Dolomites Research Notes on Approximation (DRNA)
Vol. 3, (2010), p. 39 – 78

ISSN: 2035-6803

Efficient approximation algorithms. Part II: Scattered data interpolation based on strip searching procedures

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 new algorithm for bivariate interpolation of large sets of scattered and track data is presented. Then, the extension to the sphere is analyzed. The method, whose different versions depend partially 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 local 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 method is extended to the sphere using a modified spherical Shepard's interpolant with the employment of zonal basis functions as local approximants. The proposed algorithms are very fast, owing to the optimal nearest neighbour searching, and achieve a good accuracy. The efficiency and reliability of the algorithms are shown by several numerical tests, performed also by Renka's algorithms for a comparison.

Keywords: surface modelling, Shepard's type formulas, local methods, scattered and track data interpolation, radial basis functions, zonal basis functions, interpolation algorithms.

AMS Subject classification[2010]: 65D05, 65D15, 65D17.

1 Introduction

We consider the problem of interpolating a continuous function $f : \mathbb{R}^2 \to \mathbb{R}$, defined on a bounded domain $D \subset \mathbb{R}^2$ and known only on a finite set $\mathcal{S}_n = \{x_i, i = 1, 2, \ldots, n\} \subset D$ of data points or nodes. It is required to find a real bivariate function F 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. In 1988 Renka [50] proposed an optimized implementation of a modified version of Shepard's method, which is still now one of the most powerful tools. Then, in 2002, Lazzaro and Montefusco [39] presented a modification of Renka's algorithm, in which local approximants (or nodal functions) based on the least squares method are replaced by others based on radial basis functions (RBFs), thus improving accuracy.

In the papers [4, 5] we presented 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 may be 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. A frequent feature of this kind of data, often called *track data*, is that two adjacent nodes along a given track are much closer together than nodes on different tracks. The considered schemes approximate the data by means of interpolation or near-interpolation operators, both based on cardinal radial basis functions, whose properties are widely discussed in [2, 3]. In particular, the scheme in [4] has been widely tested in [15], where some interesting devices are presented.

As a matter of fact, in several applied problems the function values are known along a number of parallel 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, are taken near to rather than exactly on straight or curved tracks, owing to the effects of disturbing agents, such as wind and waves.

Several methods (see, e.g., [25, 14, 7, 20, 8] and references therein) have been proposed to solve the considered problem by using different interpolation techniques and tools (tensor-product splines, least squares, radial basis functions, Chebyshev polynomials of the first or second kind, etc.). A number of papers on the subject is reviewed in [5].

Now, if we suppose to move the parallel lines or curves on the domain $D \subset \mathbb{R}^2$ 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 to general sets of scattered data. The outcome is an efficient interpolation algorithm, called *strip algorithm*, for modelling continuous surfaces [6]. 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 finds 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 local interpolant F which is based on radial basis functions or least squares approximants as nodal functions. Numerical results, compared with those of Renka's algorithm, show that the strip algorithm allows us to improve efficiency of the algorithm implementation of the modified Shepard's method, in particular with regard to the execution CPU times [17].

Since numerical results point out a good performance of the bivariate interpolation algorithm, we extend it to the spherical setting. Given the unit sphere $\mathbb{S}^2 = \{x \in \mathbb{R}^3 : \|x\|_2 = 1\}$ in $\mathbb{R}^3$, we consider the problem of interpolating a function $f : \mathbb{S}^2 \to \mathbb{R}$, defined on a finite set $\mathcal{S}_n = \{x_i, i = 1, 2, \ldots, n\} \subset \Omega$ of distinct data points or nodes. We want to construct a (smooth) multivariate function $F : \mathbb{S}^2 \to \mathbb{R}$, which interpolates the data values or function values f_i at the nodes x_i, namely $F(x_i) = f_i$, $i = 1, 2, \ldots, n$.

This data fitting problem where the underlying domain is on a sphere arises in many areas, including e.g. geophysics and meteorology. In these cases, in general, the sphere $\mathbb{S}^2$ is taken as a model of the Earth.

Several methods have been proposed to solve the spherical interpolation problem for scattered data (for an overview, see [23]), but, as far as we know, with the exception of the macro-element methods based on spherical splines discussed in [1], most methods are inefficient when dealing with large scattered data sets.

To solve the interpolation problem on the sphere, instead of the Euclidean metric we consider the geodetic metric $g : \mathbb{S}^2 \times \mathbb{S}^2 \to [0, \pi]$, which is defined by $g(x, y) = \arccos(x^T y)$, for any

$x, y \in \mathbb{S}^2$, and we replace the radial basis function $\phi : [0, \infty) \to \mathbb{R}$ with a *zonal basis function* (ZBF) $\psi : [0, \pi] \to \mathbb{R}$. Thus we seek an interpolant F from the linear space spanned by the n functions $\psi(g(\cdot, x_j))$, $j = 1, 2, \ldots, n$. The uniqueness of such a solution clearly depends on the choice of the zonal basis function ψ. Studies on (conditional) positive definiteness of ψ started in the 1940s, when Schoenberg [56] characterized the class of positive definite functions on the sphere (see also [58, 38]). In the early 1990s, Schoenberg's work has been extended: the papers by Xu and Cheney [67] and by Menegatto [43] are both addressed to the problem of characterizing strictly (conditionally) positive definite functions on $\mathbb{S}^{m-1}$, for $m \geq 2$. Furthermore, in 1995, Cheney [19] showed how these functions can be used to provide a unique solution to spherical interpolation problem. The final result, which consists in specializing the radial basis function method to the sphere, is commonly called the zonal basis function method. A large number of papers has been devoted in the last years to investigate both theoretical and computational aspects of the zonal basis function method and its modifications, see for example [44, 63, 46, 36, 41, 48, 47, 12, 40, 42, 60, 61] and [59, 45, 34, 32] for applications.

The extension to the sphere of the strip algorithm on $\mathbb{R}^2$ is based on the partition of the unit sphere in a convenient number of spherical zones, the construction of localizing neighbourhoods, that is spherical caps, and then, specifically, a spherical zone searching procedure. The interpolation formula we propose is a further variant of the modified Shepard's method for the sphere, which uses zonal basis function interpolants as nodal functions, already proposed in [21, 16, 18] (see also [9, 10]). Hence, this local interpolation approach enables to exploit the accuracy of ZBFs, and, at the same time, to overcome common disadvantages, such as the unstability due to the need of solving large linear systems (possibly, ill-conditioned) and the inefficiency of the global ZBF method. Moreover, the proposed algorithm, called *spherical zone algorithm*, is very fast, owing to the optimal nearest neighbour searching, achieves a good accuracy, and guarantees a high parallelism.

The paper is organized as follows. Section 2 is devoted to briefly remind 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 3 we describe the strip algorithm, dwelling on the details that allow the procedure to be accurate and computationally efficient. Section 4 is devoted to presentation of the local interpolation scheme on the sphere, that is, the modified spherical Shepard's method with zonal basis functions as local approximants. In Section 5 we describe the spherical zone algorithm, focusing only on the parts which differ from the strip algorithm. Some computational aspects of the considered interpolation algorithms, such as computational complexity and storage requirements, are presented in Section 6. In Section 7 numerical results show the goodness of the presented interpolation methods and the effectiveness of the related algorithms. In particular, numerical comparisons with Renka's algorithms are presented in both cases.

2 Modified Shepard's method

The classical Shepard's formula has two crucial drawbacks, namely the occurrence of flat spots at the nodes (i.e., the first partial derivatives vanish there) and the dependence of the operator on all the nodes (see, e.g., [2]). To avoid these shortcomings, a modified version of Shepard's method has been developed by Franke and Nielson [27], and then improved by Renka [50]. An interesting modification has been suggested by Lazzaro and Montefusco [39].

We consider the following definition of the modified Shepard's method.

Definition 2.1. *Given a set $\mathcal{S}_n = \{x_i, i = 1, 2, \dots, n\}$ of distinct nodes, arbitrarily distributed on a bounded domain $D \subset \mathbb{R}^m$, with the corresponding set $\mathcal{F}_n = \{f_i, i = 1, 2, \dots, n\}$ of associated values of an unknown continuous function $f : D \to \mathbb{R}$, the modified Shepard's method $F : D \to \mathbb{R}$ takes the form*

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

The nodal functions L_j, $j = 1, 2, \dots, n$, 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 $\bar{W}_j$, $j = 1, 2, \dots, n$, are given by

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

where

$$W_j(x) = \tau(x, x_j)/\alpha(x, x_j), \tag{3}$$

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

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

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

where L_j is a quadratic m-variate polynomial with coefficients $a_j = [a_{j1}, a_{j2}, \dots, a_{jh}]^T$, $h = \binom{m+2}{2}$ is less than the number n_L of nodes of the considered neighbourhood of x_j, and $W_j(x_i) = \tau(x_i, x_j)/\alpha(x_i, x_j)$.

The RBF interpolation method is the most used when we have to interpolate scattered data (see [13, 64]). A RBF interpolant has the form

$$L_j(x) = \sum_{i=1}^{n_L} a_i \varphi(\|x - x_i\|_2) + \sum_{k=1}^{U} b_k \pi_k(x), \tag{4}$$

where the radial basis functions $\varphi(\|\cdot - x_i\|_2)$ depend on the n_L nodes of the considered neighbourhood of x_j, and the $(v-1)$-degree polynomials $\pi_k(x)$ belong to the space $\mathcal{P}_{v-1}^m$ of 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

$$L_j(x_i) = f_i, \qquad i = 1, 2, \dots, n_L,$$

and the side conditions

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

Hence, to compute the coefficients $a = [a_1, a_2, \dots, a_{n_L}]^T$ and $b = [b_1, b_2, \dots, b_U]^T$ in (4), 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, and f denotes the column vector of the function values f_j corresponding to the x_j.

The most popular choices for φ are

$$\begin{array}{lll} \varphi(r) = r^{2v-m} \log r, \quad 2v - m \in 2\mathbb{N}, & & \text{(generalized thin plate spline)} \\ \varphi(r) = \mathrm{e}^{-\alpha^2 r^2}, & & \text{(Gaussian)} \\ \varphi(r) = (c^2 + r^2)^{v/2}, & & \text{(generalized Hardy's multiquadric)} \end{array}$$

where α and c are positive constants, v is an integer (Hardy takes $v = \pm 1$), and $r = || \cdot - x_i||_2$. The Gaussian and the inverse multiquadric (IMQ), which occurs for $v < 0$ in the generalized multiquadric function, are positive definite functions, whereas 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. The addition of the polynomial term in (4) in order to guarantee a unique solution of the considered system is necessary only for the conditionally positive definite functions.

Since the classical Shepard's interpolant 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 (1) we can use various localizing functions $\tau(\cdot, x_j)$.

A first, simple but efficient, localizing function with compact support is

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

where $\mathcal{C}(x_j; \rho)$ is a hypercube of centre at x_j and side ρ.

Another interesting localizing function is given by

$$\tau_2(t) = \begin{cases} -2^{(3\epsilon)} t^3 + 3 \cdot 2^{(2\epsilon)} t^2 - 3 \cdot 2^{\epsilon} t + 1, & \text{if } 0 \le t \le 1/2^{\epsilon}, \\ 0, & \text{if } t > 1/2^{\epsilon}, \end{cases}$$

where $\epsilon \in \mathbb{R}^+$ and $t = \| \cdot - x_j\|_2^2$. In fact, we have $\tau_2(0) = 1$ and $\tau_2(1/2^{\epsilon}) = 0$; the function is convex and its tangent plane at $t = 1/2^{\epsilon}$ is horizontal; the localizing effect increases with ϵ. Localizing functions like τ_2, possibly with different orders of continuity, may represent an alternative choice to the families of localizing functions based on truncated power functions (see [55]).

3 Strip algorithm

In this section, we consider the problem of approximating a continuous 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 nodes, which may be quite scattered or situated on tracks. The function values corresponding to the nodes are collected in the set $\mathcal{F}_n = \{f_i, i = 1, 2, \ldots, n\}$. The method and the relative algorithm could be extended in a straightforward way to more general domains D. Our aim is to describe an 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 [50, 51], which is currently considered as a standard procedure.

Briefly, the process we propose can be described as follows:

1. Partition the domain into a finite number of parallel strips.
2. Consider a *strip searching procedure* that finds the minimal number of strips to be examined, in order to localize a convenient set of neighbour nodes for each strip point, i.e. each node lying on a strip.
3. Approximate the unknown function f by an interpolant F which uses radial basis functions or least squares approximants as nodal functions.

These three steps correspond to data partition, localization and evaluation phases, respectively.

3.1 Strip algorithm for scattered data

We begin describing the strip algorithm for scattered data interpolation; then, in Subsection 3.2, we will consider the strip algorithm for data located on tracks.

The strip algorithm for scattered data 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 evaluation (grid) points; $\mathcal{G}_s = \{(x_{Gi}, y_{Gi}), i = 1, 2, \ldots, s\}$, set of evaluation (grid) points; n_L and n_W, localization parameters.

OUTPUT: $\mathcal{A}_s = \{F_i \equiv F(x_{Gi}, y_{Gi}), i = 1, 2, \ldots, s\}$, 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 shall be constructed (see `Stage 5` below), whose half-size depends on the sample dimension n, the considered value n_L, and the positive integer k_1, i.e.

$$\delta_x^L = \delta_y^L = \sqrt{k_1 \frac{n_L}{n}}, \qquad k_1 = 1, 2, \ldots \tag{5}$$

As an example, in Figure 1 three local square neighbourhoods are shown.

`Stage 3.` The number q of strips to be considered is found taking

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

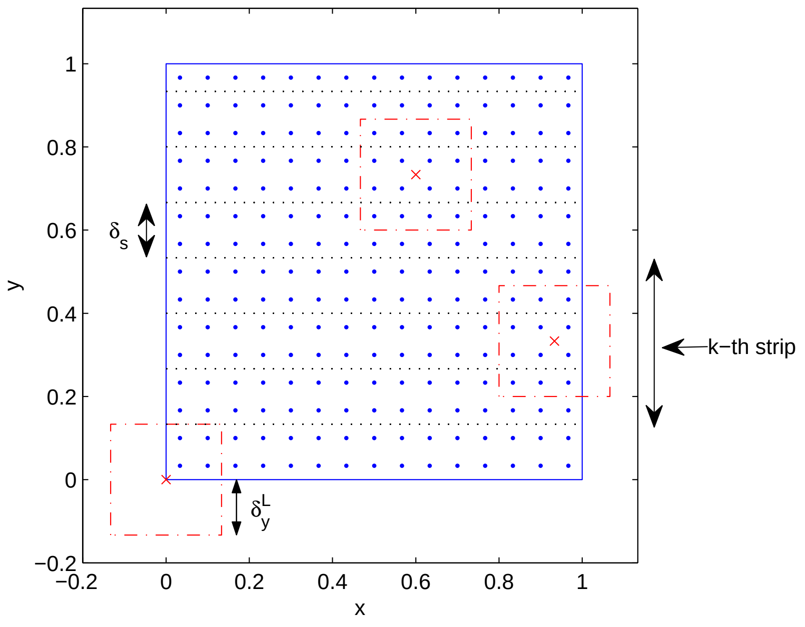


Figure 1: Example of square neighbourhoods with $k_1 = 1$, $n_L = 4$, $n = 225$ and partition of the domain in strips.

where $\lceil \cdot \rceil$ is the greatest integer less or equal to the argument, and then the strips are numbered from 1 to q.

`Stage 4.` On the domain D a family of q strips of equal width δ_s (with the possible exception of one of them) and parallel to the x-axis is constructed, so that the set $\mathcal{S}_n$ of nodes is partitioned by the strip structure into q subsets $\mathcal{S}_{n_k}, k = 1, 2, \dots, q$, whose elements are $(x_{k1}, y_{k1}), (x_{k2}, y_{k2}), \dots, (x_{kn_k}, y_{kn_k})$, $k = 1, 2, \dots, q$ (see Figure 1). Considering scattered data the experience suggests to take $\delta_s \equiv \delta_y^L$. Then, the nodes of $\mathcal{S}_{n_k}$ belonging to the k-th strip 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*).

Algorithm 1: sorting procedure.

```
STEP 1   Set $count = 0$.
STEP 2   For $k = 1, 2, \dots, q$ 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$(n_k, x, y, f)$.
```

`Stage 5.` To identify the strips to be considered in order to construct a suitable neighbourhood for each node, we adopt the following rule which is composed of two steps:

1. We introduce the parameter

$$i^* = \left\lceil \frac{\delta_y^L}{\delta_s} \right\rceil,$$

which in the case of scattered data equals one.

2. For each strip k, $k = 1, 2, \dots, q$, a strip searching procedure is considered, examining the nodes from the $(k - i^*)$-th strip to the $(k + i^*)$-th strip. For scattered data the search of the nearby nodes is limited to (only) three strips: the strip on which the considered node lies, the previous and the next strips.

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

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

Algorithm 2: strip searching procedure.

```
STEP 1    For $k = 1, 2, \dots, q$ 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 $h = begin\_strip_k, \dots, end\_strip_k$ do
      STEP 5    Set $m_h = 0$.
      STEP 6    For $i = begin, \dots, end$ do
         STEP 7    For $j = begin\_strip_i, \dots, end\_strip_i$ 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 6.` All the nodes belonging to the square neighbourhood centred at (x_i, y_i), $i = 1, 2, \dots, n$, are ordered by applying a distance-based sorting process, that is a *quicksort$_d$ procedure*.

`Stage 7.` Taking only the n_L nodes closest to the centre (x_i, y_i), $i = 1, 2, \dots, n$, of the neighbourhood, a local interpolant L_i, $i = 1, 2, \dots, n$, is constructed.

`Stage 8.` For each grid point $(x_{Gj}, y_{Gj}) \in \mathcal{G}_s$, $j = 1, 2, \dots, s$, 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,

$$\delta_x^W = \delta_y^W = \sqrt{k_2 \frac{n_W}{n}}, \qquad k_2 = 1, 2, \dots \tag{6}$$

`Stage 9.` A searching procedure is applied to determine all nodes belonging to a (local) neighbourhood of centre (x_{Gj}, y_{Gj}) and half-side δ_x^W.

`Stage 10.` The nodes of each neighbourhood are first ordered by applying a distance-based sorting procedure (*quicksort$_d$*).

`Stage 11.` Considering only the n_W nodes closest to the grid point (x_{Gj}, y_{Gj}), $j = 1, 2, \dots, s$, it is found a local weight function $\bar{W}_i$, $i = 1, 2, \dots, n$.

`Stage 12.` Applying the modified Shepard's formula (1), the surface can be approximated at any grid point $(x_{Gj}, y_{Gj}) \in \mathcal{G}_s$.

Remark. Supposing a uniform distribution of nodes on the domain D, the size of local square neighbourhoods is found so that each neighbourhood contains 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 (5) (or (6) in `Stage 8`) estimates for $k_1 = 1$ (or $k_2 = 1$), $4n_L$ (or $4n_W$) at least nodes for each inner neighbourhood of D. 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 Figure 1). However, the approach we propose is completely automatic, since the procedure identifies 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 method works successfully even if the distribution of nodes is not uniform.

3.2 Strip algorithm for track data

Now we consider a set of nodes which may be irregularly spaced and collocated on each line or curve in different positions. Moreover, a feature of this kind of data, called track 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 works were devoted to the study of approximating schemes for track data (see, e.g., [14, 7, 20, 37, 8]).

The strip algorithm for track data interpolation 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 3.1, the following changes are required:

`Stage 3.` After determining the strip size δ_s by the relation

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

where q is the number of tracks (and hence of strips too), the strips are numbered from 1 to q.

`Stage 5.` This process uses a different strategy to construct the strip structure. In the algorithm for scattered data the strip size derives from the neighbourhood half-size to optimize the searching procedure of the nearby nodes. Conversely, the strip algorithm 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 two strips.

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

1. Computation of the ratio between the semi-size δ_y^L of square neighbourhood and the strip size δ_s, namely
$$k^* = \frac{\delta_y^L}{\delta_s} = q\delta_y^L.$$
Then, taking the smallest integer greater than k^*, i.e. $i^* = \lceil k^* \rceil$, $k^* \in \mathbb{R}^+$, we find the number of strips to be examined for each node.

2. Referring to the strip k, $k = 1, 2, \dots, q$, a strip searching procedure is applied, to examine the nodes from the $(k - i^*)$-th strip to the $(k + i^*)$-th strip.

Also in this case we need to reduce the total number of strips to be examined for the nodes of the first and last strips.

4 Modified spherical Shepard's method

In this section we describe a local method for the multivariate interpolation of large scattered data sets lying on the sphere $\mathbb{S}^{m-1}$. The scheme is based on the local use of zonal basis functions (i.e. ZBF interpolants as nodal functions) and represents a further variant of the well-known modified Shepard's method. Hence, this local interpolation approach exploits the characteristic of accuracy of ZBFs, overcoming common disadvantages as the instability due to the need of solving large linear systems (possibly, badly conditioned) and the inefficiency of the ZBF global interpolation method. A similar approach was already introduced at first by Pottmann and Eck [49] for MQs, and then by De Rossi [21] for ZBFs.

We consider the following definition of the modified spherical Shepard's method.

Definition 4.1. *Given a set $\mathcal{S}_n = \{x_i, i = 1, 2, \dots, n\}$ of distinct data points arbitrarily distributed on the sphere $\mathbb{S}^{m-1}$, with associated the corresponding set $\mathcal{F}_n = \{f_i, i = 1, 2, \dots, n\}$ of data values of an unknown function $f : \mathbb{S}^{m-1} \to \mathbb{R}$, the modified spherical Shepard's interpolant $F : \mathbb{S}^{m-1} \to \mathbb{R}$ takes the form*

$$F(x) = \sum_{j=1}^{n} Z_j(x) \bar{W}_j(x). \tag{7}$$

The nodal functions Z_j, $j = 1, 2, \dots, n$, are local approximants to f at x_j, constructed on the n_Z nodes closest to x_j and satisfying the interpolation conditions $Z_j(x_j) = f_j$. The weight functions $\bar{W}_j$, $j = 1, 2, \dots, n$, are given in (2) and (3), being $\alpha(\cdot, x_j) = \arccos(\cdot^T x_j)$ and

$$\tau(x, x_j) = \begin{cases} 1, & \textit{if } x \in \mathcal{C}(x_j; \rho), \\ 0, & \textit{otherwise}, \end{cases}$$

where $\mathcal{C}(x_j; \rho)$ is a spherical cap of centre at x_j and spherical radius ρ.

As regard to the choice of nodal functions we use a ZBF interpolant, which has the form

$$Z_j(x) = \sum_{i=1}^{n_Z} a_i \psi(g(x, x_i)), \qquad j = 1, 2, \dots, n, \tag{8}$$

where the zonal basis functions $\psi(g(\cdot, x_i))$ depend on the n_Z nodes of the considered neighbourhood of x_j, and $g(x, x_i) = \arccos(x^T x_i)$ is the geodesic distance. It is required that Z_j satisfies the interpolation conditions

$$Z_j(x_i) = f_i, \qquad i = 1, 2, \dots, n_Z.$$

Hence, to compute the coefficients $a = [a_1, a_2, \ldots, a_{n_Z}]^T$ in (8), it is required to solve uniquely the system of linear equations $Ka = f$, where $K = \{\psi(g(x_j, x_i))\}$ is a $n_Z \times n_Z$ matrix, f denotes the column vector of the function values f_j corresponding to the x_j.

In general, one can generate ZBFs by exploiting the results listed in [11], and by requiring, if possible, that the function ψ is (strictly) positive definite on the unit sphere. However, we observe that a certain number of ZBFs can be viewed as the specialization of the more general RBFs. In fact, given any Euclidean RBF, namely $\phi : [0, \infty) \to \mathbb{R}$, there is a natural way to associate it with a zonal basis function (or, in this case, more appropriately a *spherical radial basis function*). For instance in $\mathbb{R}^m$, since

$$||x - y||_2 = \sqrt{2 - 2x^T y} = 2 \sin \frac{g(x, y)}{2},$$

for any $x, y \in \mathbb{S}^{m-1}$, we have

$$\phi\left(||x - y||_2\right) = \psi\left(g(x, y)\right),$$

with $\psi(t) = \phi\left(2 \sin(t/2)\right)$, $t \in [0, \pi]$.

The most popular choices for ψ are

$$\begin{aligned}
\psi_1(t) &= \mathrm{e}^{-\alpha(2-2\cos t)}, && \text{(spherical Gaussian)}\\
\psi_2(t) &= \left(1 + \gamma^2 - 2\gamma \cos t\right)^{1/2}, && \text{(spherical MQ)}\\
\psi_3(t) &= \left(1 - \gamma^2\right)\left(1 + \gamma^2 - 2\gamma \cos t\right)^{3/2}, && \text{(spherical MQ II)}\\
\psi_4(t) &= \left(1 + \gamma^2 - 2\gamma \cos t\right)^{-1/2}, && \text{(spherical IMQ)}\\
\psi_5(t) &= \left(1 - \beta^2\right)\left(1 + \beta^2 - 2\beta \cos t\right)^{-3/2}, && \text{(Poisson spline)}\\
\psi_6(t) &= \beta^{-1} \log \left\{1 + 2\beta \left[1 - \beta + \left(1 + \beta^2 - 2\beta \cos t\right)^{1/2}\right]^{-1}\right\}, && \text{(logarithmic spline)}
\end{aligned}$$

where $\alpha > 0$, $\gamma, \beta \in (0, 1)$ and t measures the geodesic distance on the sphere. The spherical Gaussian [21] and the spherical inverse multiquadric (IMQ) [11, 23] are (strictly) positive definite functions on $\mathbb{S}^{m-1}$, $m \geq 1$, while the Poisson spline [30, 11] and the logarithmic spline [30, 35] are (strictly) positive definite functions on $\mathbb{S}^2$. This guarantees the existence of a unique solution of the considered system. Otherwise, as shown in [22], the spherical multiquadric (MQ) [23, 11] is (strictly) conditionally positive definite functions of order one (see [31] for further details). The spherical splines given in [35] are also of particular interest.

Therefore, there are many examples of strictly positive definite ZBFs, which can be used to solve the interpolation problem on the sphere. Sometimes, it can be highly advantageous to work with locally supported functions since they lead to sparse linear systems. Wendland [62] found a class of radial basis functions which are smooth and locally supported. Moreover, for any given m there is a Wendland's function that is strictly positive definite on $\mathbb{R}^m$ for that specific value of m. They consist of a product of a truncated power function and a low degree polynomial. Wendland's functions can be transformed to work directly with geodesic distance on the sphere assuming the form

$$\begin{aligned}
\psi_7(t) &= \left(1 - 2h \sin(t/2)\right)_+^4 \left(8h \sin(t/2) + 1\right), && \text{(spherical } C^2\text{-Wendland)}\\
\psi_8(t) &= \left(1 - 2h \sin(t/2)\right)_+^6 \left[35h^2 \left(2 \sin(t/2)\right)^2 + 18h \left(2 \sin(t/2)\right) + 3\right], && \text{(spherical } C^4\text{-Wendland)}
\end{aligned}$$

where h is a real positive number. The support of these functions is given by $[0, \arcsin(1/2h)]$. Some locally supported spherical RBF were constructed directly for the sphere (see [57, 23]). Other locally supported functions are discussed in [66].

5 Spherical zone algorithm

In this section we propose an extension of the strip algorithm to the spherical interpolation of large sets of scattered data or, with some modifications, to the spherical interpolation of track data lying on $\mathbb{S}^2 \subset \mathbb{R}^3$. The new algorithm is based on the partition of the sphere in a suitable number of parallel spherical zones, and, starting from these, on the construction for any data point of a circular neighbourhood (i.e., a spherical cap) containing a convenient number of data points. Then, the well-known modified Shepard's formula for spherical interpolation is applied with some effective improvements.

Since the strip algorithm has been already explained and the algorithm for the sphere, called the spherical zone algorithm, roughly follows the same pattern, in the following we are going to focus only on the parts in which they differ. The spherical setting leads to consider a spherical zone structure instead of a strip structure to organize data, and the square neighbourhoods are substituted by circular neighbourhoods (spherical caps) in a straightforward way. In particular, we remark that in the spherical zone algorithm two spherical zone structures are used to optimize the searching procedure, one to construct the circular neighbourhoods of the node, and the other in the evaluation phase, where we construct the circular neighbourhoods of the evaluation points. This trick improves the efficiency of the spherical zone algorithm compared with the strip one.

More in detail, in the spherical algorithm we construct for each node a local circular neighbourhood whose spherical radius is given by

$$\delta_Z = \arccos\left(1 - 2\sqrt{k_1}\frac{n_Z}{n}\right), \qquad k_1 = 1, 2, \dots,$$

where k_1 has the same meaning as in the strip algorithm. Thus, the number of spherical zones is found by taking

$$q = \left\lceil \frac{\pi}{\delta_Z} \right\rceil.$$

Then we construct on the sphere a suitable family of q spherical zones of equal width and parallel to the xy-plane. The set $\mathcal{S}_n$ of nodes is partitioned by the spherical zone structure, and, as in the strip algorithm, we define the number of spherical zones to be examined for each node.

A local interpolant Z_j, $j = 1, 2, \dots, n$, is found for each node, taking only the n_Z nodes closest to the node. To determine local weights for each node a spherical caps of radius

$$\delta_W = \arccos\left(1 - 2\sqrt{k_2}\frac{n_W}{n}\right), \qquad k_2 = 1, 2, \dots$$

is used. Then, we define the number

$$r = \left\lceil \frac{\pi}{\delta_W} \right\rceil,$$

in order to organize the data in a second family of spherical zones. A local weight function $\bar{W}_j$, $j = 1, 2, \dots, n$, is found considering only the n_W nodes closest to an evaluation point. Finally, we apply the modified spherical Shepard's formula (7).

All the considerations contained in Subsection 3.2 on track data can be also extended to the spherical case.

6 Complexity of the interpolation algorithms

The computational complexity of the strip interpolation algorithm is characterized by the use of the standard sorting routine `quicksort`, which requires on average a time complexity $\mathcal{O}(M \log M)$, where M is the number of nodes to be sorted. More precisely, we have a preprocessing phase for building the data structure, in which the computational cost has order:

- $\mathcal{O}(n \log n)$ for the first sorting of all n nodes;
- $\mathcal{O}(m_i \log m_i)$, $i = 1, 2, \dots, n$, to sort the nodes in the i–th local neighbourhood and, since $m_i \geq n_L$, for all neighbourhoods we have $\sum_{i=1}^{n} \mathcal{O}(m_i \log m_i) \geq n \cdot \mathcal{O}(n_L \log n_L)$.

Moreover, n linear systems of dimension n_L are to be solved in order to compute the coefficients of the local interpolants, thus requiring

- $\mathcal{O}(n \cdot n_L^3/6)$ and $\mathcal{O}(n \cdot 5^3/6)$ arithmetic operations for computing RBF interpolants and least squares approximants, respectively.

In the evaluation phase we support a computational cost of order:

- $s \cdot \mathcal{O}(n_W \log n_W)$ to sort the nodes of the local neighbourhoods which are centred at the evaluation points;
- $s \cdot \mathcal{O}(n_W \cdot n_L)$ for the evaluation of Shepard's interpolant at all evaluation points.

We remark that when the data structure is built, no further search time is required, since all points are stored in an ordered sequence. In particular, we point out that in our algorithms the number of nodes needed in each neighbourhood is prescribed, namely n_L and n_W in the two phases; it follows that the data structure is built in such a way that exactly n_L and n_W nodes belong to each neighbourhood. Finally, in the algorithm we employed $(m + 1) \cdot n \cdot n_L$ and $(m + 1) \cdot s \cdot n_W$ storage locations in the building of the data structure for the localization of nodal functions and Shepard's interpolant, respectively.

This complexity analysis can be directly extended to the spherical zone interpolation algorithm, substituting n_L by n_Z and using ZBFs instead of RBFs.

7 Numerical results

7.1 Experiments on bivariate interpolation

In this subsection we summarize the extensive and detailed investigation we performed to test and verify the proposed algorithm, especially for the sake 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 some sets of n randomly scattered and track nodes (x_i, y_i), for $i = 1, 2, \ldots, n$, in the square $[0,1] \times [0,1] \subset \mathbb{R}^2$, and the corresponding function values f_i. The pseudorandom nodes were obtained by using the Matlab command `rand`, which generates uniformly distributed random numbers on the interval (0,1). In particular, we generated track data sets choosing a certain number of lines, selecting some points on them and perturbing the coordinates of a random term belonging to $(0, \mu)$. The parameter μ is chosen such that two adjacent nodes along a given track are much closer together than nodes on different tracks. 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 $n = 2^{i-1} \cdot 10^3$, $i = 1, 2, \ldots, 5$. However, it is remarkable that, in general, also reducing considerably the number n of the scattered or track data (e.g., to a few thousand nodes), the proposed method 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. To give an idea in Figure 2, we plot two sets of $n = 1000$ scattered and track nodes.

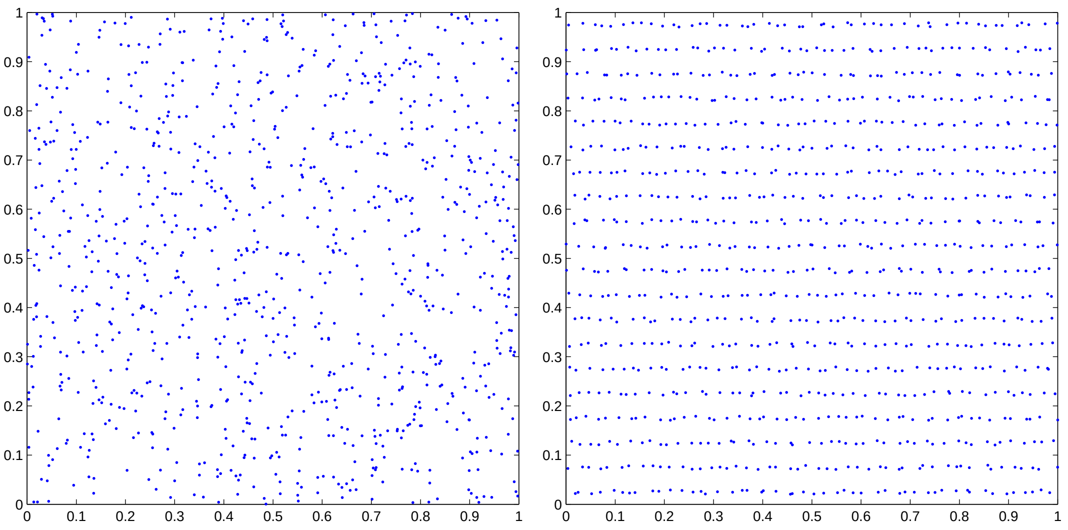


Figure 2: Plot of scattered (left) and track (right) data point sets ($n = 1000$).

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 [26, 28, 53, 39]), f_2 and f_3 (see [53, 39] and [53], respectively), and Nielson's test function f_4 (see [29]). The analytic expressions of such functions are:

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

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

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

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

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

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

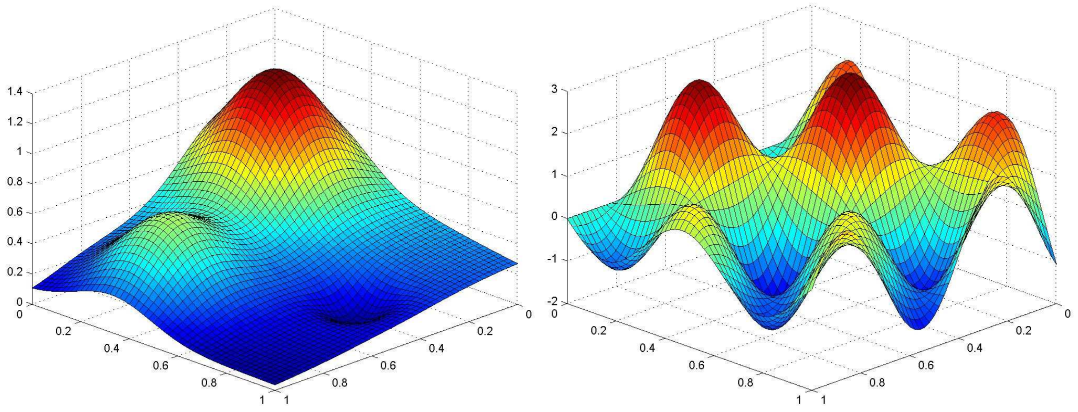


Figure 3: Test functions f_1 and f_2.

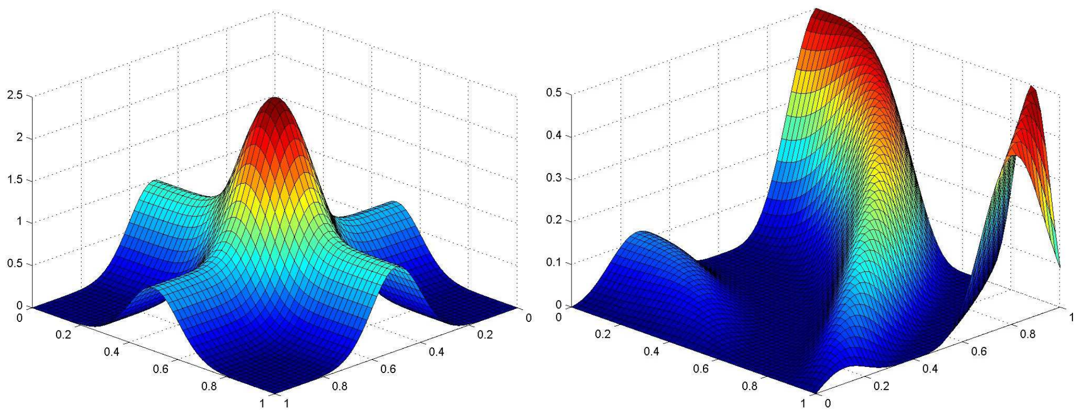


Figure 4: Test functions f_3 and f_4.

Renka's algorithm we used 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 to be compared with the strip one. The comparison was performed using in the strip algorithm the localizing function τ_2. In particular, when we used $\tau_2(t)$, ϵ was chosen such that $t = 1/2^\epsilon$. Therefore, after algebraic manipulations we obtained $\epsilon = -\log_2 2(\delta_y^L)^2$, δ_y^L being the half-size of the square neighbourhood. 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 they depend on the behaviour of the test function, the node distribution and the number of the scattered data points.

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

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

Table 1: MAEs and RMSEs for the function f_1.

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

Table 2: MAEs and RMSEs for the function f_2.

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

Table 3: MAEs and RMSEs for the function f_3.

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 probably given by the different choices of the nearest neighbours: Renka's algorithm

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

Table 4: MAEs and RMSEs for the function f_4.

works with circular neighbourhoods, while the strip one with square neighbourhoods. Moreover, the strip algorithm uses τ_2 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 – 8. The improvement is considerable, since the errors go down of one or two order of magnitude. This result is given by the faster convergence achieved by radial basis approximants in comparison with least squares approximants. The values of the shape parameters in RBFs were chosen to be $v = 2$, $\alpha^2 = 10$, and $c^2 = 0.1$, and we defined interpolants so that positive definiteness was guaranteed.

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

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

As we already pointed out, the strip algorithm organizes the 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 to be 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 Figure 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

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

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

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

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

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

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

function in the weights. Finally, we note that the execution time is only partially influenced by the number of evaluations at the grid points (see Table 10).

	Renka's Algorithm		Strip Algorithm		Strip Algorithm IMQ	
n	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).

Moreover, Tables 11 – 14 show the errors obtained for track data by running Renka's algorithm and the strip algorithm using τ_2 as localizing function, and $n_L = 13$, and $n_W = 8$ as localizing parameters for both methods. 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.

RMSEs and execution times for the three algorithms are listed in Table 15 and plotted in Figure 6. 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.

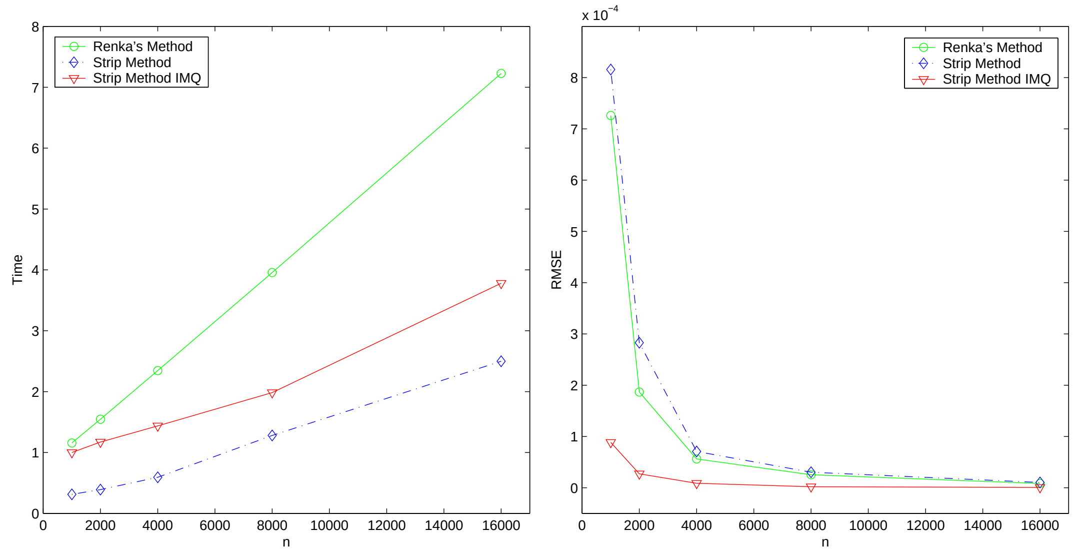


Figure 5: Execution times (left) and RMSEs (right) obtained by Renka's algorithm and the strip algorithm using the localizing function τ_1 with $n_L = 13$ and $n_W = 10$ for f_1 (scattered data).

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.

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

Table 11: MAEs and RMSEs 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	4.9131E − 2 4.8687E − 3	2.2466E − 2 1.6902E − 3	1.4329E − 2 9.3309E − 4	3.9565E − 3 2.4749E − 4	1.5992E − 3 8.5918E − 5
Strip Algorithm	1.1157E − 1 6.6890E − 3	6.5274E − 2 2.4470E − 3	1.6349E − 2 1.0451E − 3	5.9115E − 3 2.8643E − 4	1.7236E − 3 9.9225E − 5
Strip Algorithm IMQ	7.2506E − 3 6.9901E − 4	2.9117E − 3 1.8903E − 4	1.1772E − 3 4.9248E − 5	4.8850E − 4 1.5553E − 5	2.8893E − 5 2.3111E − 6

Table 12: MAEs and RMSEs 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	2.4941E − 2 2.4595E − 3	8.9470E − 3 6.3990E − 4	1.1832E − 2 5.2097E − 4	1.2963E − 3 9.3484E − 5	5.8299E − 4 3.8863E − 5
Strip Algorithm	3.1233E − 2 2.4415E − 3	7.1965E − 3 6.0235E − 4	8.9015E − 3 4.3469E − 4	7.3296E − 4 6.6895E − 5	7.4650E − 4 3.8891E − 5
Strip Algorithm IMQ	2.2564E − 3 2.0806E − 4	4.7039E − 4 4.5709E − 5	8.4037E − 5 1.2407E − 5	3.7584E − 5 3.8863E − 6	8.4982E − 6 7.3300E − 7

Table 13: MAEs and RMSEs 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	4.4156E − 2 1.8620E − 3	2.5727E − 2 8.5732E − 4	6.7194E − 3 2.8142E − 4	2.2599E − 3 9.1837E − 5	7.6237E − 4 2.8901E − 5
Strip Algorithm	9.3219E − 2 3.4486E − 3	2.0524E − 2 6.5429E − 4	5.6436E − 3 2.7522E − 4	3.4568E − 3 1.0017E − 4	8.5366E − 4 3.5563E − 5
Strip Algorithm IMQ	5.2565E − 3 3.0229E − 4	1.4817E − 3 6.8442E − 5	2.7202E − 3 1.6652E − 5	5.7315E − 4 1.3570E − 5	7.1047E − 5 1.6744E − 6

Table 14: MAEs and RMSEs 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).

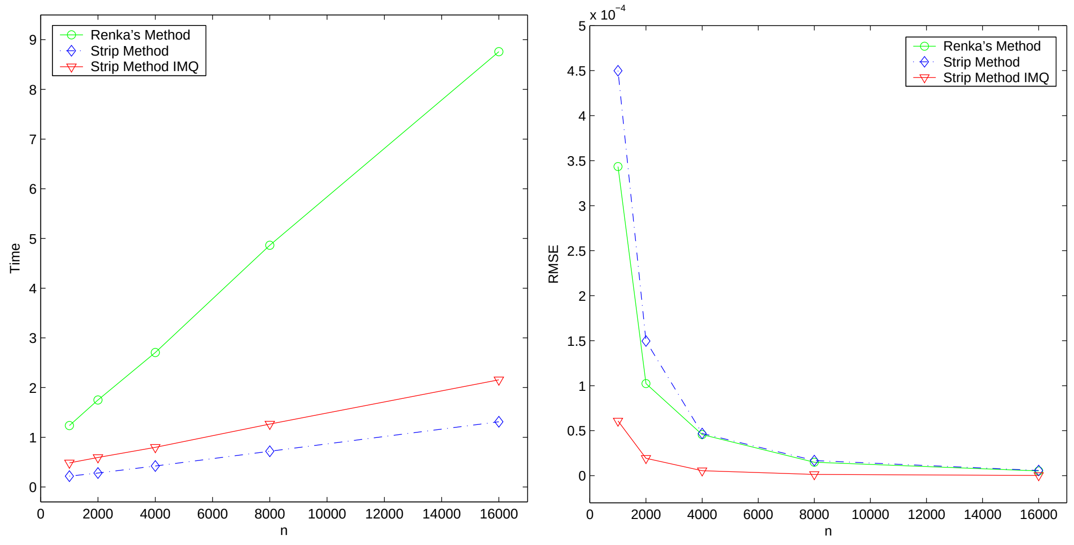


Figure 6: Execution times (left) and RMSEs (right) obtained by Renka's algorithm and the strip algorithm using the localizing function τ_1 with $n_L = 13$ and $n_W = 10$ for f_1 (track data).

7.2 Experiments on spherical interpolation

In this subsection we present a few of many numerical experiments we performed to test and verify the effectiveness of the proposed algorithm for scattered and track data on the sphere. In fact, this extensive and detailed investigation gives a numerical validation of the spherical algorithm, which we implemented in C/C++ language, turning to MATLAB environment for drawing some pictures. Also in this case, all the numerical results are obtained on a Pentium IV computer (2.8 GHz).

In the various tests we consider some data sets of dimension $n = 2^{i-1} \cdot 10^3$, $i = 1, 2, \dots, 5$. In particular, as scattered data points we take Halton points, that is pseudorandom data points generated on the sphere by using the program proposed in [65] (see also [33]). As track data points $(x_i, y_i, z_i) \in \mathbb{S}^2 \subset \mathbb{R}^3$, for $i = 1, 2, \dots, n$, we consider random nodes collocated on or close to parallel circles which satisfy the basic property characterizing this kind of nodes. Examples of $n = 1000$ Halton and track data points are shown in Figure 7 (a) and (b), respectively.

Now, in order to have an idea of the node distribution and to understand how uniform are the data sets, we consider two common indicators of data regularity: the *separation distance* and the *fill distance*. The former, also referred to as packing radius, is given by

$$q_{\mathcal{S}_n} = \frac{1}{2} \min_{i \neq j} g(x_i, x_j),$$

while the latter, which is a measure of the data distribution, is usually defined as

$$h_{\mathcal{S}_n, \mathcal{E}_s} = \sup_{x \in \mathcal{E}_s} \min_{x_j \in \mathcal{S}_n} g(x, x_j),$$

where $g(\cdot, x_j)$ is the geodesic distance. In Table 16 and Table 17 we report the value of $q_{\mathcal{S}_n}$ and $h_{\mathcal{S}_n, \mathcal{E}_s}$ for Halton and track data points, respectively.

n	1000	2000	4000	8000	16000
$q_{\mathcal{S}_n}$	5.5567E − 3	2.6953E − 3	4.8397E − 4	4.8397E − 4	4.8397E − 4
$h_{\mathcal{S}_n, \mathcal{E}_s}$	1.0778E − 1	8.9836E − 2	6.5901E − 2	4.3906E − 2	3.0152E − 2

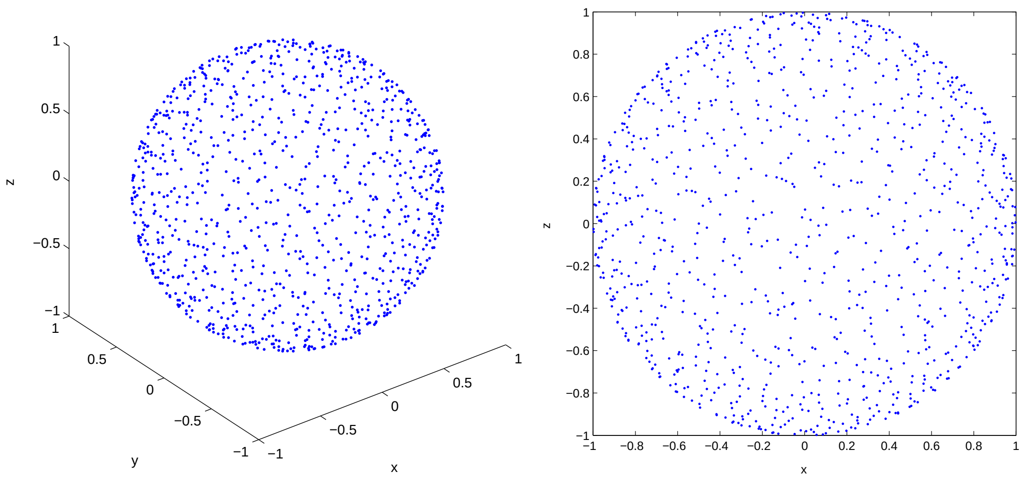
Table 16: Separation distance $q_{\mathcal{S}_n}$ and fill distance $h_{\mathcal{S}_n, \mathcal{E}_s}$ for Halton data points on the sphere by varying n.

The performance of our algorithm is verified taking the data values by the restriction on $\mathbb{S}^2$ of the following trivariate test functions [49, 35], the last one being the well-known Franke's test function (see [50]):

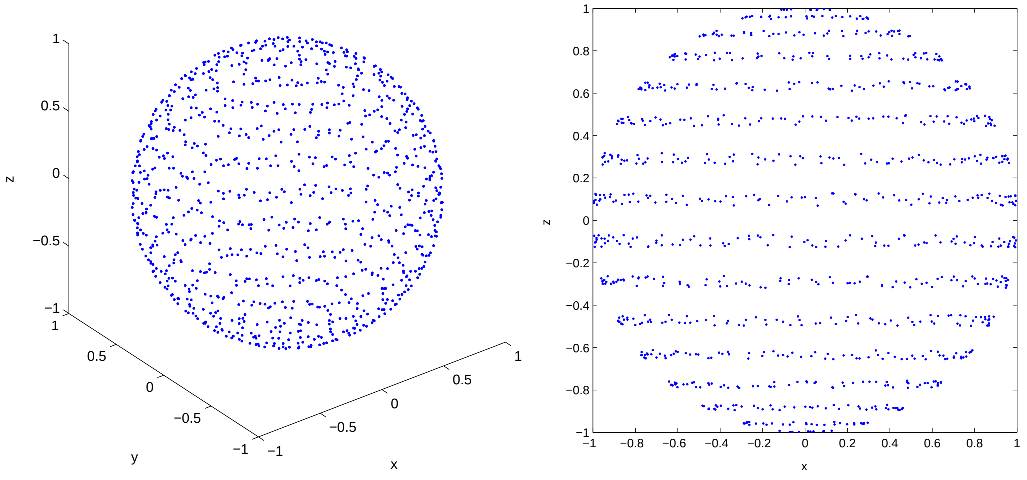
$$f_1(x, y, z) = \frac{1 + 2x + 3y + 4z}{6},$$

$$f_2(x, y, z) = \frac{9x^3 - 2x^2y + 3xy^2 - 4y^3 + 2z^3 - xyz}{10},$$

$$f_3(x, y, z) = \frac{\mathrm{e}^x + 2\mathrm{e}^{y+z}}{10},$$



(a)



(b)

Figure 7: Plots of Halton data points (a) and track data points (b) on the unit sphere ($n = 1000$).

n	1000	2000	4000	8000	16000
$q_{\mathcal{S}_n}$	4.4123E − 3	2.7094E − 3	1.3627E − 3	7.5835E − 4	2.6344E − 4
$h_{\mathcal{S}_n,\mathcal{E}_s}$	1.1363E − 1	7.9741E − 2	5.5800E − 2	3.9020E − 2	2.9325E − 2

Table 17: Separation distance $q_{\mathcal{S}_n}$ and fill distance $h_{\mathcal{S}_n,\mathcal{E}_s}$ for track data points on the sphere by varying n.

$$f_4(x,y,z) = \sin x \sin y \sin z,$$

$$f_5(x,y,z) = \frac{3}{4}\exp\left[-\frac{(9x-2)^2+(9y-2)^2+(9z-2)^2}{4}\right] + \frac{3}{4}\exp\left[-\frac{(9x+1)^2}{49} - \frac{9y+1}{10} - \frac{9z+1}{10}\right]$$
$$+\frac{1}{2}\exp\left[-\frac{(9x-7)^2+(9y-3)^2+(9z-5)^2}{4}\right] - \frac{1}{5}\exp\left[-(9x-4)^2-(9y-7)^2-(9z-5)^2\right].$$

Some information about the execution of the interpolation algorithm described in Section 5 is reported in Table 18 and Table 19; specifically, in the track data cases we note that the spherical zones to be examined are 5 (i.e. $k^* = 5$) as a rule, and at least 3 ($k^* = 3$) as a minimum.

	Localization phase			Evaluation phase		
n	# zones	δ_Z	# iter k_1	# zones	δ_W	# iter k_2
1000	11	0.3020	2	14	0.2261	2
2000	15	0.2131	2	18	0.1768	3
4000	21	0.1506	2	26	0.1249	3
8000	30	0.1064	2	36	0.0883	3
16000	38	0.0833	3	51	0.0624	3

Table 18: Information on spherical algorithm for Halton data points.

We are also interested to stress the effectiveness of the spherical zone searching procedure which allows to reduce the CPU times. For this reason, we make a comparison among three algorithms: the spherical interpolation algorithm implemented by using the zone structure on the sphere, the algorithm where quicksort routines are used but the sphere $\mathbb{S}^2$ is not partitioned in spherical zones, and the algorithm proposed in [21] using the 3D cell-search method (see Table 20). These results are obtained by using Halton data points and the ZBF ψ_6 with localization parameters $n_Z = 15$ and $n_W = 10$ for the test function f_1. We emphasize that the use of the zone structure produces a considerable saving of time. This result was expected taking into account the computational costs reported in Section 6.

Then, we focus our attention also on the computation of errors. MAEs and RMSEs are

	Localization phase			Evaluation phase		
n	# zones	δ_Z	# iter k_1	# zones	δ_W	# iter k_2
1000	16	0.3020	2	16	0.2261	2
2000	22	0.2131	2	22	0.1597	2
4000	32	0.1506	2	32	0.1129	2
8000	46	0.0798	2	46	0.1064	2
16000	64	0.0752	2	64	0.0564	2

Table 19: Information on spherical algorithm for track data points.

n	ZONE STRUCTURE	NO-ZONE STRUCTURE	3D CELL SEARCH
1000	0.481	0.751	166.399
2000	0.841	1.823	375.449
4000	1.702	5.548	916.067
8000	4.026	18.697	2164.322
16000	9.264	68.379	5604.887

Table 20: CPU times (in seconds) obtained by running the spherical zone algorithm, the algorithm without partitioning in zones, and the 3D cell-search algorithm.

computed by evaluating the interpolants on a set of $s = 600$ evaluation points, which are generated by the spiral method of Saff and Kuijlaars [54] and provide a uniform distribution on the unit sphere. In Tables 21 and 22 we show the MAEs and RMSEs achieved on Halton points by using ψ_4 and ψ_6 with $\gamma = \beta = 0.7$, $n_Z = 15$ and $n_W = 10$, for the first four test functions.

We note that the local scheme is very accurate, especially when the number of data points grows, even if no search of optimal values for the parameters is performed.

Since the choice of appropriate localizing parameters n_Z and n_W is a non-trivial problem, we performed several tests finding via trials and errors good results for $n_Z = 16$ and $n_W = 9$. These values are not the only allowable; in fact, there are many elements which influence the final results, such as the data point distribution (in particular the separation distance), the kind of ZBF basis, the relative value of ZBF's shape parameter (i.e., if a basis function is "flat" or "peaked"), the behaviour of the data values (test functions), etc..

In Figures 8 – 11, we plot the behaviour of the RMSEs by varying the shape parameter for the different zonal basis functions (from ψ_1 to ψ_8) when the data are scattered, whereas Figure 12 gives an example for the track data case (only with ψ_1 and ψ_2). All results are obtained on test function f_5. Note that each evaluation is achieved by subdividing the interval of the shape

n	f_1	f_2	f_3	f_4
1000	2.5101E − 3	3.7769E − 3	2.0022E − 3	2.0022E − 3
	5.0926E − 4	5.1455E − 4	3.3969E − 4	3.3969E − 4
2000	1.2924E − 3	1.0688E − 3	8.5904E − 4	5.8907E − 4
	2.0323E − 4	1.7905E − 4	1.4012E − 4	7.3540E − 5
4000	2.4042E − 4	3.1177E − 4	1.9583E − 4	1.3363E − 4
	4.7891E − 5	4.9201E − 5	3.3424E − 5	1.6935E − 5
8000	7.4362E − 5	6.4579E − 5	4.0073E − 5	2.4360E − 5
	1.2049E − 5	1.0369E − 5	7.8723E − 6	4.0642E − 6
16000	4.5552E − 5	2.2178E − 5	1.8007E − 5	4.9581E − 6
	3.3560E − 6	2.8248E − 6	1.9938E − 6	9.0486E − 7

Table 21: MAEs and RMSEs computed on spiral points by using ψ_4.

n	f_1	f_2	f_3	f_4
1000	1.2445E − 3	1.9831E − 3	4.9512E − 4	4.8661E − 4
	2.0671E − 4	2.4179E − 4	1.0690E − 4	1.0567E − 4
2000	3.0853E − 4	5.0038E − 4	2.5460E − 4	2.6100E − 4
	7.1861E − 5	7.8670E − 5	4.2001E − 5	3.7240E − 5
4000	9.7610E − 5	1.3159E − 4	5.8100E − 5	5.3480E − 5
	1.7739E − 5	2.1183E − 5	9.8421E − 6	8.1228E − 6
8000	2.7372E − 5	2.7909E − 5	1.4293E − 5	1.1469E − 5
	4.4705E − 6	4.3736E − 6	2.3327E − 6	2.0051E − 6
16000	1.8184E − 5	8.0451E − 6	6.9378E − 6	2.9160E − 6
	1.2999E − 6	1.1528E − 6	6.3696E − 7	4.4485E − 7

Table 22: MAEs and RMSEs computed on spiral points by using ψ_6.

parameter into 100 equispaced values.

By analyzing numerical tests and the related pictures, we observe that the spherical Wendland's functions ψ_7 and ψ_8, which have compact support, reveal a great stability, but they do not have a so good accuracy as classical zonal basis functions (not compactly supported). However, these graphs give an idea on the stability and enable us to choose good values for the shape parameters.

Finally, we use a (fixed) good value of the shape parameter to obtain the numerical results pointed out in Tables 23 – 28 using ZBFs $\psi_1 - \psi_6$ and in Tables 29 – 30 using Wendland's functions for scattered data. All tables also contain the exponent of the observed RMSE-convergence rate $\mathcal{O}(h^{rate})$, which is found by using the formula

$$\text{rate}_k = \frac{\ln\left(e_{k-1}/e_k\right)}{\ln\left(h_{k-1}/h_k\right)}, \qquad k = 2, 3, \ldots,$$

where the symbol e_k is the k-th RMSE, and h_k denotes the fill distance of the k-th computational set.

Observing these errors allows the choice of optimal values for the parameters in the zonal basis functions, thus obtaining good accuracy and therefore high order of convergence. However, we remark that convergence orders have not a uniform behaviour. The explanation of this phenomenon may be found in the really scattered nature of the data sets considered in numerical tests.

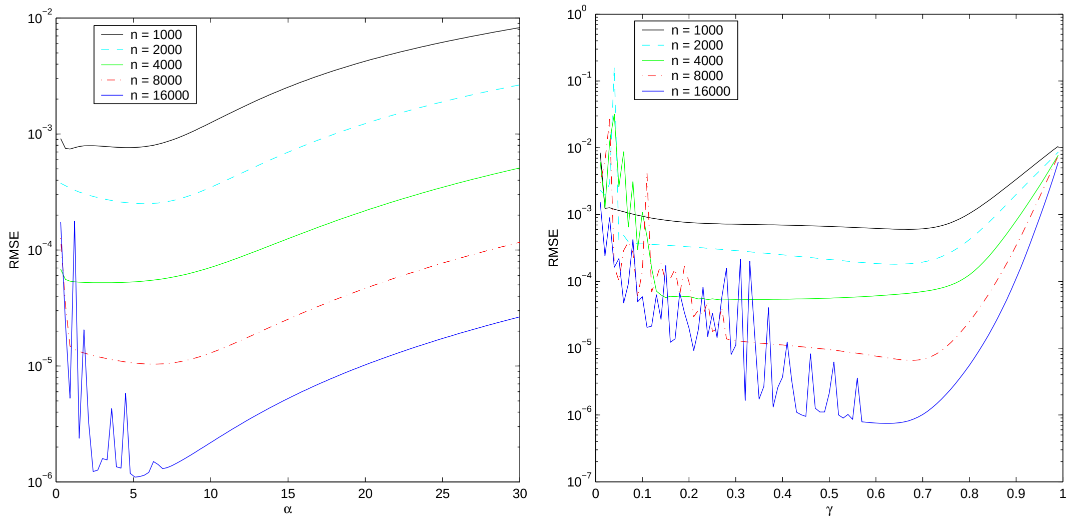


Figure 8: RMSEs obtained on Halton data points by varying the shape parameter α of ψ_1 (left) and γ of ψ_2 (right).

In Tables 31 – 32 we list the results obtained for track data, taking into account errors reported in the following Figure 12, which refers to the ZBFs ψ_1 and ψ_2. Also for the special case of track data we observe the high accuracy of the method.

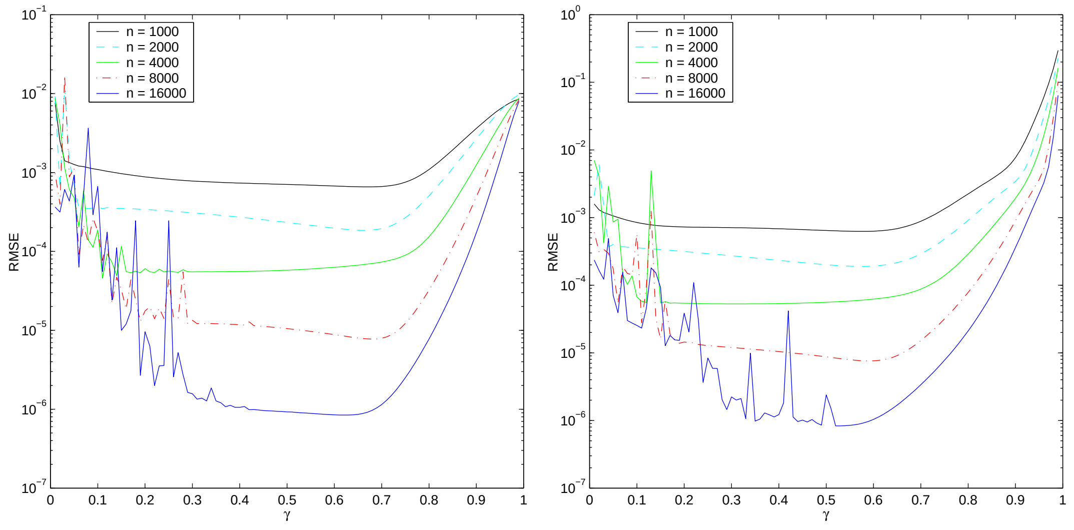


Figure 9: RMSEs obtained on Halton data points by varying the shape parameter γ of ψ_3 (left) and ψ_4 (right).

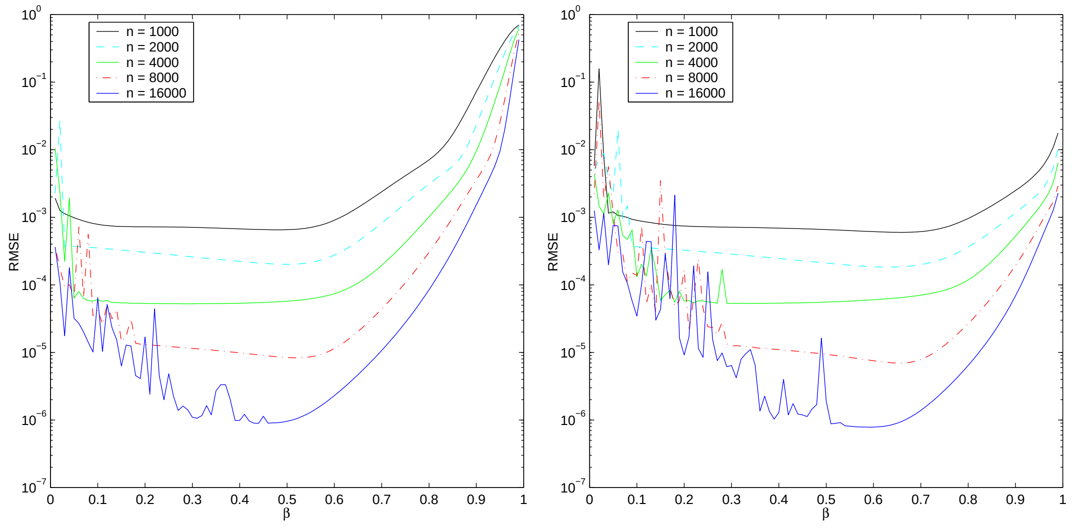


Figure 10: RMSEs obtained on Halton data points by varying the shape parameter β of ψ_5 (left) and ψ_6 (right).

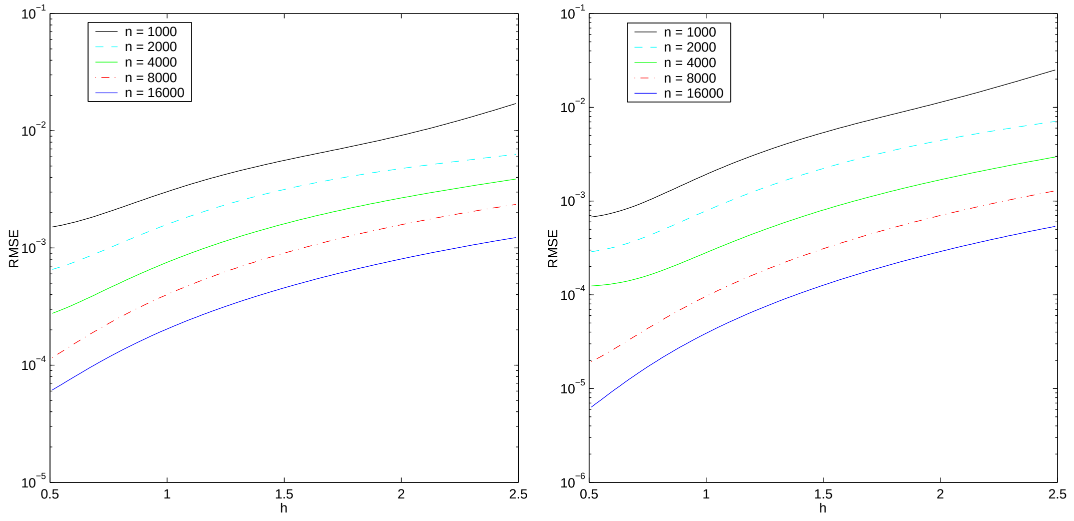


Figure 11: RMSEs obtained on Halton data points by varying the shape parameter h of ψ_7 (left) and ψ_8 (right).

n	MAE	RMSE	rate
1000	1.0593E − 2	1.2491E − 3	–
2000	3.8277E − 3	3.4669E − 4	7.0385
4000	8.7404E − 4	7.0771E − 5	5.1285
8000	1.3837E − 4	1.2958E − 5	4.1806
16000	1.8961E − 5	2.1869E − 6	4.7345

Table 23: MAEs, RMSEs, and RMSE-convergence rates obtained by using ψ_1 for $\alpha = 10$ (Halton data).

n	MAE	RMSE	rate
1000	6.2711E − 3	6.2056E − 4	–
2000	3.0984E − 3	1.8504E − 4	6.6447
4000	1.1786E − 3	6.0627E − 5	3.6014
8000	1.2006E − 4	7.6050E − 6	5.1118
16000	1.3208E − 5	7.5592E − 7	6.1432

Table 24: MAEs, RMSEs, and RMSE-convergence rates obtained by using ψ_2 for $\gamma = 0.6$ (Halton data).

n	MAE	RMSE	rate
1000	6.8931E − 3	6.7388E − 4	–
2000	2.9525E − 3	1.9703E − 4	6.7526
4000	1.2910E − 3	6.2842E − 5	3.6882
8000	1.5016E − 4	8.8239E − 6	4.8341
16000	1.4570E − 5	8.4902E − 7	6.2297

Table 25: MAEs, RMSEs, and RMSE-convergence rates obtained by using ψ_3 for $\gamma = 0.6$ (Halton data).

n	MAE	RMSE	rate
1000	6.0126E − 3	6.2923E − 4	–
2000	2.9243E − 3	1.9192E − 4	6.5205
4000	1.0855E − 3	6.2297E − 5	3.6315
8000	9.5441E − 5	7.6118E − 6	5.1766
16000	1.1349E − 5	1.0378E − 6	5.3023

Table 26: MAEs, RMSEs, and RMSE-convergence rates obtained by using ψ_4 for $\gamma = 0.6$ (Halton data).

n	MAE	RMSE	rate
1000	6.0533E − 3	9.0421E − 4	–
2000	2.6274E − 3	2.7539E − 4	6.5285
4000	1.0009E − 3	7.4065E − 5	4.2386
8000	7.6038E − 5	1.1420E − 5	4.6037
16000	1.8155E − 5	2.2952E − 6	4.2697

Table 27: MAEs, RMSEs, and RMSE-convergence rates obtained by using ψ_5 for $\beta = 0.6$ (Halton data).

n	MAE	RMSE	rate
1000	6.1548E − 3	6.1353E − 4	–
2000	3.1113E − 3	1.8543E − 4	6.5706
4000	1.1379E − 3	6.0164E − 5	3.6329
8000	1.1595E − 4	7.5365E − 6	5.1152
16000	1.2749E − 5	7.8507E − 7	6.0185

Table 28: MAEs, RMSEs, and RMSE-convergence rates obtained by using ψ_6 for $\beta = 0.6$ (Halton data).

n	MAE	RMSE	rate
1000	1.9172E − 2	1.4975E − 3	–
2000	8.2313E − 3	6.4699E − 4	4.6084
4000	4.0149E − 3	2.7230E − 4	2.7932
8000	7.5092E − 4	1.1269E − 4	2.1725
16000	5.3943E − 4	5.9685E − 5	1.6912

Table 29: MAEs, RMSEs, and RMSE-convergence rates obtained by using ψ_7 for $h = 0.5$ (Halton data).

n	MAE	RMSE	rate
1000	6.6524E − 3	6.7099E − 4	–
2000	3.2808E − 3	2.8618E − 4	4.6793
4000	2.1175E − 3	1.2394E − 4	2.7009
8000	1.5393E − 4	1.8853E − 5	4.6371
16000	5.6685E − 5	6.0762E − 6	3.0130

Table 30: MAEs, RMSEs, and RMSE-convergence rates obtained by using ψ_8 for $h = 0.5$ (Halton data).

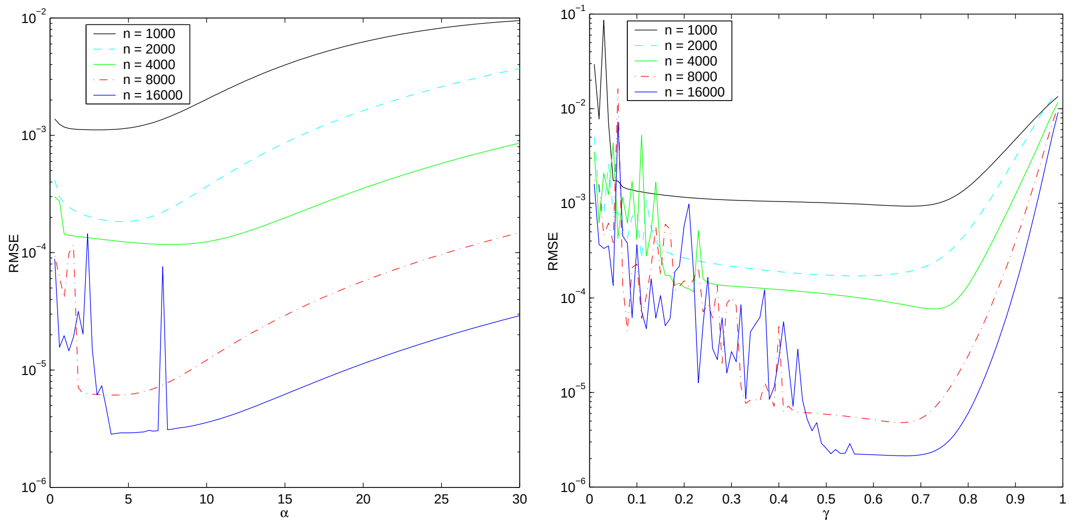


Figure 12: RMSEs obtained on track data points by varying the shape parameter α of ψ_1 (left) and γ of ψ_2 (right).

n	MAE	RMSE	rate
1000	1.6364E − 2	2.0271E − 3	–
2000	3.0162E − 3	3.6511E − 4	4.8400
4000	2.4065E − 3	1.2398E − 4	3.0253
8000	7.2649E − 5	1.2154E − 5	6.4928
16000	4.0557E − 5	3.5696E − 6	4.2894

Table 31: MAEs, RMSEs, and RMSE-convergence rates obtained by using ψ_1 for $\alpha = 10$ (track data).

n	MAE	RMSE	rate
1000	1.4950E − 2	9.6467E − 4	–
2000	3.1752E − 3	1.7236E − 4	4.8627
4000	2.2226E − 3	9.5659E − 5	1.6492
8000	6.7671E − 5	5.1579E − 6	8.1640
16000	3.4004E − 5	2.1975E − 6	2.9871

Table 32: MAEs, RMSEs, and RMSE-convergence rates obtained by using ψ_2 for $\gamma = 0.6$ (track data).

Finally, we compared our algorithm with the ACM Algorithm 773 by R. J. Renka [52]. In that paper methods and software were proposed to construct C^0 and C^1 interpolants. Briefly, they consist of generating a triangulation of the nodes, estimating locally (LG) or globally (GG) gradients at the nodes (for the C^1 interpolants), and constructing a triangle-based interpolant of the data and gradient estimates. Numerical results concerning the comparison with our code in terms of accuracy and efficiency are listed in Table 33. We considered the test function f_1 and in the modified spherical Shepard's method the zonal basis functions ψ_2 with $\gamma = 0.0002$. The input and output files for the two algorithms are the same, that is the data files already used in the other tests and the output file of the 600 spiral points used as evaluation data set. Table 33 shows that CPU times required by the two algorithms are quite equivalent, while the algorithm using zonal basis functions and the zonal searching procedure is slightly more accurate. These results were obtained considering a small number of nodes for the localization, i.e. $n_Z = n_W = 5$. We remark that the choice of the localization parameters can lead to a good compromise between accuracy and efficiency, in such a way that our algorithm is comparable with the Renka one, and it could be even more accurate.

n	8000	16000
C^0 method	1.3677E − 4 0.953	6.9565E − 5 3.047
C^1 LG method	4.0845E − 6 1.297	6.4932E − 6 3.516
C^1 GG method	4.0901E − 6 1.750	6.4917E − 6 4.094
ZBF method	3.1288E − 7 1.531	1.6786E − 7 4.265

Table 33: Comparison with the ACM Algorithm 773: RMSEs and CPU times (in seconds) on Halton points for f_1.

Acknowledgements

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

References

[1] P. Alfeld, M. Neamtu, L. L. Schumaker, Fitting scattered data on sphere-like surfaces using spherical splines, *J. Comput. Appl. Math.* **73** (1996), 5–43.

[2] 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.

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

[4] 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.

[5] 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.

[6] G. Allasia, R. Besenghi, R. Cavoretto, A. De Rossi, A strip method for continuous surface modelling from scattered and track data, Quaderno Scientifico del Dipartimento di Matematica, Università di Torino, n.2/2009, 1–23, submitted.

[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, B. R. Piper, K. L. Rescorla, Interpolation to arbitrary data on a surface, in: G. E. Farin (Ed.), *Geometric Modeling*, SIAM, Philadelphia, PA, 1987, pp. 281–289.

[10] R. E. Barnhill, H. S. Ou, Surfaces defined on surfaces, *Comput. Aided Geom. Design* **7** (1990), 323–336.

[11] B. J. C. Baxter, S. Hubbert, Radial basis function on the sphere, in: *Recent Progress in Multivariate Approximation*, Internat. Ser. Numer. Math., vol. 137, Birkhäuser, Basel, Switzerland, 2001, pp. 33–47.

[12] B. J. C. Baxter, On spherical averages of radial basis functions, *Found. Comput. Math.* **8** (2008), 395–407.

[13] M. D. Buhmann, *Radial Basis Functions: Theory and Implementation*, Cambridge Monogr. Appl. Comput. Math., vol. 12, Cambridge Univ. Press, Cambridge, 2003.

[14] R. E. Carlson, T. A. Foley, Interpolation of track data with radial basis methods, *Comput. Math. Appl.* **12** (1992), 27–34.

[15] 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, A. A. 2005–06.

[16] R. Cavoretto, A. De Rossi, A spherical interpolation algorithm using zonal basis functions, 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. 258–269.

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

[18] R. Cavoretto, A. De Rossi, Fast and accurate interpolation of large scattered data sets on the sphere, *J. Comput. Appl. Math.* **234** (2010), 1505–1521.

[19] E. W. Cheney, Approximation and interpolation on the spheres, in: S. P. Singh (Ed.), *Approximation Theory, Wavelets and Applications*, Kluwer Acad. Publ., Dordrecht, 1995, pp. 47–53.

[20] A. Crampton, J. C. Mason, Surface approximation of curved data using separable radial basis functions, in: P. Ciarlini et al. (Eds.), *Advanced Mathematical and Computational Tools in Metrology V*, World Scientific, Singapore, 2000, pp. 119–126.

[21] 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.

[22] G. E. Fasshauer, Radial basis functions on spheres, Ph.D. Dissertation, Vanderbilt University, 1995.

[23] G. E. Fasshauer, L. L. Schumaker, Scattered data fitting on the sphere, in: M. Dæhlen, T. Lyche, L. L. Schumaker (Eds.), *Mathematical Methods for Curves and Surfaces*, Vanderbilt Univ. Press, Nashville, TN, 1998, pp. 117–166.

[24] G. E. Fasshauer, *Meshfree Approximation Methods with MATLAB*, World Scientific Publishers, Singapore, 2007.

[25] T. A. Foley, G. M. Nielson, Multivariate interpolation to scattered data using delta iteration, in: E. W. Cheney (Ed.), *Approximation Theory III*, Academic Press, New York, 1980, pp. 419–424.

[26] 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.

[27] R. Franke, G. Nielson, Smooth interpolation of large sets of scattered data, *Internat. J. Numer. Methods Engrg.* **15** (1980), 1691–1704.

[28] R. Franke, Scattered data interpolation: tests of some methods, *Math. Comp.* **38** (1982), 181–200.

[29] R. Franke, H. Hagen, Least squares surface approximation using multiquadrics and parametric domain distorsion, *Comput. Aided Geom. Design* **16** (1999), 177–196.

[30] W. Freeden, Spherical spline interpolation: basic theory and computational aspects, *J. Comput. Appl. Math.* **11** (1984), 367–375.

[31] W. Freeden, T. Gervens, M. Schreiner, *Constructive Approximation on the Sphere with Applications to Geomathematics*, Clarendon Press, Oxford, 1998.

[32] E. J. Fuselier, F. J. Narcowich, J. D. Ward, G. B. Wright, Error and stability estimates for surface-divergence free RBF interpolants on the sphere, *Math. Comp.* **78** (2009), 2157–2186.

[33] J. H. Halton, On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals, *Numer. Math.* **2** (1960), 84–90.

[34] K. Hesse, H. N. Mhaskar, I. H. Sloan, Quadrature in Besov spaces on the Euclidean sphere, *J. Complexity* **23** (2007), 528–552.

[35] S. Hubbert, *Radial Basis Function Interpolation on the Sphere*, PhD Thesis, Imperial College London, 2002.

[36] S. Hubbert, T. Morton, On the accuracy of surface spline interpolation on the unit sphere, in: M. Neamtu, E. B. Saff (Eds.) *Advances in Constructive Approximation*, Mod. Methods Math., Nashboro Press, Brentwood, TN, 2004, pp. 227–242.

[37] O. Kounchev, *Multivariate Polysplines: Applications to Numerical and Wavelet Analysis*, Academic Press, San Diego, CA, 2001.

[38] M.-J. Lai, L. L. Schumaker, *Spline Functions on Triangulations*, Encyclopedia of Mathematics and its Applications, vol. 110, Cambridge Univ. Press, Cambridge, 2007.

[39] 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.

[40] Q. T. Le Gia, K. Hesse, Local radial basis function approximation on the sphere, *Bull. Aust. Math. Soc.* **77** (2008), 197–224.

[41] Q. T. Le Gia, F. J. Narcowich, J. D. Ward, H. Wendland, Continuous and discrete least-squares approximation by radial basis functions on spheres, *J. Approx. Theory* **143** (2006), 124–133.

[42] Q. T. Le Gia, T. Tran, Interpolation on the sphere: a fast solution technique, *ANZIAM J.* (E), **50** (2008), C354–C370.

[43] V. A. Menegatto, Strictly positive definite kernels on the circle, *Rocky Mountain J. Math.* **25** (1995), 1149–1163.

[44] H. N. Mhaskar, F. J. Narcowich, J. D. Ward, Approximation properties of zonal function networks using scattered data on the sphere, *Adv. Comput. Math.* **11** (1999), 121–137.

[45] H. N. Mhaskar, Weighted quadrature formulas and approximation by zonal function networks on the sphere, *J. Complexity* **22** (2006), 348–370.

[46] F. J. Narcowich, J. D. Ward, Scattered data interpolation on spheres: error estimates and locally supported basis functions, *SIAM J. Math. Anal.* **33** (2002), 1393–1410.

[47] F. J. Narcowich, X. Sun, J. D. Ward, H. Wendland, Direct and inverse Sobolev error estimates for scattered data interpolation via spherical basis functions, *Found. Comput. Math.* **7** (2007), 369–390.

[48] F. J. Narcowich, X. Sun, J. D. Ward, Approximation power of RBFs and their associated SBFs: a connection, *Adv. Comput. Math.* **27** (2007), 107–124.

[49] H. Pottmann, M. Eck, Modified multiquadric methods for scattered data interpolation over a sphere, *Comput. Aided Geom. Design* **7** (1990), 313–321.

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

[51] R. J. Renka, Algorithm 660: QSHEP2D: Quadratic Shepard method for bivariate interpolation of scattered data, *ACM Trans. Math. Software* **14** (1988), 149–150.

[52] R. J. Renka, Algorithm 773: SSRFPACK: interpolation of scattered data on the surface of a sphere with a surface under tension, *ACM Trans. Math. Software* **23** (1997), 435–442.

[53] 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.

[54] E. Saff, A. B. J. Kuijlaars, Distributing many points on a sphere, *Math. Intelligencer* **19** (1997), 5–11.

[55] 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.

[56] I. J. Schoenberg, Positive definite functions on the spheres, *Duke Math. J.* **9** (1942), 96–108.

[57] M. Schreiner, Locally supported kernels for spherical spline interpolation, *J. Approx. Theory* **89** (1997), 172–194.

[58] I. H. Sloan, R. S. Womersley, Constructive polynomial approximation on the sphere, *J. Approx. Theory* **103** (2000), 91–118.

[59] I. H. Sloan, R. S. Womersley, Extremal systems of points and numerical integration on the sphere, *Adv. Comput. Math.* **21** (2004), 107–125.

[60] I. H. Sloan, A. Sommariva, Approximation on the sphere using radial basis functions plus polynomials, *Adv. Comput. Math.* **29** (2008), 147–177.

[61] I. H. Sloan, H. Wendland, Inf-sup condition for spherical polynomials and radial basis functions on spheres, *Math. Comp.* **78** (2009), 1319–1331.

[62] H. Wendland, Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree, *Adv. Comput. Math.* **4** (1995), 389–396.

[63] H. Wendland, Moving least squares approximation on the sphere, in: T. Lyche, L. L. Schumaker (Eds.), *Mathematical Methods for Curves and Surfaces*, Vanderbilt Univ. Press, Nashville, TN, 2001, pp. 517–526.

[64] H. Wendland, *Scattered Data Approximation*, Cambridge Monogr. Appl. Comput. Math., vol. 17, Cambridge Univ. Press, Cambridge, 2005.

[65] T.-T. Wong, W.-S. Luk, P.-A. Heng, Sampling with Hammersley and Halton points, *J. Graphics Tools* **2** (1997), 9–24.

[66] Z. Wu, Multivariate compactly supported positive definite radial functions, *Adv. Comput. Math.* **4** (1995), 283–292.

[67] Y. Xu, E. W. Cheney, Strictly positive definite functions on the spheres, *Proc. Amer. Math. Soc.* **116** (1992), 977–981.