherical interpolation of large scattered data sets using zonal basis functions, in: M. Dæhlen, K. Mørken, L.L. Schumaker (Eds.), Mathematical Methods for Curves and Surfaces, Nashboro Press, Brentwood, TN, 2005, pp. 125–134.
[18] G.E. Fasshauer, Meshfree Approximation Methods with MATLAB, World Scientific Publishers, Singapore, 2007.
[19] R. Franke, A critical comparison of some methods for interpolation of scattered data, Report TR-NPS-53-79-003 (1979) Naval Postgraduate School, Monterey, California.
[20] R. Franke, Scattered data interpolation: tests of some methods, Math. Comp. 38 (1982) 181–200.
[21] R. Franke, G. Nielson, Smooth interpolation of large sets of scattered data, Internat. J. Numer. Methods Engrg. 15 (1980) 1691–1704.
[22] R. Franke, H. Hagen, Least squares surface approximation using multiquadrics and parametric domain distorsion, Comput. Aided Geom. Design 16 (1999) 177–196.
[23] W.J. Gordon, J.A. Wixom, Shepard's method of metric interpolation to bivariate and multivariate data, Math. Comp. 32 (1978) 253–264.
[24] A. Iske, Radial basis functions: basics, advanced topics and meshfree methods for transport problems, Rend. Sem. Mat. Univ. Pol. Torino 61 (2003) 247–285.
[25] O. Kounchev, Multivariate Polysplines: Applications to Numerical and Wavelet Analysis, Academic Press, San Diego, CA, 2001.
[26] D. Lazzaro, L.B. Montefusco, Radial basis functions for the multivariate interpolation of large scattered data sets, J. Comput. Appl. Math. 140 (2002) 521–536.
[27] R.J. Renka, Multivariate interpolation of large sets of scattered data, ACM Trans. Mathematical Software 14 (1988) 139–148.
[28] R.J. Renka, Algorithm 660: QSHEP2D: Quadratic Shepard method for bivariate interpolation of scattered data, ACM Trans. Math. Software 14 (1988) 149–150.
[29] R.J. Renka, Algorithm 661: QSHEP3D: Quadratic Shepard method for trivariate interpolation of scattered data, ACM Trans. Math. Software 14 (1988) 151–152.
[30] R.J. Renka, R. Brown, Algorithm 792: accuracy tests of ACM algorithms for interpolation of scattered data in the plane, ACM Trans. Math. Software 25 (1999) 78–94.
[31] R. Schaback, Creating surfaces from scattered data using radial basis functions, in: M. Dæhlen, T. Lyche, L.L. Schumaker (Eds.), Mathematical Methods for Curves and Surfaces, Vanderbilt Univ. Press, Nashville, TN, 1995, pp. 477–496.
[32] D. Shepard, A two-dimensional interpolation function for irregularly spaced data, Proc. 23rd ACM Nat. Conf., Brandon/Systems Press Inc., Princeton, 1968, 517–524.
[33] H. Wendland, Scattered Data Approximation, Cambridge Monogr. Appl. Comput. Math., vol. 17, Cambridge Univ. Press, Cambridge, 2005.