Dolomites Research Notes on Approximation
Proceedings of *Kernel-based Methods and Function Approximation 2016*, Volume 9 · 2016 · Pages 3–12

Fast strategy for PU interpolation: An application for the reconstruction of separatrix manifolds

Alessandra De Rossi[a] · Emma Perracchione[a] · Ezio Venturino[a]

Abstract

In this paper, the Partition of Unity (PU) method is performed by blending Radial Basis Functions (RBFs) as local approximants and using locally supported weights. In particular, we present a new multidimensional data structure which makes use of an integer-based scheme. This approach allows to perform an optimized space-partitioning structure. Moreover, because of its flexibility, it turns out to be extremely meaningful in the reconstruction of the attraction basins in dynamical systems.

1 Introduction

Over the last years, the topic of numerical approximation of multivariate data has gained popularity in various areas of applied mathematics and scientific computing. Here, we analyze an application in the field of biomathematics. Precisely, we consider the problem of reconstructing unknown surfaces partitioning the phase state into disjoint sets.

When a dynamical system admits more than one stable steady state, the phase space is thus partitioned into different regions, called the basins of attraction of each equilibrium. In such case, the final outcome of a mathematical model depends only on the basin of attraction to which the initial condition belongs. To establish the ultimate system behavior, it is therefore important to assess for each possible attractor its domain of attraction.

More specifically, the approximation of the attraction basins leads to a method consisting of two steps [10]:

i. detection of the points lying on the separatrix manifolds;

ii. interpolation of such points in a suitable way.

For the first item we have implemented several routines, which will be presented in detail in this paper, allowing to detect the points lying on the surfaces determining the basins of attraction. The MATLAB software here discussed is available in [10].

Concerning the second item, we consider a meshfree approximation method and specifically we focus on RBF interpolation. Meshless approaches take advantage of being flexible with respect to the geometry. They are also easy to implement in high dimensions, but the computational cost associated with the solution of large linear systems is not affordable [13]. To avoid this drawback, we focus on local techniques, such as the Partition of Unity (PU) scheme. It has been used for interpolation since around 1960 [12, 29]. Later, it has been coupled with RBFs [14, 32]. Moreover, the PU method for solving Partial Differential Equations (PDEs), first introduced in in the mid 1990s in [3, 25], is nowadays a popular technique [27, 28].

As the global RBF-based method, the PU scheme turns out to be accurate and moreover it is also efficient, but, on the contrary, its implementation in high dimensions is far from being simple. In fact, in such local approach, the efficient organization of the multivariate scattered data set turns out to be a challenging computational issue. Here, starting from the 2D and 3D searching procedures shown in [8, 9], we propose a new multidimensional partitioning structure, named the Integer-based Partitioning Structure (I-PS), which, as a bonus, also reduces the complexity cost of the procedures presented in the above mentioned papers.

The paper is organized as follows. In Section 2, after describing the PU method, we focus on the efficient multidimensional partitioning structure. Then, in Section 3 we present the implicit PU approach and the general framework to detect the points lying on the separatrix manifolds. Section 4 is devoted to numerical results. A final discussion in Section 5 concludes the paper.

2 Efficient computation of PU interpolants

In order to reconstruct the separatrix manifolds, we use the PU method with local RBFs interpolants. Such approach allows to deal with a large number of points in a reasonable time. To be more precise, since the attraction basins are usually described by implicit equations, we perform an implicit PU approximation. Thus, before discussing how to find the separatrix points, we briefly review the theoretical aspects concerning the PU interpolant.

[a]Department of Mathematics “G. Peano”, University of Torino

2.1 The partition of unity method

Given $\mathcal{X}_N = \{\boldsymbol{x}_i \in \Omega, i = 1, \dots, N\}$ a set of distinct *data points* (or *data sites* or *nodes*), distributed in a domain $\Omega \subseteq \mathbb{R}^M$, and an associated set $\mathcal{F}_N = \{f_i, i = 1, \dots, N\}$ of *data values* or *function values*, the problem consists in recovering the unknown function $f : \Omega \to \mathbb{R}$. The PU method enables us to decompose this initial problem into many small ones; in fact its basic idea is to start with a partition of the open and bounded domain Ω into d *subdomains or patches* Ω_j, such that $\Omega \subseteq \cup_{j=1}^{d} \Omega_j$, with some mild overlapping among them [3, 14, 25, 32].

Moreover, according to [32], some additional assumptions on the regularity of the covering $\{\Omega_j\}_{j=1}^{d}$ are needed. In particular, we require a regular covering for $(\Omega, \mathcal{X}_N)$, i.e. $\{\Omega_j\}_{j=1}^{d}$ must fulfill the following properties:

i. for each $\boldsymbol{x} \in \Omega$, the number of subdomains Ω_j, with $\boldsymbol{x} \in \Omega_j$ is bounded by a global constant C;
ii. each subdomain Ω_j satisfies an interior cone condition;
iii. the local fill distances $h_{\mathcal{X}_{N_j}}$ are uniformly bounded by the global fill distance $h_{\mathcal{X}_N}$, where $\mathcal{X}_{N_j} = \mathcal{X}_N \cap \Omega_j$ and

$$h_{\mathcal{X}_N} = \sup_{\boldsymbol{x} \in \Omega} \left(\min_{\boldsymbol{x}_k \in \mathcal{X}_N} \left\| \boldsymbol{x} - \boldsymbol{x}_k \right\|_2 \right).$$

Associated with these subdomains, we consider a partition of unity, i.e. a family of non-negative, continuous functions W_j, $j = 1, \dots, d$, which:

i. are compactly supported, for which precisely $\operatorname{supp}\left(W_j\right) \subseteq \Omega_j$;
ii. form a k-stable partition of unity, i.e.

$$\sum_{j=1}^{d} W_j(\boldsymbol{x}) = 1, \quad \boldsymbol{x} \in \Omega,$$

and for every $\boldsymbol{\beta} \in \mathbb{N}^M$, with $|\boldsymbol{\beta}| \le k$, a constant $C_{\boldsymbol{\beta}} > 0$ exists such that

$$\left\| D^{\boldsymbol{\beta}} W_j \right\|_{L^\infty(\Omega_j)} \le \frac{C_{\boldsymbol{\beta}}}{\left(\sup_{\boldsymbol{x},\boldsymbol{y} \in \Omega_j} \left\| \boldsymbol{x} - \boldsymbol{y} \right\|_2 \right)^{|\boldsymbol{\beta}|}}, \quad j = 1, \dots, d.$$

Among several weight functions W_j, a popular and widely used choice is given by the Shepard's weights, defined as

$$W_j(\boldsymbol{x}) = \frac{\bar{W}_j(\boldsymbol{x})}{\sum_{k=1}^{d} \bar{W}_k(\boldsymbol{x})}, \quad j = 1, \dots, d,$$

where $\bar{W}_j$ are compactly supported functions, with support on Ω_j. Moreover, such family forms a partition of unity.

Once we choose $\{W_j\}_{j=1}^{d}$, the global interpolant is given by

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

where R_j denotes a RBF interpolant defined on a subdomain Ω_j of the form (see [5, 15, 22])

$$R_j(\boldsymbol{x}) = \sum_{k=1}^{N_j} c_k^j \phi(||\boldsymbol{x} - \boldsymbol{x}_k^j||_2), \tag{2}$$

with N_j indicating the number of points on Ω_j, $\boldsymbol{x}_k^j \in \mathcal{X}_{N_j}$, $k = 1, \dots, N_j$, and ϕ being the RBF. The coefficients $\{c_k^j\}_{k=1}^{N_j}$ in (2) are determined by imposing that

$$R_j(\boldsymbol{x}_i^j) = f_i^j, \quad i = 1, \dots, N_j.$$

Thus, the problem of finding the PU interpolant (1) reduces to solve d linear systems of the form

$$A_j \boldsymbol{c}_j = \boldsymbol{f}_j, \tag{3}$$

where $\boldsymbol{c}_j = (c_1^j, \dots, c_{N_j}^j)^T$, $\boldsymbol{f}_j = (f_1^j, \dots, f_{N_j}^j)^T$ and the entries of $A_j \in \mathbb{R}^{N_j \times N_j}$ are given by $(A_j)_{ik} = \phi(||\boldsymbol{x}_i^j - \boldsymbol{x}_k^j||_2)$, $i, k = 1, \dots, N_j$.

In order to give error bounds, we need to define the space $C_\nu^k(\mathbb{R}^M)$ of all functions $f \in C^k$ whose derivatives of order $|\boldsymbol{\beta}| = k$ satisfy $D^{\boldsymbol{\beta}} f(\boldsymbol{x}) = \mathcal{O}\left(||\boldsymbol{x}||_2^\nu\right)$ for $||\boldsymbol{x}||_2 \longrightarrow 0$. We are now able to give the following convergence result [14, 31].

Theorem 2.1. *Let $\Omega \subseteq \mathbb{R}^M$ be open and bounded and suppose that $\mathcal{X}_N = \{\boldsymbol{x}_i, i = 1,\dots,N\} \subseteq \Omega$. Let $\phi \in C_\nu^k(\mathbb{R}^M)$ 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 $f \in \mathcal{N}_\phi(\Omega) = span\{\phi(||\boldsymbol{x} - \cdot||_2), \boldsymbol{x} \in \Omega\}$, where $\mathcal{N}_\phi$ is the native space of ϕ, and its PU interpolant, with $R_j \in \mathcal{T}_\Phi = span\{\phi(||\boldsymbol{x}_k^j - \cdot||_2), \boldsymbol{x}_k^j \in \mathcal{X}_{N_j}\}$, can be bounded by*

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

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

From Theorem 2.1 we can deduce that the PU interpolant allows to accurately compute large RBF interpolants by solving small interpolation problems.

In the next subsection we focus on an efficient implementation of the PU method. In literature, techniques as *kd-trees*, which allow to partition data in any dimension, have been successfully developed, see [1, 11, 14, 31]. However, they are not specifically implemented for the PU method.

Other techniques, built *ad hoc* for bivariate and trivariate PU interpolation, can be found in [8, 9]. Unfortunately, even if they turn out to be really efficient, none of them can be trivially generalized to work in higher dimensions. Thus in this paper, we propose a new multidimensional partitioning structure especially useful for storing the points among the different subdomains.

2.2 The integer-based partitioning structure

In order to implement the PU method, we need to define the so-called *bounding box*

$$\mathcal{L} = \left[\min_{m=1,\dots,M} \left(\min_{i=1,\dots,N} x_{im} \right), \max_{m=1,\dots,M} \left(\max_{i=1,\dots,N} x_{im} \right) \right]^M. \tag{4}$$

Then, we generate both the set of evaluation points $\mathcal{E}_s = \{\tilde{\boldsymbol{x}}_i, i = 1,\dots,s\} \subseteq \Omega$ and the set of PU subdomains centres $\mathcal{C}_d = \{\bar{\boldsymbol{x}}_j, j = 1,\dots,d\} \subseteq \Omega$. Moreover, as usual, we consider hyperspherical patches of radius δ as subdomains. Such radius is chosen so that the properties listed in Subsection 2.1 are satisfied; for reference values see [8, 14].

Afterwards, in order to solve the local interpolation problems, we need to develop a procedure enabling us to organize the scattered data set in any dimension, as kd-trees [1, 11]. Now, referring to the 2D and 3D procedures [8, 9], that in what follows we will call the Sorting-based Partitioning Structures (S-PSs), we can build a multidimensional procedure in which points are stored into q^M blocks, where

$$q = \left\lceil \frac{L}{\delta} \right\rceil, \tag{5}$$

with L denoting the edge of the bounding box $\mathcal{L}$. More precisely, starting from the subspace of dimension $M-1$, obtained projecting along the first coordinate and thus parallel to the remaining ones, we number blocks from 1 to q^M. In order to fix the idea in a 2D context, refer to Figure 1.

From (5) we can deduce that our scheme, as the S-PS, depends on the PU radius δ. In such framework, we will be able to get an efficient procedure to find the nearest points.

In bivariate interpolation, blocks are generated by the intersection of two orthogonal strips. In multivariate problems, blocks are generated by the intersection of M hyperrectangles. In what follows, with abuse of notation, we will continue to call such hyperrectangles with the term strips.

Then, to store the points among the different patches, the following computational issue, known as *containing query*, needs to be solved, i.e.

- given a PU centre $\bar{\boldsymbol{x}}_j$, find the k-th block containing the centre.

Such problem can be easily solved taking into account that, given a PU centre $\bar{\boldsymbol{x}}_j$, if k_m is the index of the strip parallel to the subspace of dimension $M-1$ generated by x_r, $r = 1,\dots,M$ and $r \neq m$, containing the m-th coordinate of $\bar{\boldsymbol{x}}_j$, then the index of the k-th block containing the subdomain centre is

$$k = \sum_{m=1}^{M-1} (k_m - 1) q^{M-m} + k_M. \tag{6}$$

To find the indices k_m, $m = 1,\dots,M$, in (6), we use an integer-based procedure consisting in rounding off to an integer value. Specifically, for each PU centre $\bar{\boldsymbol{x}}_j = (\bar{x}_{j1},\dots,\bar{x}_{jM})$, we have that

$$k_m = \left\lceil \frac{\bar{x}_{jm}}{\delta} \right\rceil. \tag{7}$$

Then, exactly the same procedure is adopted in order to store into the different blocks both scattered data and evaluation points, i.e. the I-PS assigns:

i. to each scattered point $\boldsymbol{x}_i$, $i = 1,\dots,N$, the index of the block in which it lies;

ii. to each evaluation point $\tilde{\boldsymbol{x}}_i$, $i = 1,\dots,s$, the index of the block in which it lies.

After organizing in blocks the data set, given a subdomain Ω_j, we need to:

i. find all data sites belonging to the subdomain Ω_j;
ii. determine all evaluation points belonging to the subdomain Ω_j.

To solve such computational issue, assuming that the j-th centre belongs to the k-th block, the fact that we search for the points in the j-th patch among those in the k-th block and in its 3^M-1 neighboring blocks, i.e. in the k-th neighbourhood, easily follows from (5). This is also graphically pointed out in Figure 1.

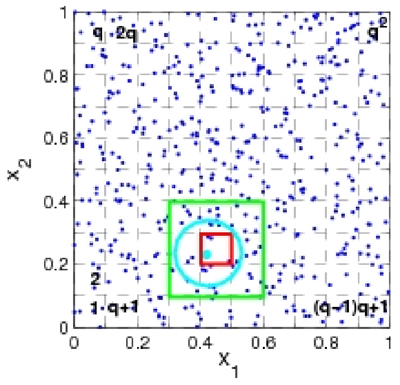

Figure 1: An example of a 2D integer-based partitioning structure: a set of scattered in $\mathcal{L}=[0,1]^2$ (blue), the k-th block (red), a subdomain centre belonging to the k-th block (cyan) and the neighbourhood set (green).

Remark 2.1. In [8, 9] the scattered data set is organized by means of recursive calls to a sorting routine. At first, in such algorithm a sorting procedure is performed to order data sites along the first coordinate. Then, recursive sorting calls order data along the other coordinates. Thus, such partitioning scheme and its complexity depend on the space dimension. Here, such step is replaced by rounding off to an integer value as in (7). As a consequence, since there is no need to use a sorting procedure, the partitioning structure successfully works for a generic dimension M. Moreover, aside from the fact that the I-PS is multidimensional, again thanks to (7), it reduces the complexity of the sorting-based storing procedure from $\mathcal{O}(N\log N)$ to $\mathcal{O}(N)$ time complexity.

To conclude this subsection, we report in Table 1 the CPU times obtained by running the I-PS and the S-PS, considering different sets of bivariate Halton data [14]. Tests have been carried out on a Intel(R) Core(TM) i7 CPU 4712MQ 2.13 GHz processor.

Table 1: CPU times (in seconds) obtained by running the sorting-based procedure (t_{S-PS}) and the integer-based one (t_{I-PS}).

N	25000	50000	100000	200000
t_{I-PS}	5.13	10.68	21.99	45.00
t_{S-PS}	5.21	12.40	28.77	71.55

Given this efficient and flexible routine, we can now focus on the problem of approximating basins of attraction in dynamical systems.

3 Approximation of separatrix manifolds for three equilibria

The surfaces defining the domains of attraction are determined locally (by linearization) in well-known examples [26]. Specifically, even if some techniques to prove the existence of invariant sets have already been developed, none of them, except for particular and well-known cases, allows to have a graphical representation of the separatrix manifolds [23]. Such techniques are based on results from algebraic topology, and thus such methods are not constructive in the sense that they do not give a precise structure and location of the invariant sets. For autonomous systems, the construction of a suitable Lyapunov function can be reduced to the solution of a linear first order PDE. In so doing, such equation is then approximated using meshless collocation methods [16, 17, 18].

In what follows, focusing on dynamical systems composed by three equations, we describe our routines used to detect the separatrix points and we briefly review the implicit PU approach which is used to obtain a graphical representation of the attraction basins.

3.1 Reconstruction of 3D objects via partition of unity interpolation

A common problem in computer aided design and computer graphics is the reconstruction of surfaces defined in terms of point cloud data, i.e. a set of unorganized, irregular points in 3D. Such applications also arise in computer graphics, modeling complicated 3D objects or in medical imaging [6, 14, 30].

Given a point cloud data set, i.e. a data set of the form

$$\mathcal{X}_N = \{\boldsymbol{x}_i \in \mathbb{R}^3, i = 1, \ldots, N\},$$

belonging to an unknown two dimensional manifold $\mathscr{M}$, which can be seen as a surface in $\mathbb{R}^3$, we seek a reasonable approximation to $\mathscr{M}$.

Therefore, in this implicit approach, $\mathscr{M}$ is the surface of all points $\boldsymbol{x} \in \mathbb{R}^3$ satisfying

$$f(\boldsymbol{x}) = 0, \tag{8}$$

for some function f. In other words, the equation (8) is the zero iso-surface of the trivariate function f, and therefore this iso-surface coincides with $\mathscr{M}$. The approximated surface can be constructed via PU interpolation [14]. Unfortunately, the solution of this problem, by imposing the interpolation conditions (8), leads to the trivial solution given by the identically zero function. Therefore, in order to approximate the unknown surface we use additional significant interpolation conditions, i.e. we add an extra set of *off-surface points*. Once we define the augmented data set, we can then compute a three dimensional interpolant $\mathcal{I}$ to the total set of points [14].

Let us suppose that for each point $\boldsymbol{x}_i$ the oriented normal $\boldsymbol{n}_i \in \mathbb{R}^3$ is also given. Then, we construct the extra off-surface points by taking a small step away along the surface normals, i.e. we obtain for each data point $\boldsymbol{x}_i$ two additional off-surface points. One point lies *outside* the manifold $\mathscr{M}$ and is given by

$$\boldsymbol{x}_{N+i} = \boldsymbol{x}_i + \Delta \boldsymbol{n}_i,$$

whereas the other point lies *inside* $\mathscr{M}$ and is given by

$$\boldsymbol{x}_{2N+i} = \boldsymbol{x}_i - \Delta \boldsymbol{n}_i,$$

where Δ is the stepsize. The union of the sets $\mathcal{X}_\Delta^+ = \{\boldsymbol{x}_{N+1}, \ldots, \boldsymbol{x}_{2N}\}$, $\mathcal{X}_\Delta^- = \{\boldsymbol{x}_{2N+1}, \ldots, \boldsymbol{x}_{3N}\}$ and $\mathcal{X}_N$ gives the overall set of points on which the interpolation conditions are assigned. Now, after creating the data set the process becomes fairly simple; in fact we compute the interpolant $\mathcal{I}$ whose zero contour (iso-surface $\mathcal{I} = 0$) interpolates the given point cloud data, and whose iso-surfaces $\mathcal{I} = 1$ and $\mathcal{I} = -1$ interpolate $\mathcal{X}_\Delta^+$ and $\mathcal{X}_\Delta^-$, respectively.

The estimation of the normal unit vectors has been carried out considering [4, 20, 21, 24].

3.2 Detection of separatrix points for three equilibria

In order to approximate the basins of attraction, when the system admits three stable equilibria, the general idea is to find the points lying on the surfaces determining the domains of attraction and finally to interpolate them with a suitable method. The steps of the so-called *detection-interpolation* algorithm are summarized in the `3D-Detec-Interp Algorithm` (see Algorithm 1).

Specifically, in the detection-interpolation scheme, after considering n equispaced initial conditions on each edge of a cube $[0, \gamma]^3$, with $\gamma \in \mathbb{R}^+$, we construct a grid on the faces of the cube (see `Step 2` in the `3D-Detec-Interp Algorithm`)

$$\begin{array}{lcl}
\boldsymbol{p}_{k_1k_2}^{(1)} = (p_{k_11}, p_{k_22}, 0) & \text{and} & \boldsymbol{p}_{k_1k_2}^{(2)} = (p_{k_11}, p_{k_22}, \gamma), \\
\boldsymbol{p}_{k_1k_2}^{(3)} = (p_{k_11}, 0, p_{k_23}) & \text{and} & \boldsymbol{p}_{k_1k_2}^{(4)} = (p_{k_11}, \gamma, p_{k_23}), \\
\boldsymbol{p}_{k_1k_2}^{(5)} = (0, p_{k_12}, p_{k_23}) & \text{and} & \boldsymbol{p}_{k_1k_2}^{(6)} = (\gamma, p_{k_12}, p_{k_23}),
\end{array} \tag{9}$$

with $k_1, k_2 = 1, \ldots, n$, and apply the `Detec Algorithm` (see Algorithm 2) with initial conditions (9), [10]. Essentially, the detection routine performs a bisection algorithm between two initial conditions and determines a point lying on a surface dividing the domains of attraction.

To describe our routines, we first analyze the inputs of the detection-interpolation algorithm:

$n \in \mathbb{N}^+$: number of equispaced points on each edge of the cube; used to define the set of initial data.

$\gamma \in \mathbb{R}^+$: edge of the cube.

$\tau \in \mathbb{R}^+$: tolerance; used during the bisection routine.

$t \in \mathbb{R}^+$: integration time; used during an integration routine. We cannot provide a suitable choice for this parameter because it depends on the dynamical system, but the algorithm checks if the integration time is sufficient.

$\boldsymbol{\lambda} \in \mathbb{R}_+^l$: vector of model parameters, where l is the number of model parameters.

$Q \in \mathbb{N}^+$: number of equilibrium points to be interpolated; typically the origin when it is unstable and the point having all non zero coordinates, such as in population dynamics the coexistence equilibrium, if it is a saddle point.

$E \in \mathbb{R}^{m \times 3}$: matrix of the equilibria, where m is the total number of equilibria. The matrix is organized as follows:

i. from row 1 to row 3: the three stable equilibrium points;

ii. from row 4 to row $Q + 3$: the equilibrium points to be interpolated by the separating surfaces;

iii. from row $Q + 4$ to row m: remaining equilibria (feasible or unfeasible).

$\boldsymbol{\varepsilon} \in \mathbb{R}^3_+$: vector of shape parameters.

$\boldsymbol{d}^{PU} \in \mathbb{N}^3_+$: the number of PU centres along one direction.

$\boldsymbol{K} \in \mathbb{N}^3_+$: vector containing the number of the nearest points; used to estimate the normals.

As outputs three different sets of points are provided by the detection-interpolation algorithm:

$$X_{N_3'} = \{\boldsymbol{x}^{(3)}_{i_3} = \left(x^{(3)}_{i_3 1}, x^{(3)}_{i_3 2}, x^{(3)}_{i_3 3}\right), i_3 = 1, \dots, N_3'\},$$
$$X_{N_2'} = \{\boldsymbol{x}^{(2)}_{i_2} = \left(x^{(2)}_{i_2 1}, x^{(2)}_{i_2 2}, x^{(2)}_{i_2 3}\right), i_2 = 1, \dots, N_2'\},$$
$$X_{N_1'} = \{\boldsymbol{x}^{(1)}_{i_1} = \left(x^{(1)}_{i_1 1}, x^{(1)}_{i_1 2}, x^{(1)}_{i_1 3}\right), i_1 = 1, \dots, N_1'\},$$

where N_3', N_2' and N_1' are the number of points lying on the surfaces delimiting the domain of attraction of E_1 and E_2, E_1 and E_3, and E_2 and E_3, respectively (see `Step 5` in the `3D-Detec-Interp Algorithm`). These sets in pairs, together with the Q equilibria to be interpolated, provide the sets $\mathcal{X}_{N_1}$, $\mathcal{X}_{N_2}$ and $\mathcal{X}_{N_3}$ that identify the three basins of attraction (see `Step 6` in the `3D-Detec-Interp Algorithm`).

Remark 3.1. The number of equilibrium points Q to be interpolated depends on the dynamical system. Specifically, in case of two stable equilibria a saddle point partitions the phase space into two regions. In case of three equilibria, several saddles are involved in the dynamics. However, the three separating manifolds intersect together only at one saddle where all the populations are nonnegative.

Finally, the algorithm interpolates such points and returns values of the interpolants $\mathcal{I}_1$, $\mathcal{I}_2$ and $\mathcal{I}_3$. They approximate the basins of attraction of the first, second and third equilibrium point, respectively (see `Step 7` of the `3D-Detec-Interp Algorithm`).

```
INPUTS: $n, \gamma, \tau, t, \boldsymbol{\lambda}, Q, E, \boldsymbol{\varepsilon}, \boldsymbol{d}^{PU}, \boldsymbol{K}$.
OUTPUTS: $X_{N_1'}, X_{N_2'}, X_{N_3'}, \mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3$.
Step 1: Check if the system presents exactly three stable equilibria.
Step 2: Definition of initial conditions as in (9). Set $i_1 = i_2 = i_3 = 0$, $s = 1$.
Step 3: While $s <= 5$
    Step 4: For $k_1 = 1, \dots, n$
        Step 5: For $k_2 = 1, \dots, n$
                $[\boldsymbol{x}^{(3)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(1)}]$ =Detec$\left(\boldsymbol{p}^{(s)}_{k_1 k_2}, \boldsymbol{p}^{(s+1)}_{k_1 k_2}, t, \tau, \boldsymbol{\lambda}, E\right)$,
                If Detec returns $\boldsymbol{x}^{(3)}$,
                $i_3 = i_3 + 1$,
                add $\boldsymbol{x}^{(3)}$ to $X_{N_3'}$,
                else if returns $\boldsymbol{x}^{(2)}$,
                $i_2 = i_2 + 1$,
                add $\boldsymbol{x}^{(2)}$ to $X_{N_2'}$,
                else if returns $\boldsymbol{x}^{(1)}$,
                $i_1 = i_1 + 1$,
                add $\boldsymbol{x}^{(1)}$ to $X_{N_1'}$.
        $s = s + 2$.
Step 6: Define $\mathcal{Q}$ as the set composed by the $Q$ equilibria which need to be interpolated.
        Set $\mathcal{X}_{N_1} = X_{N_2'} \cup X_{N_3'} \cup \mathcal{Q}$, $\mathcal{X}_{N_2} = X_{N_1'} \cup X_{N_3'} \cup \mathcal{Q}$, $\mathcal{X}_{N_3} = X_{N_1'} \cup X_{N_2'} \cup \mathcal{Q}$.
Step 7: $\mathcal{I}_1$ = Interp$(\mathcal{X}_{N_1}, \varepsilon_1, d_1^{PU}, K_1)$, $\mathcal{I}_2$ = Interp$(\mathcal{X}_{N_2}, \varepsilon_2, d_2^{PU}, K_2)$,
        $\mathcal{I}_3$ =Interp$(\mathcal{X}_{N_3}, \varepsilon_3, d_3^{PU}, K_3)$.
```

Algorithm 1: The `3D-Detec-Interp Algorithm.` The detection-interpolation pseudo-code. It summarizes the steps needed to determine the points lying on the surfaces delimiting the basins of attraction and to reconstruct these surfaces.

More in detail, once we fix the model parameters, we check if the system has three stable equilibria. Then, the detection routine is applied with the initial conditions (9) (see `Step 2-5` in the `3D-Detec-Interp Algorithm`).

The inputs of the `Detec Algorithm` are the parameters τ, t, $\boldsymbol{\lambda}$, E described above and the two initial conditions $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$. Starting from such initial conditions the routine, after performing a bisection algorithm, returns a point lying on a separatrix surface. The output is a point, named

i. $\boldsymbol{x}^{(3)}$, if the point lies on the surface delimiting the domain of attraction of both the first and the second equilibrium point, or

ii. $\boldsymbol{x}^{(2)}$, if lies on the surface determining the basin of attraction of both the first and the third equilibrium point, or

iii. $\boldsymbol{x}^{(1)}$, if the point lies on the surface delimiting the domain of attraction of both the second and the third equilibrium point.

Note that, after defining the tolerances used to check where the initial conditions stabilize (see `Step 1` in the `Detec Algorithm`), we integrate the system and check if the trajectories stabilize to certain equilibria. Then, we check if an initial condition coincides with an unstable steady state or if it lies on a stable submanifold of a saddle point. Let us assume that the first initial data $\boldsymbol{p}_1$ either overlaps an unstable equilibrium or lies on a stable submanifold of a saddle point. In that case $\boldsymbol{p}_1$ is moved away from it in order to obtain, by the bisection routine, a point lying on a separatrix surface (see `Step 3` in the `Detec Algorithm`). In fact the fundamental assumption to perform the bisection is that the two initial conditions evolve towards different attractors. Specifically, in such situation we replace $\boldsymbol{p}_1 = (p_{11}, p_{12}, p_{13})$ with either one of the formulae:

$$\begin{cases} \boldsymbol{p}_1' = (|p_{11} - \tau/4|, p_{12}, p_{13}), & \text{or}, \\ \boldsymbol{p}_1' = (p_{11}, |p_{12} - \tau/4|, p_{13}), & \text{or}, \\ \boldsymbol{p}_1' = (p_{11}, p_{12}, |p_{13} - \tau/4|). \end{cases}$$

The initial data $\boldsymbol{p}_1'$ is chosen in a way so that the trajectories from $\boldsymbol{p}_1'$ and $\boldsymbol{p}_2$ tend to different stable steady states.

Then, let us assume that from $\boldsymbol{p}_1$ the system stabilizes to E_1; depending on where the second condition ultimately stabilizes three cases can occur (see `Step 4` in the `Detec Algorithm`):

i. $\boldsymbol{p}_2$ stabilizes to E_1. The routine discards these initial conditions.

ii. $\boldsymbol{p}_2$ stabilizes to E_2. The bisection procedure is performed between $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$. Specifically, after computing the first midpoint $\boldsymbol{m}$, the system is integrated with initial data $\boldsymbol{m}$ and then we check where this trajectory converges. If the latter evolves towards the first equilibrium point then we set $\boldsymbol{p}_1 = \boldsymbol{m}$; else we set $\boldsymbol{p}_2 = \boldsymbol{m}$, and we repeat this as long as $||\boldsymbol{p}_1 - \boldsymbol{p}_2||_2 > \tau$. When the tolerance is satisfied the bisection stops and a point named $\boldsymbol{x}^{(3)}$ is found.

iii. $\boldsymbol{p}_3$ stabilizes to E_3. With the technique described above the routine determines a point named $\boldsymbol{x}^{(2)}$.

Similarly, if the two initial conditions evolve towards the second and third attractor, a point $\boldsymbol{x}^{(1)}$ is detected.

Summarizing, once we apply the detection algorithm with initial conditions (9), we detect three different sets of points, which in pairs describe the three basins of attraction. Finally, we interpolate the points lying on each surface with the implicit PU method.

Remark 3.2. For the integration of the system we use the MATLAB solver `ode45`. At each step, this intrinsic routine estimates the local error e_i of the i-th component of the approximate solution $\tilde{y}_i$. This error must be less than or equal to an acceptable threshold, which is a function of the specified scalar relative tolerance $e^{(r)}$ and of a vector of absolute tolerances $\boldsymbol{e}^{(a)}$ [2]

$$|e_i| \le \max(e^{(r)}|\tilde{y}_i|, e_i^{(a)}).$$

The value $e^{(r)}$ is a measure of the relative error, the default value being 10^{-3}. It controls the number of correct digits in all solution components. Instead $\boldsymbol{e}^{(a)}$ determines the accuracy when the solution approaches zero. The default value is 10^{-6} for each component. Note that at each step the local truncation error is computed assuming that the exact solution satisfies the numerical scheme and, as a consequence, a good compromise in defining tolerances used to check where the initial condition stabilizes is the one adopted in the detection routine (see `Step 1` in the `Detec Algorithm`).

4 Numerical results

A model chosen to test the detection-interpolation algorithm is the standard competition model. Letting x, y and z denote three populations, we consider the following system [19]

$$\begin{aligned} \frac{dx}{dt} &= p\Big(1 - \frac{x}{u}\Big)x - axy - bxz, \\ \frac{dy}{dt} &= q\Big(1 - \frac{y}{v}\Big)y - cxy - eyz, \\ \frac{dz}{dt} &= r\Big(1 - \frac{z}{w}\Big)z - fxz - gyz, \end{aligned} \tag{10}$$

where p, q and r are the growth rates of x, y and z, respectively, a, b, c, e, f and g are the competition rates, u, v and w are the carrying capacities of the three populations. The model (10) describes the interaction of three competing populations within the same environment [19]. A model with possibly similar related behaviors is introduced in [7].

There are eight equilibrium points. The origin $E_0 = (0, 0, 0)$ and the points associated with the survival of only one population $E_1 = (u, 0, 0)$, $E_2 = (0, v, 0)$ and $E_3 = (0, 0, w)$. Then we have the equilibria with two coexisting populations:

$$E_4 = \left(\frac{uq(av-p)}{cuva-pq}, \frac{pv(cu-q)}{cuva-pq}, 0\right), \quad E_5 = \left(\frac{ur(bw-p)}{fuwb-rp}, 0, \frac{wp(fu-r)}{fuwb-rp}\right), \quad E_6 = \left(0, \frac{vr(we-q)}{gvwe-qr}, \frac{wq(vg-r)}{gvwe-qr}\right).$$

```
INPUTS: $\boldsymbol{p}_1, \boldsymbol{p}_2, t, \tau, \boldsymbol{\lambda}, E$.
OUTPUTS: $\boldsymbol{x}^{(3)}$ or $\boldsymbol{x}^{(2)}$ or $\boldsymbol{x}^{(1)}$ or a failure message.
Step 1: Define $\boldsymbol{\sigma}_i$ to estimate convergence:
            $\sigma_{ik} = 10^{-1}|E_{ik}|$,   or,
            $\sigma_{ik} = 10^{-2}$,   if   $|E_{ik}| < 10^{-2}$,   $i = 1, \dots, m$,   $k = 1, 2, 3$.
Step 2: If $\boldsymbol{p}_1 \to E_i$ and $\boldsymbol{p}_2 \to E_k$, $i$, $k \in \{1, \dots, m\}$, continue,
        else display 'integration time not sufficient', return.
Step 3: If $\boldsymbol{p}_1 \to E_l$ or/and $\boldsymbol{p}_2 \to E_p$, $l$ and/or $p \notin \{1, 2, 3\}$, move $\boldsymbol{p}_1$ or/and $\boldsymbol{p}_2$.
        If $\boldsymbol{p}_1 \to E_i$ and $\boldsymbol{p}_2 \to E_k$, $i, k \in \{1, 2, 3\}$ and $i \neq k$, continue,
        else display 'bisection cannot be performed', return.
Step 4: If $\boldsymbol{p}_1 \to E_1$ or $\boldsymbol{p}_2 \to E_1$,
        eventually $\boldsymbol{p}_1 \rightleftharpoons \boldsymbol{p}_2$ so that $\boldsymbol{p}_1 \to E_1$.
                If $\boldsymbol{p}_2 \to E_1$ display 'bisection cannot be performed', return.
                  else if  $\boldsymbol{p}_2 \to E_2$,
                  $\boldsymbol{x}^{(3)} =$ Bisection($\boldsymbol{p}_1, \boldsymbol{p}_2, t, \tau, \boldsymbol{\lambda}, E$), return $\boldsymbol{x}^{(3)}$,
                  else $\boldsymbol{x}^{(2)} =$ Bisection($\boldsymbol{p}_1, \boldsymbol{p}_2, t, \tau, \boldsymbol{\lambda}, E$), return $\boldsymbol{x}^{(2)}$.
        else if $\boldsymbol{p}_1 \to E_2$ or $\boldsymbol{p}_2 \to E_2$,
        eventually $\boldsymbol{p}_1 \rightleftharpoons \boldsymbol{p}_2$ so that $\boldsymbol{p}_1 \to E_2$ .
                If $\boldsymbol{p}_2 \to E_2$ display 'bisection cannot be performed', return.
                  else $\boldsymbol{x}^{(1)} =$ Bisection($\boldsymbol{p}_1, \boldsymbol{p}_2, t, \tau, \boldsymbol{\lambda}, E$), return $\boldsymbol{x}^{(1)}$,
        else display 'bisection cannot be performed', return.
```

Algorithm 2: The `Detec Algorithm`. The detection pseudo-code. It summarizes the steps needed to find a point lying on a surface delimiting a basin of attraction.

Finally, we have the coexistence equilibrium $E_7 = (x_7, y_7, z_7)$. It can be assessed only via numerical simulations.

Letting $p = 1$, $q = 2$, $r = 2$, $a = 5$, $b = 4$, $c = 3$, $e = 7$, $f = 7$, $g = 10$, $u = 3$, $v = 2$, $w = 1$, the points associated with the survival of only one population, i.e. $E_1 = (3, 0, 0)$, $E_2 = (0, 2, 0)$ and $E_3 = (0, 0, 1)$, are stable, the origin $E_0 = (0, 0, 0)$ is an unstable equilibrium and the coexistence equilibrium $E_7 \approx (0.1899, 0.0270, 0.2005)$ is a saddle point. The remaining equilibria $E_4 \approx (0.6163, 0.1591, 0)$, $E_5 \approx (0.2195, 0, 0.5317)$ and $E_6 \approx (0, 0.1714, 0.2647)$ are other saddle points. The manifolds joining these saddles partition the phase space into the different basins of attraction, but intersect only at the coexistence saddle point, labeled E_7. In this situation we can use the detection-interpolation routine to approximate the basins of attraction.

Here, after establishing conditions to be imposed on the parameters so that the separatrix manifolds exist, at first we compute the normal vectors and consistently orient them to the surfaces by choosing the nearest neighbors K_i, $i = 1, 2, 3$. Typically we set K_i, $i = 1, 2, 3$, between 5 and 10. Finally, we interpolate the points lying on the separatrix surfaces with the implicit PU method, using as local approximants in (1) the compactly supported Wendland's C^2 function

$$\phi(r) = (1 - \varepsilon r)_+^4 (4\varepsilon r + 1), \tag{11}$$

where $\varepsilon \in \mathbb{R}^+$ is the shape parameter and $(\cdot)_+$ denotes the truncated power function. The choice of the Wendland's C^2 function follows from the fact that it can compute a large number of scattered data in a stable way. In other words, such function enables us to achieve a good compromise between stability and accuracy. For this reason compactly supported RBFs are strongly advised. However, this approach turns out to be very flexible; indeed, different choices of local approximants are allowed. Concerning the shape parameter of the basis function, we find good results by taking the shape parameter vector so that $0.01 \leq \varepsilon_i \leq 0.1$, $i = 1, 2, 3$.

The choices of the inputs, described above, are suitable assuming to start with $8 \leq n \leq 15$ equispaced initial conditions on each edge of the cube $[0, \gamma]^3$ and $10^{-3} \leq \tau \leq 10^{-5}$.

Figure 2 shows the separatrix points and the basins of attraction of E_1, E_2 and E_3 (left to right, top to bottom). They are the result of our routines with the input parameters as follows: $n = 15$, $\gamma = 6$, $\tau = 10^{-3}$, $t = 90$, $Q = 2$ (E_0, E_7), $\boldsymbol{\varepsilon} = (0.1, 0.09, 0.08)$, $\boldsymbol{d}^{PU} = (3, 4, 4)$, $\boldsymbol{K} = (7, 8, 6)$.

5 Conclusions and work in progress

In this paper we present a new strategy enabling us to efficiently implement the PU method. It works in any dimension and turns out to be faster than other existing searching techniques. In fact, the procedure used to store the scattered nodes among the different blocks only requires $\mathcal{O}(N)$ operations.

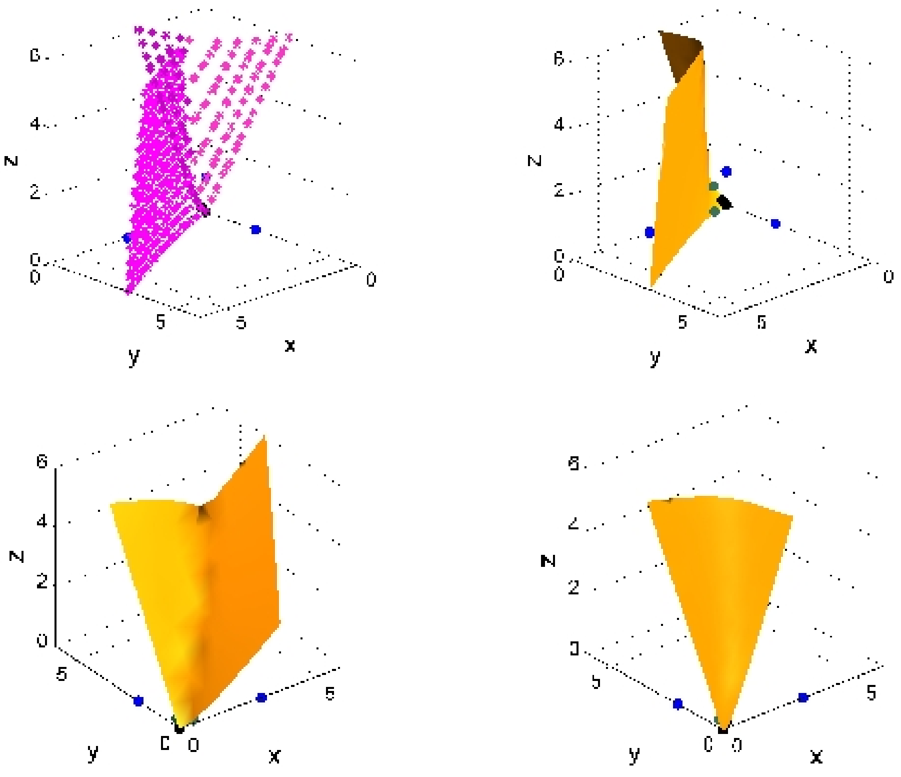

Figure 2: Set of points lying on the surfaces determining the domains of attraction (top left) and the reconstruction of the basin of attraction of E_1 (top right), E_2 and E_3 (bottom, left to right). The four figures (left to right, top to bottom) show the progress of the algorithm: first it generates the points on the separatrices, then in turn each individual basin of attraction. The black and blue circles represent the unstable origin, the coexistence saddle point and the stable equilibria, respectively. Moreover the other saddles (E_4, E_5 and E_6) which, in pairs, lie on the separatrix manifolds of the attraction basins are identified by green circles.

Furthermore, by considering the problem of reconstructing the attraction basins in dynamical systems, we show the versatility of our technique. Moreover, we point out that in many applications a robust and efficient representation of the attraction domains turns out to be very useful. In fact, the knowledge of the attraction basins allows to eventually suggest measures to move the initial condition far away from an unwanted final configuration of the system.

Work in progress consists in extending the proposed partitioning scheme so that it enables us to consider subdomains having variable radii. Such approach turns out to be meaningful especially when strongly non-uniform data are considered. These kinds of data are rather common in real life problems, such as in the reconstruction of 3D objects or in Earth's topography applications.

6 Acknowledgements

The authors sincerely thank the two anonymous referees for helping to improve the paper. This work has been partially supported by Gruppo Nazionale per il Calcolo Scientifico GNCS-INdAM and by the project "Metodi numerici nelle scienze applicate".

References

[1] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *J. ACM* **45**:891–923, 1998.

[2] K.E. Atkinson, W. Han, D.E. Stewart. Numerical Solution of Ordinary Differential Equations. John Wiley & Sons, Inc., Hoboken, 2009.

[3] I. Babuška, J. M. Melenk. The partition of unity method. *Internat. J. Numer. Methods. Engrg.* **40**:727–758, 1997.

[4] M. Behzad, G. Chartrand, L. Lesniak-Foster. Graphs and Digraphs. Prindle, Weber and Schmidt, Boston, 1979.

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

[6] J. C. Carr, W. R. Fright, R. K. Beatson. Surface interpolation with radial basis functions for medical imaging. *IEEE Trans. Med. Imaging* **16**:96–107, 1997.

[7] R. Cavoretto, S. Collino, B. Giardino, E. Venturino. A two-strain ecoepidemic competition model. *Theo. Ecol.* **8**:37–52, 2015.

[8] R. Cavoretto, A. De Rossi. A trivariate interpolation algorithm using a cube-partition searching procedure. *SIAM J. Sci. Comput.* **37**:A1891–A1908, 2015.

[9] R. Cavoretto, A. De Rossi, E. Perracchione. Efficient computation of partition of unity interpolants through a block-based searching technique. *Comput. Appl. Math.* **71**:2568–2584, 2016.

[10] R. Cavoretto, A. De Rossi, E. Perracchione, E. Venturino. Robust approximation algorithms for the detection of attraction basins in dynamical systems. *J. Sci. Comput.* **68**:395–415, 2016.

[11] M. De Berg, M. Van Kreveld, M. Overmars, O. Schwarzkopf. Computational Geometry. Springer, Berlin, 1997.

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

[13] G. E. Fasshauer. Positive definite kernels: Past, present and future. *Dolomites Res. Notes Approx.* **4**:21–63, 2011.

[14] G. E. Fasshauer. Meshfree Approximations Methods with MATLAB. World Scientific, Singapore, 2007.

[15] G. E. Fasshauer, M. J. McCourt. Kernel-based Approximation Methods using MATLAB. World Scientific, Singapore, 2015.

[16] P. Giesl, H. Wendland. Numerical determination of the basin of attraction for asymptotically autonomous dynamical systems. *Nonlinear Anal. Theor.* **75**:2823–2840, 2012.

[17] P. Giesl, H. Wendland. Numerical determination of the basin of attraction for exponentially asymptotically autonomous dynamical systems. Nonlinear Anal. Theor. **74**:3191–3203, 2011.

[18] P. Giesl, H. Wendland. Approximating the basin of attraction of time-periodic odes by meshless collocation. *Discrete Contin. Dyn. Syst.* **25**:1249–1274, 2009.

[19] A. Gosso, V. La Morgia, P. Marchisio, O. Telve, E. Venturino. Does a larger carrying capacity for an exotic species allow environment invasion? – Some considerations on the competition of red and grey squirrels. *J. Biol. Systems* **20**:221–234, 2012.

[20] R. J. Gould. Graph Theory. Dover Publications, Inc., Mineola, N.Y., 2012.

[21] H. Hoppe. Surface Reconstruction from Unorganized Points. PhD thesis, University of Washington, 1994.

[22] A. Iske. Scattered data approximation by positive definite kernel functions. *Rend. Sem. Mat. Univ. Pol. Torino* **69**:217–246, 2011.

[23] T. Johnson, W. Tucker. Automated computation of robust normal forms of planar analytic vector fields. *Discrete Contin. Dyn. Syst.* **12**:769–782, 2009.

[24] I. T. Jolliffe. Principal Component Analysis. Springer-Verlag, New York, 1986.

[25] J. M. Melenk, I. Babuška. The partition of unity finite element method: Basic theory and applications. *Comput. Methods. Appl. Mech. Engrg.* **139**:289–314, 1996.

[26] J. D. Murray. Mathematical Biology. Springer-Verlag, Berlin, 1993.

[27] A. Safdari-Vaighani, A. Heryudono, E. Larsson. A radial basis function partition of unity collocation method for convection-diffusion equations arising in financial applications. *J. Sci. Comput.* **64**:341–367, 2015.

[28] V. Shcherbakov, E. Larsson. Radial basis function partition of unity methods for pricing vanilla basket options. *Comput. Math. Appl.* **71**:185–200, 2016.

[29] D. Shepard. A two-dimensional interpolation function for irregularly spaced data. Proc. of 23rd ACM Nat. Conf., Brandon/Systems Press Inc., Princeton, NJ, 1968, 517–524.

[30] G. Turk, J. F. O' Brien. Modelling with implicit surfaces that interpolate. *ACM Trans. Graph.* **21**:855–873, 2002.

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

[32] 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, Nashville, 2002, 473-483.