Computers and Mathematics with Applications 67 (2014) 1024–1038

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

A meshless interpolation algorithm using a cell-based searching procedure

Roberto Cavoretto *, Alessandra De Rossi

Department of Mathematics "G. Peano", University of Torino, via Carlo Alberto 10, I–10123 Torino, Italy

ARTICLE INFO

Article history:
Received 18 June 2013
Received in revised form 4 December 2013
Accepted 6 January 2014

Keywords:
Meshless approximation
Fast algorithms
Partition of unity methods
Radial basis functions
Scattered data

ABSTRACT

In this paper we propose a fast algorithm for bivariate interpolation of large scattered data sets. It is based on the partition of unity method for constructing a global interpolant by blending radial basis functions as local approximants and using locally supported weight functions. The partition of unity algorithm is efficiently implemented and optimized by connecting the method with an effective cell-based searching procedure. More precisely, we construct a cell structure, which partitions the domain and strictly depends on the dimension of the subdomains, thus providing a meaningful improvement in the searching process compared to the nearest neighbour searching techniques presented in Allasia et al. (2011) and Cavoretto and De Rossi (2010, 2012). In fact, this efficient algorithm and, in particular, the new searching procedure enable us a fast computation also in several applications, where the amount of data to be interpolated is often very large, up to many thousands or even millions of points. Analysis of computational complexity shows the high efficiency of the proposed interpolation algorithm. This is also supported by numerical experiments.

1. Introduction

In the last decades, efficient methods and algorithms using radial basis functions (RBFs) have gained popularity in various areas of scientific computing such as multivariate interpolation, approximation theory, meshfree (or meshless) methods, neural networks, computer graphics, computer aided geometric design (CAGD) and machine learning. In particular, the need of having fast algorithms and powerful and flexible software is of great interest mainly in applications, where the amount of data to be interpolated is often very large, say many thousands or even millions of points (see, e.g., [1–5] for an overview).

In the literature, several techniques and alternative approaches have been proposed to have stable and accurate numerical algorithms (see, e.g., [6–10] and references therein), but, except for [11], none allows to deal with a truly great number of data in a relatively small amount of time.

In this paper we focus on the problem of constructing a new fast algorithm for bivariate interpolation of large sets of scattered data. It is based on the *partition of unity method* for constructing a global interpolant by blending radial basis functions as local approximants and using locally supported weight functions. The partition of unity method was firstly suggested in [12,13] in the mid 1990s in the context of meshfree Galerkin methods for the solution of partial differential equations (PDEs), but now it is also an effective method for fast computation in the field of approximation theory (see, for example, [2,14,5]). Similar local approaches involving the modified Shepard's method have already been studied in previous works (see, e.g., [15–21]).

* Corresponding author. Tel.: +39 0116702837.
E-mail addresses: roberto.cavoretto@unito.it (R. Cavoretto), alessandra.derossi@unito.it (A. De Rossi).

0898-1221/$ – see front matter © 2014 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.camwa.2014.01.007

Thus, starting from the results of our previous researches (see [15,22–26]) where efficient searching procedures based either on the partition of a plane domain in strips or on the partition of a sphere in spherical zones are considered, we extend the previous ideas replacing the strip-based partition structure with a cell-based one. The latter leads to the creation of a *cell-based searching procedure*, whose origin comes from the repeated use of a *quicksort* routine with respect to different directions, enabling us to pass from unordered to ordered data structures. In particular, this process turns out to be strictly related to the construction of a partition of the domain Ω in square cells. It consists in generating two orthogonal families of parallel strips, namely a crossed-strip structure, where the original data set is suitably split up into ordered and well-organized data subsets.

Now, exploiting the ordered data structure and the domain partition, the crossed-strip algorithm is efficiently implemented and optimized by connecting the interpolation method with the effective cell-based searching procedure. More precisely, the technique is characterized by the construction of a double structure of crossed strips, called *cell structure*. It partitions the domain Ω in square cells and strictly depends on the dimension of its subdomains, providing a meaningful improvement in the searching procedures of the nearest neighbour points compared to the searching techniques presented in [15,22,24]. The final result is an efficient algorithm for bivariate interpolation of generally scattered data points, whose construction process can briefly be summarized in three stages: (i) partition the domain Ω into a suitable number of cells; (ii) consider an optimized cell-based searching procedure establishing the minimal number of cells to be examined, in order to localize the subset of nodes belonging to each subdomain; (iii) apply the partition of unity method combined with local radial basis functions.

Finally, an analysis of computational complexity shows the high efficiency of this interpolation algorithm, enabling a fast computation of a very large amount of data as shown by several numerical experiments.

The paper is organized as follows. In Section 2 we recall some theoretical preliminaries: at first, we discuss the solvability of the interpolation problem, referring to existence and uniqueness of radial basis function interpolants, we then give a general description of the partition of unity method, which uses radial basis functions as local approximants. In Section 3, we present in detail the cell-based partition algorithm for bivariate interpolation of generally scattered data points, which is efficiently implemented and optimized by using a nearest neighbour searching procedure. Computational complexity and storage requirements of the interpolation algorithm are analysed. In Section 4, we show numerical results concerning efficiency and accuracy of the partition of unity algorithm, while Section 5 contains an application to real data. Finally, Section 6 deals with conclusions and future work.

2. Preliminaries

2.1. Radial basis function interpolation

Let $\mathcal{X}_n = \{\mathbf{x}_i,\ i = 1, 2, \ldots, n\}$ be a set of distinct data points or nodes, arbitrarily distributed in a domain $\Omega \subseteq \mathbb{R}^N,\ N \geq 1$, with an associated set $\mathcal{F}_n = \{f_i,\ i = 1, 2, \ldots, n\}$ of data values or function values, which are obtained by sampling some (unknown) function $f : \Omega \to \mathbb{R}$ at the nodes, i.e., $f_i = f(\mathbf{x}_i),\ i = 1, 2, \ldots, n$. Thus, we can now give a precise formulation of the scattered data interpolation problem.

Problem 2.1. Given the point sets $\mathcal{X}_n$ and $\mathcal{F}_n$, find a (continuous) function $R : \Omega \to \mathbb{R}$ such that

$$R(\mathbf{x}_i) = f_i, \quad i = 1, 2, \ldots, n. \tag{1}$$

Now, using a RBF expansion to solve the scattered data interpolation problem in Ω, the above-mentioned problem can be written as follows.

Definition 2.1. Given the point sets $\mathcal{X}_n$ and $\mathcal{F}_n$, a radial basis function interpolant $R : \Omega \to \mathbb{R}$ assumes the form

$$R(\mathbf{x}) = \sum_{j=1}^{n} c_j \phi(d(\mathbf{x}, \mathbf{x}_j)), \quad \mathbf{x} \in \Omega, \tag{2}$$

where $d(\mathbf{x}, \mathbf{x}_j) = \|\mathbf{x} - \mathbf{x}_j\|_2$ is the Euclidean distance, $\phi : [0, \infty) \to \mathbb{R}$ is called *radial basis function*, and R satisfies the interpolation conditions (1).

Solving the interpolation problem under this assumption leads to a system of linear equations of the form

$$A\mathbf{c} = \mathbf{f},$$

where the entries of the interpolation matrix $A \in \mathbb{R}^{n \times n}$ are given by

$$A_{i,j} = \phi(d(\mathbf{x}_i, \mathbf{x}_j)), \quad i, j = 1, 2, \ldots, n, \tag{3}$$

$\mathbf{c} = [c_1, c_2, \ldots, c_n]^T$, and $\mathbf{f} = [f_1, f_2, \ldots, f_n]^T$. Then, the interpolation problem is well-posed, i.e., a solution to the problem exists and is unique in the interpolation space

$$T_\phi = \text{span}\{\phi(d(\cdot, \mathbf{x}_1)), \ldots, \phi(d(\cdot, \mathbf{x}_n))\}$$

if and only if the matrix A is nonsingular. In fact, it is known that a sufficient condition to have nonsingularity is that the corresponding matrix is positive definite. Thus, if A is a positive definite matrix, then all its eigenvalues are positive and therefore A is nonsingular (see, e.g., [2]).

Definition 2.2. Let $\mathcal{X}_n = \{\mathbf{x}_i,\ i = 1, 2, \ldots, n\}$ be a set of n distinct data points on $\Omega \subseteq \mathbb{R}^N$. A continuous function $\phi : [0, \infty) \to \mathbb{R}$ is called *positive definite* of order n on Ω, if

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j \phi(d(\mathbf{x}_i, \mathbf{x}_j)) \geq 0, \tag{4}$$

for any $\mathbf{c} = [c_1, c_2, \ldots, c_n]^T \in \mathbb{R}^n$. The function ϕ is called *strictly positive definite* of order n if the quadratic form (4) is zero only for $\mathbf{c} \equiv \mathbf{0}$. If ϕ is strictly positive definite for any n, then it is called strictly positive definite.

Therefore, if ϕ is strictly positive definite, the interpolant (2) is unique, since the corresponding interpolation matrix (3) is positive definite and hence nonsingular.

There are many examples of strictly positive definite RBFs (both globally and compactly supported), which can be used to solve the scattered data interpolation problem. The most popular choices for globally supported RBFs are

$$\begin{aligned} &\phi_1(r) = \mathrm{e}^{-\alpha^2 r^2}, && \text{(Gaussian)} \\ &\phi_2(r) = (1 + \gamma^2 r^2)^{m/2}, && \text{(generalized multiquadric)} \end{aligned}$$

where $\alpha, \gamma \in \mathbb{R}^+,\ m \in \mathbb{Z}$, and $r = \|\mathbf{x} - \mathbf{x}_i\|_2$. The Gaussian and the Inverse MultiQuadric (IMQ) (the latter occurs for $m < 0$ in the generalized multiquadric function) are strictly positive definite functions, and this guarantees the existence of a unique solution of the related system of equations. Otherwise, the multiquadric (MQ), i.e. for $m > 0$ in the generalized multiquadric function, is strictly conditionally positive definite function of order m and requires the addition of a polynomial term of order $m - 1$ together with side conditions in order to obtain an invertible interpolation matrix (see, e.g., [5]).

It can be highly advantageous to work with locally supported functions since they might lead to sparse linear systems. Wendland [27] found a class of radial basis functions which are smooth, compactly supported, and strictly positive definite on $\mathbb{R}^N$ for any N. They consist of a product of a truncated power function and a low degree polynomial. For example, one can take

$$\begin{aligned} &\phi_3(r) = (1 - cr)_+^4\,(4cr + 1)\,, && \text{(Wendland's } C^2 \text{ function)} \\ &\phi_4(r) = (1 - cr)_+^6\left(35c^2r^2 + 18cr + 3\right), && \text{(Wendland's } C^4 \text{ function)} \end{aligned}$$

where $c \in \mathbb{R}^+$, and the truncated power function $(r)_+$ is defined as r for $r \geq 0$ and 0 for $r < 0$. In particular, Wendland's functions are nonnegative for $r \in [0, 1/c]$, are zero for $r > 1/c$, and belong to C^2 and C^4, respectively; moreover, they are strictly positive definite in $\mathbb{R}^3$, even though one can also construct strictly positive definite functions (with higher-order smoothness) on $\mathbb{R}^N,\ N > 3$.

2.2. Partition of unity method

The basic idea of the partition of unity method is to start with a partition of the open and bounded domain $\Omega \subseteq \mathbb{R}^N$ into d subdomains Ω_j such that $\Omega \subseteq \bigcup_{j=1}^{d} \Omega_j$ with some mild overlap among the subdomains. At first, we choose a partition of unity, i.e. a family of compactly supported, non-negative, continuous functions W_j with $\mathrm{supp}(W_j) \subseteq \Omega_j$ such that

$$\sum_{j=1}^{d} W_j(\mathbf{x}) = 1. \tag{5}$$

Then, for each subdomain Ω_j we consider a radial basis function R_j as the local approximant and form the global approximant given by

$$\mathcal{I}(\mathbf{x}) = \sum_{j=1}^{d} R_j(\mathbf{x}) W_j(\mathbf{x}), \quad \mathbf{x} \in \Omega. \tag{6}$$

Note that if the local approximants satisfy the interpolation conditions at data point $\mathbf{x}_i$, i.e. $R_j(\mathbf{x}_i) = f(\mathbf{x}_i)$, then the global approximant also interpolates at this node, i.e. $\mathcal{I}(\mathbf{x}_i) = f(\mathbf{x}_i)$, for $i = 1, 2, \ldots, n$.

More precisely, we give the following definition (see [14]).

Definition 2.3. Let $\Omega \subseteq \mathbb{R}^N$ be a bounded set. Let $\{\Omega_j\}_{j=1}^{d}$ be an open and bounded covering of Ω. This means that all Ω_j are open and bounded and that Ω is contained in their union. Set $\delta_j = \mathrm{diam}(\Omega_j) = \sup_{\mathbf{x},\,\mathbf{y}\in\Omega_j} \|\mathbf{x} - \mathbf{y}\|_2$. We call a family of

nonnegative functions $\{W_j\}_{j=1}^d$ with $W_j \in C^k(\mathbb{R}^N)$ a k-stable partition of unity with respect to the covering $\{\Omega_j\}_{j=1}^d$ if

(1) $\text{supp}(W_j) \subseteq \Omega_j$;
(2) $\sum_{j=1}^d W_j(\mathbf{x}) \equiv 1$ on Ω;
(3) for every $\beta \in \mathbb{N}_0^N$ with $|\beta| \leq k$ there exists a constant $C_\beta > 0$ such that

$$\|D^\beta W_j\|_{L_\infty(\Omega_j)} \leq C_\beta / \delta_j^{|\beta|},$$

for all $1 \leq j \leq d$.

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

$$q_{\mathcal{X}_n} = \frac{1}{2} \min_{i \neq j} d(\mathbf{x}_i, \mathbf{x}_j), \tag{7}$$

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

$$h_{\mathcal{X}_n,\Omega} = \sup_{\mathbf{x} \in \Omega} \min_{\mathbf{x}_j \in \mathcal{X}_n} d(\mathbf{x}, \mathbf{x}_j). \tag{8}$$

In agreement with the statements in [14], we require some additional regularity assumptions on the *covering* $\{\Omega_j\}_{j=1}^d$.

Definition 2.4. Suppose that $\Omega \subseteq \mathbb{R}^N$ is bounded and $\mathcal{X}_n = \{\mathbf{x}_i,\ i = 1, 2, \ldots, n\} \subseteq \Omega$ are given. An open and bounded covering $\{\Omega_j\}_{j=1}^d$ is called regular for $(\Omega, \mathcal{X}_n)$ if the following properties are satisfied:

(a) for each $\mathbf{x} \in \Omega$, the number of subdomains Ω_j with $\mathbf{x} \in \Omega_j$ is bounded by a global constant K;
(b) each subdomain Ω_j satisfies an interior cone condition;
(c) the local fill distances $h_{\mathcal{X}_j,\Omega_j}$ are uniformly bounded by the global fill distance $h_{\mathcal{X}_n,\Omega}$, where $\mathcal{X}_j = \mathcal{X}_n \cap \Omega_j$.

Remark 2.1. The first property (a) is required to ensure that the sum in (6) is actually a sum over at most K summands. Since K is independent of n, unlike d, which should be proportional to n, this is essential to avoid losing convergence orders. Moreover, it is crucial for an efficient evaluation of the global interpolant that only a constant number of local approximants has to be evaluated. Then, it should be possible to locate those K indices in constant time. The second and third properties (b) and (c) are important for employing the estimates on radial basis function interpolants (see [5]).

Moreover, we are able to formulate the following theorem, which yields the polynomial precision and controls the growth of error estimates (see, e.g., [5]). Here, we denote by $\pi_s^N := \pi_s(\mathbb{R}^N)$ the set of polynomials of degree at most s.

Theorem 2.1. *Suppose that $\Omega \subseteq \mathbb{R}^N$ is compact and satisfies an interior cone condition with angle $\theta \in (0, \pi/2)$ and radius $r > 0$. Let $s \in \mathbb{N}$ be fixed and there exist constants $h_0, C_1, C_2 > 0$ depending only on N, θ, r such that $h_{\mathcal{X}_n,\Omega} \leq h_0$. Then, for all $\mathcal{X}_n = \{\mathbf{x}_i,\ i = 1, 2, \ldots, n\} \subseteq \Omega$ and all $\mathbf{x} \in \Omega$, there are functions $u_k : \Omega \to \mathbb{R},\ k = 1, 2, \ldots, n$, such that*

(1) $\sum_{k=1}^n u_k(\mathbf{x}) p(\mathbf{x}_k) = p(\mathbf{x})$, *for all $p \in \pi_s(\mathbb{R}^N)$;*
(2) $\sum_{k=1}^n |u_k(\mathbf{x})| \leq C_1$;
(3) $u_j(\mathbf{x}) = 0$ *provided that* $\|\mathbf{x} - \mathbf{x}_j\|_2 > C_2 h_{\mathcal{X}_n,\Omega}$.

Therefore, after defining the space $C_\nu^k(\mathbb{R}^N)$ of all functions $f \in C^k$ whose derivatives of order $|\beta| = k$ satisfy $D^\beta f(\mathbf{x}) = \mathcal{O}(\|\mathbf{x}\|_2^\nu)$ for $\|\mathbf{x}\|_2 \to 0$, we consider the following convergence result. It is presented here for strictly positive definite functions even if it holds more in general for strictly conditionally positive definite functions (see, e.g., [2,5] and references therein).

Theorem 2.2. *Let $\Omega \subseteq \mathbb{R}^N$ be open and bounded and suppose that $\mathcal{X}_n = \{\mathbf{x}_i,\ i = 1, 2, \ldots, n\} \subseteq \Omega$. Let $\phi \in C_\nu^k(\mathbb{R}^N)$ be a strictly positive definite function. Let $\{\Omega_j\}_{j=1}^d$ be a regular covering for $(\Omega, \mathcal{X}_n)$ and let $\{W_j\}_{j=1}^d$ be k-stable for $\{\Omega_j\}_{j=1}^d$. Then the error between $f \in \mathcal{N}_\phi(\Omega)$, where $\mathcal{N}_\phi$ is the native space of ϕ, and its partition of unity interpolant (6) can be bounded by*

$$|D^\beta f(\mathbf{x}) - D^\beta \mathcal{I}(\mathbf{x})| \leq C h_{\mathcal{X}_n,\Omega}^{(k+\nu)/2-|\beta|} |f|_{\mathcal{N}_\phi(\Omega)},$$

for all $\mathbf{x} \in \Omega$ and all $|\beta| \leq k/2$.

Note that the partition of unity preserves the local approximation order for the global fit. Hence, we can efficiently compute a large RBF interpolation problem by solving small RBF ones (in parallel as well) and then combine them together with the global partition of unity $\{W_j\}_{j=1}^d$. This approach enables us to decompose a large problem into many small problems, and at the same time ensures that the accuracy obtained for the local fits is carried over to the global one. In particular, the partition of unity method can be thought as a Shepard's method with higher-order data, since local approximations R_j are used instead of data values f_j (see [15,28]).

Remark 2.2. Among several weight functions $\bar{W}_j(\mathbf{x})$ in (6), a possible choice is given by Shepard's weight

$$W_j(\mathbf{x}) = \frac{\bar{W}_j(\mathbf{x})}{\sum_{k=1}^{d} \bar{W}_k(\mathbf{x})}, \quad j = 1, 2, \ldots, d, \tag{9}$$

where $\bar{W}_j$ is the inverse of the Euclidean norm $\|\cdot\|_2$. It constitutes a partition of unity as in (5).

Remark 2.3. Apart from possible problems of storage and computer running time, when a large number of interpolation nodes is used, RBF systems may suffer from ill-conditioning. In general, the condition number is directly linked to the order of the basis functions and to the density of the interpolation nodes. Indeed, the ill-conditioning grows primarily due to the decrease in the separation distance $q_{\mathcal{X}_n}$, and not only necessarily due to the increase in the number n of data points. Moreover, since the local separation distance $q_{\mathcal{X}_j}$ is of the same size (or smallness) as the global separation distance $q_{\mathcal{X}_n}$, the partition of unity method seems to be stable as the global one.

On the other hand, if one keeps the number of nodes (or at least the separation distance) fixed and instead considers flatter basis functions by a suitable choice of the shape parameter, then the condition number of the interpolation matrix A suffers in almost the same manner. Of course, a more peaked basis function can be used to improve the condition number of A, but the accuracy of the fit gets worse.

Anyway, in agreement with the *trade-off* (or *uncertainty*) *principle* [29] we remark that the order of the basis functions should be chosen with great care, because using *standard* bases one cannot have high accuracy and stability at the same time [30]. This order should be low enough when the data density is quite high, because any excessive order has negative effects on stability. Furthermore, for low density interpolation data points, one can use high-order basis functions such as Gaussians and generalized inverse multiquadrics (that are infinitely smooth). For high density interpolation data points, one can use low-order basis functions such as compactly supported Wendland's functions (that have limited or arbitrarily low smoothness) to avoid numerical problems (see [2]). More recently, however, a number of approximation techniques have been proposed to have a stable computation with flat and infinitely smooth radial basis functions (see, e.g., [7,9] and references therein).

3. Cell-based partition algorithm

In this section we propose a new fast algorithm for bivariate interpolation of large scattered data sets lying on the domain $\Omega = [0, 1] \times [0, 1] \subset \mathbb{R}^2$. This algorithm is based on the partition of unity method for constructing a global interpolant by blending radial basis functions as local approximants and using locally supported weight functions. It is efficiently implemented and optimized by connecting the method with an effective cell-based searching procedure. More precisely, the approach is characterized by the construction of a double structure of crossed strips, called *cell structure*. It partitions the domain Ω into square cells and strictly depends on the dimension of its subdomains, providing a meaningful improvement in the searching procedures of the nearest neighbour points compared to the searching techniques presented in [15,22,24].

The process we are considering can briefly be described as follows:

(i) partition the domain Ω into a suitable number of cells;
(ii) consider an optimized cell-based searching procedure establishing the minimal number of cells to be examined, in order to localize the set of nodes for each subdomain;
(iii) apply the partition of unity method which uses radial basis functions as nodal functions.

These three stages correspond to data partition, localization and evaluation phases, respectively. Note that only one double structure of crossed strips (cell structure) is used for each of the three phases.

3.1. Cell-based searching procedure

The basic idea in the construction of this searching procedure comes from the repeated use of a *quicksort* routine with respect to different directions (essentially, along the y-axis and the x-axis), enabling us to pass from unordered to ordered data structures. This process is strictly related to the construction of a partition of the domain Ω (i.e., unit square) into square cells and consists in generating two orthogonal families of parallel strips, namely a crossed strip structure (see Fig. 1), where the original data set is suitably split up into ordered and well-organized data subsets. More precisely, in order to obtain the cell-based partition structure and then the resulting searching procedure, we may act as follows: at first, we organize all the data by means of a *quicksort*$_y$ *procedure* applied along the y-axis (the subscript denotes the sorting direction), then we consider a first family of q strips, parallel to the x-axis and order the points of each strip by using a *quicksort*$_x$ *procedure*, finally we create a second family of q strips, parallel to the y-axis, which orthogonally intersect the first strip family, thus producing a partition of Ω in square cells (see Fig. 2). Note that from now on, to define a specific cell k, we consider a double index notation using square brackets, i.e. $k = [v, w]$.

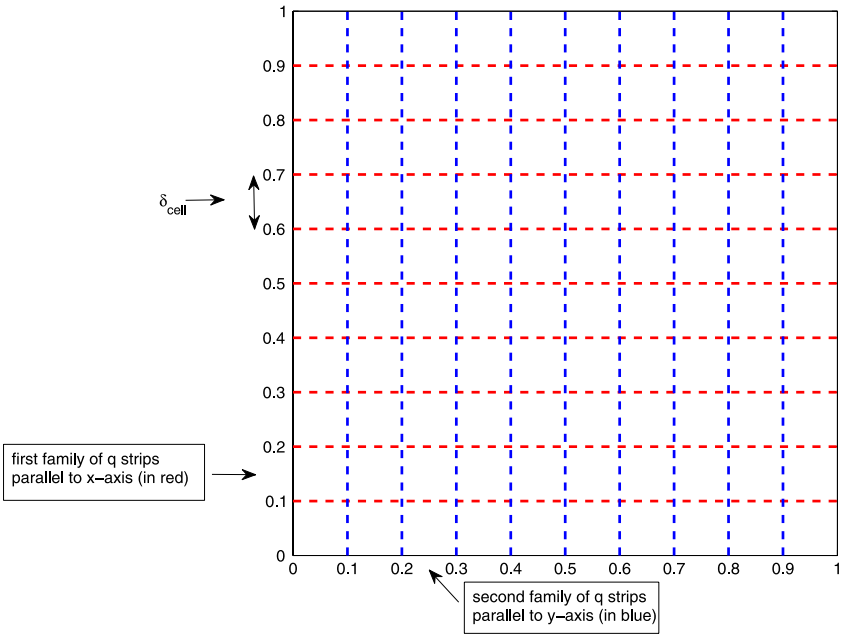


Fig. 1. Example of crossed-strip partition.

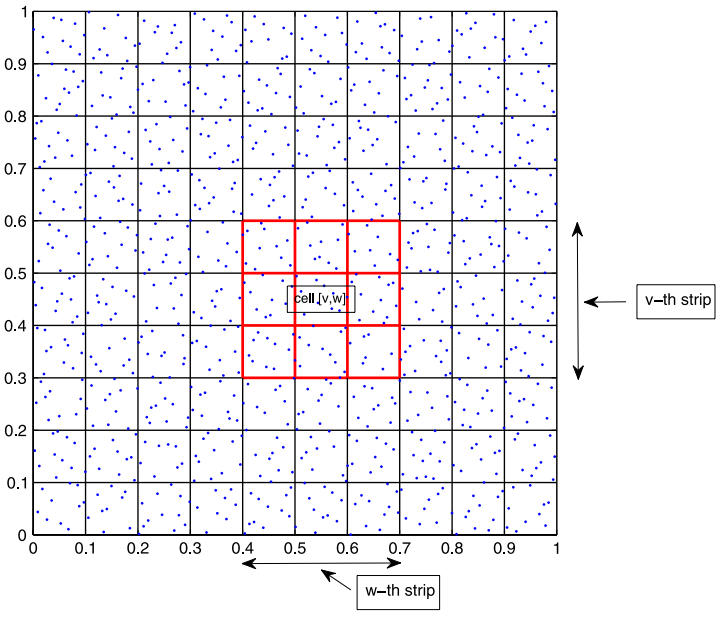


Fig. 2. Example of cell-based structure with a set of scattered data points.

The aim is to construct an efficient searching procedure to be used in the localization of points, exploiting the data structure and the domain partition we have just considered above. An effective way to obtain an efficient searching technique is to connect the interpolation or approximation method (in this case, the partition of unity method, even if such choice is not restrictive) with the cell-based partition structure. This result is obtained assuming that the cell width/side δ_{cell} is equal to the subdomain radius δ_{subdom}, i.e. $\delta_{cell} \equiv \delta_{subdom}$. Though this choice might seem to be trivial, in practice such an imposition means that the search of the nearby points, which in general is an essential aspect of local methods as the partition of unity method, is limited at most to nine cells: the cell on which the considered point lies, and the eight neighbouring cells (see Figs. 1–2). In fact, the combination between cell and subdomain sizes provides an *optimal* choice, since it allows us to search the closest points only considering a very small number of them (that is only those points belonging to one of the nine cells) and *a priori* ignoring all the other points of Ω. Obviously, then, for all those points belonging to the first and last cells, i.e. the ones close to the boundary of Ω, a reduction of the total number of cells to be examined will be required, but

this does not produce any problem on effectiveness of the cell-based searching procedure. Further details on this searching procedure are contained in Section 3.2, where we give a detailed description of the proposed algorithm.

3.2. Crossed-strip algorithm

We now present the partition of unity algorithm for bivariate interpolation of generally scattered data points.

INPUT: n, number of data; $\mathcal{X}_n = \{(x_i, y_i),\ i = 1, 2, \ldots, n\}$, set of data points; $\mathcal{F}_n = \{f_i,\ i = 1, 2, \ldots, n\}$, set of data values; d, number of subdomains; $\mathcal{C}_d = \{(\bar{x}_i, \bar{y}_i),\ i = 1, 2, \ldots, d\}$, set of subdomain points (centres); s, number of evaluation points; $\mathcal{E}_s = \{(\tilde{x}_i, \tilde{y}_i),\ i = 1, 2, \ldots, s\}$, set of evaluation points.

OUTPUT: $\mathcal{A}_s = \{\mathcal{I}(\tilde{x}_i, \tilde{y}_i),\ i = 1, 2, \ldots, s\}$, set of approximated values.

`Stage 1.` The set $\mathcal{X}_n$ of nodes and the set $\mathcal{E}_s$ of evaluation points are ordered with respect to a common direction (e.g. the y-axis), by applying a *quicksort$_y$ procedure*.

`Stage 2.` For each subdomain point $(\bar{x}_i, \bar{y}_i),\ i = 1, 2, \ldots, d$, a local circular subdomain is constructed, whose half-size (the radius) depends on the subdomain number d, that is

$$\delta_{subdom} = \sqrt{\frac{2}{d}}. \tag{10}$$

This value is suitably chosen, supposing to have a nearly uniform node distribution and assuming that the ratio $n/d \approx 4$.

`Stage 3.` A double structure of crossed strips is constructed as follows:

(i) a first family of q strips, parallel to the x-axis, is considered taking

$$q = \left\lceil \frac{1}{\delta_{subdom}} \right\rceil, \tag{11}$$

and a *quicksort$_x$ procedure* is applied to order the nodes belonging to each strip;

(ii) a second family of q strips, parallel to the y-axis, is considered.

Note that each of the two strip structures are ordered and numbered from 1 to q; moreover, the choice in (11) follows directly from the side length of the domain Ω (unit square), that here is 1, and the subdomain radius δ_{subdom}.

`Stage 4.` The domain (unit square) is partitioned by a cell-based structure consisting of q^2 square cells, whose length of the sides is given by $\delta_{cell} \equiv \delta_{subdom}$. Then, the following structure is considered:

- the sets $\mathcal{X}_n$, $\mathcal{C}_d$ and $\mathcal{E}_s$ are partitioned by the cell structure into q^2 subsets $\mathcal{X}_{n_k}$, $\mathcal{C}_{d_k}$ and $\mathcal{E}_{s_k}$, $k = 1, 2, \ldots, q^2$,

where n_k, d_k and s_k are the number of points in the k-th cell.

This stage can be summarized in Algorithm 1 as follows.

Algorithm 1 Cell-based partition structure

```
for each cell k = [v, w], v, w = 1, 2, ..., q do
    partition and count the number of points
            n_k = n_{v,w} (nodes)
            d_k = d_{v,w} (subdomain points)
            s_k = s_{v,w} (evaluation points);
    return (n_k; X_{n_k}) ∧ (d_k; C_{d_k}) ∧ (s_k; E_{s_k})
end for
```

`Stage 5.` In order to identify the cells to be examined in the searching procedure, we adopt the following rule which is composed of three steps:

(1) the width δ_{cell} of cells is chosen equal to the subdomain radius δ_{subdom}, i.e. $\delta_{cell} \equiv \delta_{subdom}$, and the ratio between these quantities is denoted by $i^* = \delta_{subdom}/\delta_{cell}$;

(2) the value i^* provides the number j^* of cells to be examined for each point by the rule $j^* = (2i^* + 1)^2$, which obviously here gives $j^* = 9$. In practice, this means that the search of the nearby points is limited at most to nine cells: the cell on which the considered point lies, and the eight neighbouring cells;

(3) for each cell $k = [v, w],\ v, w = 1, 2, \ldots, q$, a cell-based searching procedure is considered, examining the points from the cell $[v - i^*, w - i^*]$ to the cell $[v + i^*, w + i^*]$. For the points of the first and last cells (those close to the boundary of the unit square Ω), we reduce the total number of cells to be examined, setting $v - i^* = 1$ and/or $w - i^* = 1$ (when $v - i^* < 1$ and/or $w - i^* < 1$) and $v + i^* = q$ and/or $w + i^* = q$ (when $v + i^* > q$ and/or $w + i^* > q$).

Then, after defining which and how many cells are to be examined, the cell-based searching procedure (see Algorithm 2) is applied:

- for each subdomain point of $\mathcal{C}_{d_k},\ k = 1, 2, \ldots, q^2$, to determine all nodes belonging to a subdomain. The number of nodes of the subdomain centered at $(\bar{x}_i, \bar{y}_i)$ is counted and stored in $m_i,\ i = 1, 2, \ldots, d$;
- for each evaluation point of $\mathcal{E}_{s_k},\ k = 1, 2, \ldots, q^2$, in order to find all those belonging to a subdomain of centre $(\bar{x}_i, \bar{y}_i)$ and radius δ_{subdom}. The number of subdomains containing the ith evaluation point is counted and stored in $r_i,\ i = 1, 2, \ldots, s$.

Algorithm 2 Cell-based searching procedure

for $w = 1, 2, \ldots, q$ **do**
  for $v = 1, 2, \ldots, q$ **do**
    set $[first_x, first_y] = [v - i^*, w - i^*]$
      $[last_x, last_y] = [v + i^*, w + i^*]$
    if $first_x < 1$ and/or $first_y < 1$ **then**
      set $first_x = 1$ and/or $first_y = 1$
    end if
    if $last_x > q$ and/or $last_y > q$ **then**
      set $last_x = q$ and/or $last_y = q$
    end if
    for $h = subdom\_bp_{v,w}, \ldots, subdom\_ep_{v,w}$ **do**
      set $m_h = 0$
      for $j = first_y, \ldots, last_y$ **do**
        for $i = first_x, \ldots, last_x$ **do**
          for $k = bp_{i,j}, \ldots, ep_{i,j}$ **do**
            if $(x_k, y_k) \in I_h((\bar{x}, \bar{y}); \delta_{subdom})$ **then**
              set $m_h = m_h + 1$
                $STORE_{h,m_h}(x_k, y_k, f_k)$
            end if
          end for
        end for
      end for
      return $(x, y) \in I_h((\bar{x}, \bar{y}); \delta_{subdom})$
    end for
    for $h = eval\_bp_{v,w}, \ldots, eval\_ep_{v,w}$ **do**
      set $r_h = 0$
      for $j = first_y, \ldots, last_y$ **do**
        for $i = first_x, \ldots, last_x$ **do**
          for $k = subdom\_bp_{i,j} \ldots, subdom\_ep_{i,j}$ **do**
            if $(\tilde{x}_k, \tilde{y}_k) \in I_h((\bar{x}, \bar{y}); \delta_{subdom})$ **then**
              set $r_h = r_h + 1$
                $STORE_{h,r_h}(\tilde{x}_k, \tilde{y}_k)$
            end if
          end for
        end for
      end for
      return $(\tilde{x}, \tilde{y}) \in I_h((\bar{x}, \bar{y}); \delta_{subdom})$
    end for
  end for
end for

`Stage 6.` A local interpolant $R_j,\ j = 1, 2, \ldots, d$, is found for each subdomain point.

`Stage 7.` A local approximant $R_j(x, y)$ and a weight function $W_j(x, y),\ j = 1, 2, \ldots, d$, is found for each evaluation point.

`Stage 8.` Applying the global fit (6), the surface can be approximated at any evaluation point $(\tilde{x}, \tilde{y}) \in \mathcal{E}_s$.

Remark 3.1. In the algorithm the local approximants are computed by using either globally supported RBFs such as Gaussians and IMQs (that are compactly supported on the local subdomains), or compactly supported RBFs such as Wendland's functions. Moreover, the d local subdomains are given by circles centered at equally spaced points in the unit square. Finally, we point out that this approach turns out to be very flexible, since different choices of local approximants (both globally and locally supported) are allowed.

Remark 3.2. The partition of unity algorithm is easily parallelizable. In fact, the cell special structure in which the domain Ω is partitioned and in which the points are organized (see `Stage 4`), as well as the favourable choice of taking $\delta_{cell} \equiv \delta_{subdom}$ (see `Stage 4` and `Stage 5`), makes this algorithm particularly suitable for parallel computation.

Remark 3.3. An interesting observation deserving to be taken into account concerns the type of partition of unity to be used. In practice, there are many possibilities of constructing a partition of unity and a suitable choice is essential for obtaining good results. Such a choice usually turns out to be a difficult task as it basically depends on the data distribution in Ω. In particular, in this work we consider the case of scattered data, but supposing these points possess any kind of regularity. For this reason, we are able to suitably connect the subdomain size, i.e. the interpolation method, with the node number (see `Stage 2`), making the algorithm applicable in practice. Obviously, it is also possible to study different strategies of partition of unity, e.g. reducing the subdomain size and increasing the number of subdomains to cover all Ω; nevertheless, in general, this change leads not only to better condition numbers but also to an accuracy degradation. However, the way of partitioning the domain is a further variable (that is added to those already outlined in Remark 2.3), influencing significantly the approximation results both regarding accuracy and stability. Hence, this means that the choice of the type of partition of unity should be carried out considering the interplay of all the parameters.

Remark 3.4. Since this algorithm is based on a meshfree method, in general it might be extended with suitable adaptations also to other types of domains like polygons (e.g. triangles, hexagons, L-shaped domains, etc.). While the mathematical background of our method is general and does not require any change, the algorithm is constructed for a square domain so that it needs only little adjustments on the cell-based searching procedure for applying it to convex domains. However, since a 2D polygon can be inscribed in a square domain, we have to add a control to check whether a cell contains at least a point or it is empty; in the latter case we exclude it from the searching process. This addition allows us to find all points belonging to each subdomain, as described in this section.

3.3. Complexity analysis

Since the partition of unity algorithm is characterized by the construction of local RBF interpolants, we consider the local data sets $\mathcal{X}_j = \mathcal{X}_n \cap \Omega_j,\ j = 1, 2, \ldots, d$, instead of the global data set $\mathcal{X}_n$. Thus, the complexity of this algorithm is influenced by the following assumptions:

(i) a data structure is considered for each set of points (i.e., nodes, subdomain and evaluation points) such that they can be efficiently identified in each subdomain Ω_j;
(ii) a local approach has to be such that each subdomain contains only a relatively small number of nodes, i.e. each node has to be contained only in some subdomains and needs to be efficiently found.

We remark that these conditions lead to the requirement that the number d of subdomains is proportional to the number n of nodes, say $n/d \approx 4$ (see `Stage 2`). Thus, assuming to have a quasi-uniform node set $\mathcal{X}_n$ and since the covering $\{\Omega_j\}_{j=1}^{d}$ is local and regular, the size of each subdomain is proportional to $h_{\mathcal{X}_n,\Omega}$.

The partition of unity algorithm involves 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. Specifically, we have a distribution phase consisting of building the data structure, in which the computational cost has order: $\mathcal{O}(n \log n)$ for the sorting of all n nodes and $\mathcal{O}(s \log s)$ for the sorting of all s evaluation points in `Stage 1`. Moreover, in order to compute the local RBF interpolants, we have to solve d linear systems of small dimensions by Gaussian elimination, thus requiring a computational cost of order $\mathcal{O}(m_i^3),\ i = 1, 2, \ldots, d$, for each subdomain, where m_i is the number of nodes in the ith subdomain (see `Stage 6`). Moreover, in `Stage 5`, 7 and 8 we also incur a cost of $r_k \cdot \mathcal{O}(m_i),\ i = 1, 2, \ldots, d,\ k = 1, 2, \ldots, s$, for the k-th evaluation point of $\mathcal{E}_s$. Finally, the algorithm requires $3n$, $3d$ and $3s$ storage requirements for the data, and $m_i,\ i = 1, 2, \ldots, d$, locations for the coefficients of each local RBF interpolant.

4. Numerical experiments

In this section we not only present some tests to mainly verify performance and effectiveness of the cell-based partition algorithm on scattered data sets, but also provide numerical results on accuracy of the interpolation method. The code is implemented in C/C++ language, while numerical results are carried out on a Intel Core 2 Duo Computer (2.1 GHz). In the experiments we consider a node distribution containing $n = (2^k + 1)^2,\ k = 6, 7, 8, 9, 10$, uniformly random Halton nodes generated by using the program given in [31]. The partition of unity algorithm is run considering $d = 4^k,\ k = 5, 6, 7, 8, 9$, subdomain points and $s = 33 \times 33$ evaluation (or grid) points, which are contained in the unit square $\Omega = [0, 1] \times [0, 1]$. Here, for the global interpolant (6) we use Shepard's weight (9).

Now, referring to the separation distance $q_{\mathcal{X}_n}$ in (7) and the fill distance $h_{\mathcal{X}_n,\Omega}$ in (8), in Table 1 we report the value of $q_{\mathcal{X}_n}$ and $h_{\mathcal{X}_n,\Omega}$ for Halton node sets used in the numerical experiments.

Table 1
Separation distance $q_{\mathcal{X}_n}$ and fill distance $h_{\mathcal{X}_n,\Omega}$ for Halton data points by varying n.

n	4225	16 641	66 049	263 169
$q_{\mathcal{X}_n}$	2.1993E−3	5.4709E−4	2.1435E−4	1.1281E−4
$h_{\mathcal{X}_n,\Omega}$	2.1946E−2	1.0342E−2	4.4937E−3	2.6249E−3

Table 2
Information on the cell-based partition algorithm.

n	4225	16 641	66 049	263 169	1 050 625
δ_{subdom}	4.4194E−2	2.2097E−2	1.1049E−2	5.5243E−3	2.7621E−3
q^2	23^2	46^2	91^2	182^2	363^2

Table 3
CPU times (in seconds) obtained by running the cell-based partition algorithm (t_{cell}) and the strip-based partition algorithm (t_{strip}).

n	d	t_{cell}	t_{strip}
4 225	1 024	**0.3**	0.4
16 641	4 096	**0.8**	1.3
66 049	16 384	**2.6**	6.5
263 169	65 536	**10.2**	41.2
1 050 625	262 144	**41.3**	289.5

The performance of the interpolation algorithm is verified taking the data values by three test functions, namely Franke's function f_1, Nielson's function f_2 and test function f_3 (see, e.g., [32,33])

$$\begin{aligned} f_1(x,y) &= \frac{3}{4}\exp\left[-\frac{(9x-2)^2+(9y-2)^2}{4}\right]+\frac{3}{4}\exp\left[-\frac{(9x+1)^2}{49}-\frac{9y+1}{10}\right] \\ &\quad+\frac{1}{2}\exp\left[-\frac{(9x-7)^2+(9y-3)^2}{4}\right]-\frac{1}{5}\exp\left[-(9x-4)^2-(9y-7)^2\right], \\ f_2(x,y) &= \frac{1}{2}y\cos^4\left[4\left(x^2+y-1\right)\right], \\ f_3(x,y) &= 2\cos(10x)\sin(10y)+\sin(10xy). \end{aligned}$$

Some information about the execution of the interpolation algorithm reported in Table 2 gives the length of subdomain radius δ_{subdom} and the cell number q^2.

Moreover, since we are concerned to point out the effectiveness of the proposed algorithm, in Table 3 we compare CPU times (in seconds) obtained by running the cell-based partition algorithm as described in Section 3, and the strip-based partition algorithm proposed in [25]. This comparison emphasizes the high efficiency of the algorithm: in fact, the use of a cell-based structure to partition the domain Ω gives a considerable saving of time above all when the number of interpolated nodes becomes larger and larger. This is also confirmed by execution time ratios between the two algorithms in Fig. 3. Furthermore, we remark that the strip algorithm in [15] has also been compared with Renka's standard procedure in [19,20], turning out to be more efficient than Renka's algorithm.

Then, in order to test the accuracy of the local algorithm, in Tables 4–6 we report the Root Mean Square Errors (RMSEs), i.e.

$$\mathrm{RMSE} = \sqrt{\frac{1}{s}\sum_{i=1}^{s}|f(\mathbf{x}_i)-\mathcal{I}(\mathbf{x}_i)|^2},$$

which are computed for each of the considered test functions. The computation of errors is achieved by considering both globally and locally supported RBFs for suitable values of the shape parameters, i.e., $\alpha=\gamma=7$ for ϕ_1 and ϕ_2 (with $m=-1$), and $c=1$ for ϕ_3 and ϕ_4. We note that the local scheme is accurate, especially when the amount of data points grows, even if we do not consider the optimal values for the parameters, namely those values for which we get the best possible results. However, these choices give a good compromise among accuracy, efficiency and stability.

Moreover, since the aim of experiments is also to examine how errors change as the interpolation nodes grow, we experimentally estimate convergence orders. In fact, in Table 6 we also show the empirical order of convergence of RMSEs through the formula

$$p_k = \frac{\ln(e_{k-1}/e_k)}{\ln(h_{k-1}/h_k)}, \quad k=2,3,\ldots,$$

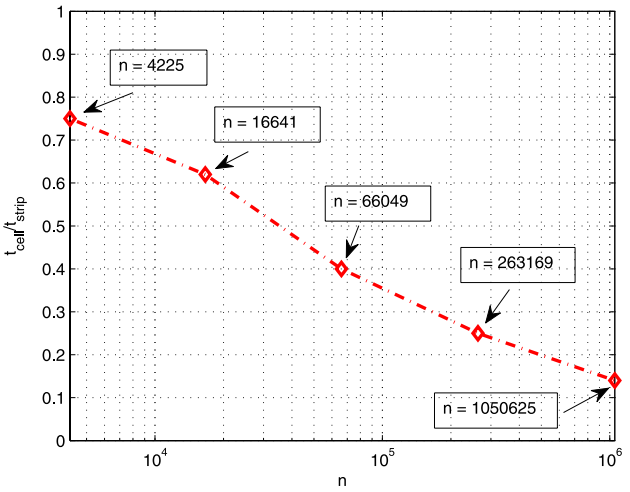


Fig. 3. CPU time ratios t_{cell}/t_{strip} by varying n.

Table 4
RMSEs obtained by RBFs with $\alpha = 7$, $\gamma = 7$ and $c = 1$ for f_1.

n	4225	16 641	66 049
ϕ_1	2.9431E−4	2.7299E−5	1.4879E−6
ϕ_2	1.6165E−4	2.2059E−5	6.3355E−7
ϕ_3	2.2145E−4	5.3127E−5	9.3027E−6
ϕ_4	8.3641E−5	1.5106E−5	5.2541E−7

Table 5
RMSEs obtained by RBFs with $\alpha = 7$, $\gamma = 7$ and $c = 1$ for f_2.

n	4225	16 641	66 049
ϕ_1	1.0113E−4	6.2180E−5	1.0435E−5
ϕ_2	9.2513E−5	5.5783E−5	9.6403E−6
ϕ_3	3.1579E−4	1.2211E−4	3.0063E−5
ϕ_4	2.2972E−4	7.6501E−5	1.2072E−5

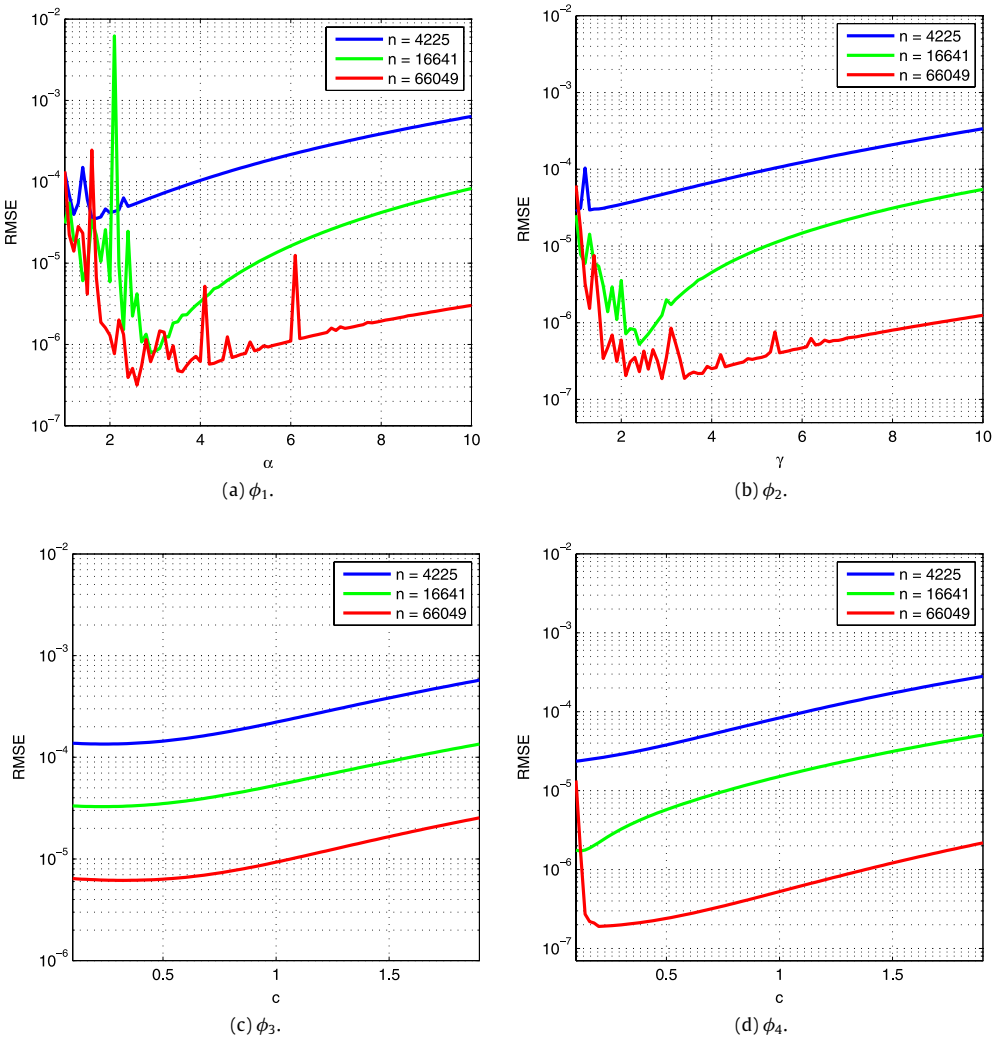
Table 6
RMSEs and convergence orders obtained by RBFs with $\alpha = 7$, $\gamma = 7$ and $c = 1$ for f_3.

n	ϕ_1		ϕ_2		ϕ_3		ϕ_4	
	RMSE	p	RMSE	p	RMSE	p	RMSE	p
4225	1.8821E−4	–	5.5864E−4	–	1.9615E−3	–	3.5543E−4	–
16 641	3.0276E−5	2.4286	6.1985E−5	2.9222	4.8960E−4	1.8447	8.7426E−5	1.8642
66 049	2.6106E−6	2.9402	7.8239E−6	2.4830	1.1496E−4	1.7384	1.4162E−5	2.1837
263 169	2.5747E−7	4.3086	2.6320E−7	6.3092	2.6171E−5	2.7527	7.2686E−7	5.5234

where e_k is the k-th RMSE, whereas h_k denotes the related fill distance. Specifically, this points out the good accuracy of the method primarily due to the use of the radial basis functions and then the high convergence order. In general, the latter does not have usually a uniform behaviour, but the explanation of this phenomenon can be found in the really scattered nature of the data sets considered in numerical tests.

Then, we analyse the behaviour of the RMSEs by varying the shape parameter for each of the considered RBFs. As an example, in Fig. 4 we plot the behaviour of the RMSEs for f_1. These graphs (and other ones that we omit for brevity) point out that, if an optimal search of the shape parameters was performed, in some cases the results of accuracy reported in this section could be improved by one or even two orders of magnitude. Note that each evaluation is carried out by choosing equispaced values of the shape parameter in the intervals $[1, 10]$ for α and γ, and $[0.1, 1.9]$ for c.

By analysing numerical tests and the related pictures, we observe that Wendland's ϕ_3 and ϕ_4 functions, which have compact support, reveal a larger stability than classical radial basis functions ϕ_1 and ϕ_2 (not compactly supported), as well as a good accuracy. These graphs give an idea on the stability and enable us to choose "sure" values for the shape parameters. These observations reflect the expected results, as reported in Remark 2.3, and suggest to use basis functions with a moderate order of smoothness, thus avoiding the ill-conditioning problems that occur when the amount of data increases or the separation distance decreases assuming values roughly smaller than 10^{-3}.



(a) ϕ_1. (b) ϕ_2.

(c) ϕ_3. (d) ϕ_4.

Fig. 4. RMSEs obtained by varying the shape parameters for f_1.

5. Application to Gattinara data

In this section we consider an application to Earth's topography, which consists of interpolating with the cell-based partition algorithm a set of real scattered data, called *Gattinara data*, belonging to the homonymous geographic area. Gattinara is a municipality in the Province of Vercelli in the Italian region Piedmont, located about 80 km North-East of Turin and about 35 km North of Vercelli.

Now, in this specific case we have 10 671 Gattinara data, whose 3D representation is shown in Fig. 5, and among them we randomly select $n = 10\,600$ nodes for the interpolation process, only reserving the remaining $s = 71$ (evaluation) points for the cross-validation (see Fig. 6). The latter technique is commonly used in applications to assess goodness of approximation results and, accordingly, performance of the partition of unity algorithm, comparing the predicted values with the original ones. In order to obtain reliable and numerically significant results on the error, it is more appropriate to use relative (or normalized) errors, such as the Relative Root Mean Square Errors (RRMSEs), whose formula is given by

$$\mathrm{RRMSE} = \sqrt{\frac{1}{s}\sum_{i=1}^{s}\frac{|f(\mathbf{x}_i) - \mathcal{I}(\mathbf{x}_i)|^2}{|f(\mathbf{x}_i)|^2}}.$$

Then, since in Section 4 numerical results on test functions have shown that compactly supported functions turn out to be more stable than globally supported RBFs, preserving, at the same time, a good level of accuracy, in what follows we focus only on Wendland's functions ϕ_3 and ϕ_4. In fact, in Table 7 we report the RRMSEs obtained by varying the shape parameter c. These results point out that the proposed approach and the related cell-based partition algorithm, which interpolates this data set in about 0.5 s, turns to be effective also in real life applications.

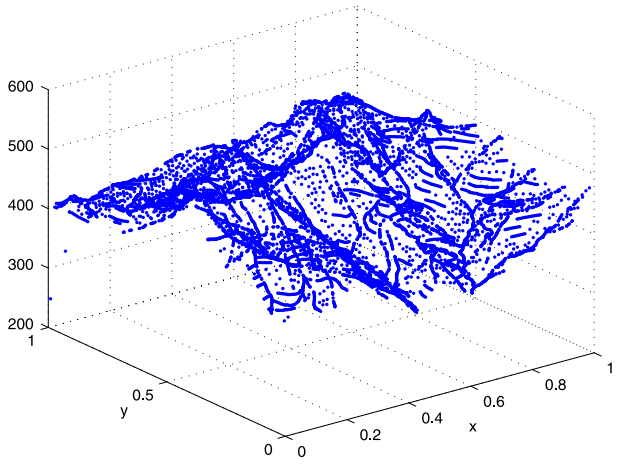


Fig. 5. A 3D view of $n = 10\,600$ Gattinara data (dot, in blue).

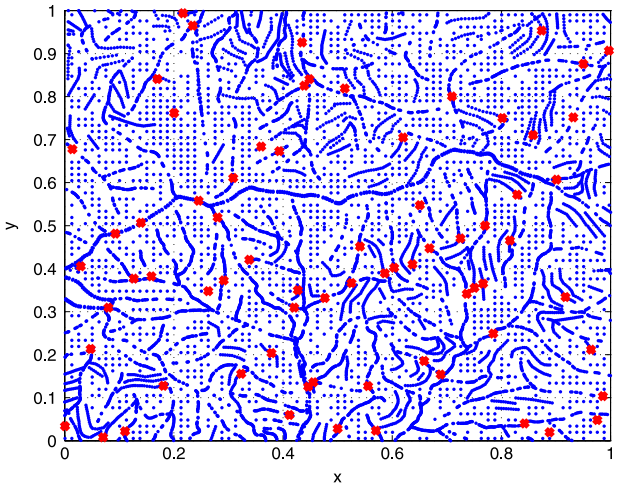


Fig. 6. A 2D view of a Gattinara data set with $n = 10\,600$ interpolation nodes (dot, in blue) and $s = 71$ evaluation points (cross, in red).

Table 7
RRMSEs obtained by using the cell-based partition algorithm on Gattinara data.

c	0.5	1.0	1.5
ϕ_3	6.0607E−3	4.9270E−3	4.9556E−3
ϕ_4	8.4848E−3	1.4159E−2	1.4060E−2

6. Conclusions and future work

In this paper we present a new local algorithm for scattered data interpolation in a bidimensional domain. This algorithm is based on a partition of unity and it may efficiently be used for the solution of large-scale interpolation. Indeed, it works well and quickly also when the amount of data to be interpolated is very large, namely for many thousands or even millions of data. This optimized implementation of the partition of unity method is obtained by applying an efficient nearest neighbour searching procedure. Moreover, the proposed algorithm is flexible since different choices of local approximants are allowable, is easily parallelizable, and completely automatic. An application to Earth's topography shows that our approach can also successfully be employed with real life data.

In future work we expect to refine the partition of unity algorithm based on the related partition of unity method adopting suitable data structures like kd-trees and range trees, combining if possible these data structures with the special partition of the domain in cells. Moreover, we are going to extend the proposed algorithm in a straightforward way in three or more dimensions and for more general domains. Then, parallel computation as well as extension to problems involving discontinuous surfaces (see, e.g., [34]) are topics which deserve to be investigated in a more in-depth way, because they have wide-range applications. On the other hand, in numerical experiments we have noted a small loss of accuracy

close to the boundary of the domain since the number of nodes of a subdomain lying on or close to the boundary of the domain is considerably reduced; this limitation is essentially due to the fact that only a little part of the subdomain intersects the domain Ω. Then, the possibility of using an adaptive approach which allows us to suitably increase the dimension of the subdomains only near to the critical region or, as suggested in [35], the use of exponential weights should successfully overcome these problems. Furthermore, we might also analyse the performances of Lobachevsky splines, already proposed in scattered data interpolation and integration in [36,37], when they are used as local approximants within local interpolation schemes. Finally, although the choice of low-order basis functions such as compactly supported Wendland's functions gives a good trade-off between stability and accuracy, in general we believe that the employment of preconditioning techniques could be of great utility. Many efforts and studies in such direction have already been carried out for radial basis function collocation matrices (see [38]) and now further extents are under consideration for radial basis function interpolation, but this topic comes out of the purposes of this article and will be treated in future works.

Acknowledgements

The authors are very grateful to the anonymous referees for their detailed and valuable comments which helped to greatly improve the paper. Moreover, the authors gratefully acknowledge the support of the Department of Mathematics "G. Peano", University of Torino, project "Numerical analysis for life sciences" (2012).

References

[1] M.D. Buhmann, Radial Basis Functions: Theory and Implementation, in: Cambridge Monogr. Appl. Comput. Math., vol. 12, Cambridge Univ. Press, Cambridge, 2003.
[2] G.E. Fasshauer, Meshfree Approximation Methods with MATLAB, World Scientific Publishers, Singapore, 2007.
[3] A. Iske, Radial basis functions: basics, advanced topics and meshfree methods for transport problems, Rend. Semin. Mat. Univ. Politec. Torino 61 (2003) 247–285.
[4] A. Iske, Scattered data approximation by positive definite kernel functions, Rend. Semin. Mat. Univ. Politec. Torino 69 (2011) 217–246.
[5] H. Wendland, Scattered Data Approximation, in: Cambridge Monogr. Appl. Comput. Math., vol. 17, Cambridge Univ. Press, Cambridge, 2005.
[6] R.K. Beatson, J.B. Cherrie, C.T. Mouat, Fast fitting of radial basis functions: methods based on preconditioned GMRES iteration, Adv. Comput. Math. 11 (1999) 253–270.
[7] G.E. Fasshauer, M.J. McCourt, Stable evaluation of Gaussian radial basis function interpolants, SIAM J. Sci. Comput. 34 (2012) A737–A762.
[8] B. Fornberg, G. Wright, Stable computation of multiquadric interpolants for all values of the shape parameter, Comput. Math. Appl. 47 (2004) 497–523.
[9] B. Fornberg, E. Larsson, N. Flyer, Stable computations with Gaussian radial basis functions, SIAM J. Sci. Comput. 33 (2011) 869–892.
[10] G. Song, J. Riddle, G.E. Fasshauer, F. Hickernell, Multivariate interpolation with increasingly flat radial basis functions of finite smoothness, Adv. Comput. Math. 36 (2012) 485–501.
[11] R.K. Beatson, W.A. Light, S. Billings, Fast solution of the radial basis function interpolation equations: domain decomposition methods, SIAM J. Sci. Comput. 22 (2000) 1717–1740.
[12] I. Babuška, J.M. Melenk, The partition of unity method, Int. J. Numer. Methods Eng. 40 (1997) 727–758.
[13] J.M. Melenk, I. Babuška, The partition of unity finite element method: basic theory and applications, Comput. Methods Appl. Mech. Engrg. 139 (1996) 289–314.
[14] H. Wendland, Fast evaluation of radial basis functions: methods based on partition of unity, in: C.K. Chui, L.L. Schumaker, J. Stöckler (Eds.), Approximation Theory X: Wavelets, Splines, and Applications, Vanderbilt Univ. Press, Nashville, TN, 2002, pp. 473–483.
[15] G. Allasia, R. Besenghi, R. Cavoretto, A. De Rossi, Scattered and track data interpolation using an efficient strip searching procedure, Appl. Math. Comput. 217 (2011) 5949–5966.
[16] M.W. Berry, K.S. Minser, Algorithm 798: high-dimensional interpolation using the modified Shepard method, ACM Trans. Math. Software 25 (1999) 353–366.
[17] F.A. Costabile, F. Dell'Accio, F. Di Tommaso, Enhancing the approximation order of local Shepard operators by Hermite polynomials, Comput. Math. Appl. 64 (2012) 3641–3655.
[18] 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.
[19] R.J. Renka, Multivariate interpolation of large sets of scattered data, ACM Trans. Math. Software 14 (1988) 139–148.
[20] R.J. Renka, Algorithm 660: QSHEP2D: quadratic Shepard method for bivariate interpolation of scattered data, ACM Trans. Math. Software 14 (1988) 149–150.
[21] W.I. Thacker, J. Zhang, L.T. Watson, J.B. Birch, M.A. Iyer, M.W. Berry, Algorithm 905: SHEPPACK: modified Shepard algorithm for interpolation of scattered multivariate data, ACM Trans. Math. Software 37 (2010) 1–20. Art. 34.
[22] 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.
[23] R. Cavoretto, A. De Rossi, Numerical comparison of different weights in Shepard's interpolants on the sphere, Appl. Math. Sci. 4 (2010) 3425–3435.
[24] R. Cavoretto, A. De Rossi, Spherical interpolation using the partition of unity method: an efficient and flexible algorithm, Appl. Math. Lett. 25 (2012) 1251–1256.
[25] R. Cavoretto, A unified version of efficient partition of unity algorithms for meshless interpolation, in: T.E. Simos, et al. (Eds.), Proceedings of the International Conference on Numerical Analysis and Applied Mathematics 2012, ICNAAM 12, in: AIP Conference Proceedings, vol. 1479, American Institute of Physics, Melville, New York, 2012, pp. 1054–1057.
[26] R. Cavoretto, A. De Rossi, Achieving accuracy and efficiency in spherical modelling of real data, Math. Methods Appl. Sci. (2014) http://dx.doi.org/10.1002/mma.2906, in press.
[27] H. Wendland, Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree, Adv. Comput. Math. 4 (1995) 389–396.
[28] R. Franke, Scattered data interpolation: tests of some methods, Math. Comp. 38 (1982) 181–200.
[29] R. Schaback, Error estimates and condition numbers for radial basis function interpolation, Adv. Comput. Math. 3 (1995) 251–264.
[30] G.E. Fasshauer, Positive definite kernels: past, present and future, Dolomites Res. Notes Approx. 4 (2011) 21–63.
[31] T.-T. Wong, W.-S. Luk, P.-A. Heng, Sampling with Hammersley and Halton points, J. Graph. Tools 2 (1997) 9–24.
[32] R. Franke, H. Hagen, Least squares surface approximation using multiquadrics and parametric domain distorsion, Comput. Aided Geom. Design 16 (1999) 177–196.
[33] 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.

[34] G. Allasia, R. Besenghi, R. Cavoretto, Adaptive detection and approximation of unknown surface discontinuities from scattered data, Simul. Model. Pract. Theory 17 (2009) 1059–1070.
[35] E. Sáenz-de-Cabezón, L. Javier Hernández, M. Teresa Rivas, E. García-Ruiz, V. Marco, I. Pérez-Moreno, F. Javier Sáenz-de-Cabezón, A computer implementation of the partition of the unity procedure and its application to arthropod population dynamics. A case study on the European grape berry moth, Math. Comput. Simul. 82 (2011) 2–14.
[36] G. Allasia, R. Cavoretto, A. De Rossi, Lobachevsky spline functions and interpolation to scattered data, Comput. Appl. Math. 32 (2013) 71–87.
[37] G. Allasia, R. Cavoretto, A. De Rossi, Numerical integration on multivariate scattered data by Lobachevsky splines, Int. J. Comput. Math. 90 (2013) 2003–2018.
[38] R. Cavoretto, A. De Rossi, M. Donatelli, S. Serra-Capizzano, Spectral analysis and preconditioning techniques for radial basis function collocation matrices, Numer. Linear Algebra Appl. 19 (2012) 31–52.