Fast Computation of RBF-PU Interpolants

Roberto Cavoretto

Joint work with

Alessandra De Rossi, Emma Perracchione

Department of Mathematics “G. Peano”
University of Torino – Italy

Seminar

May 7th, 2015 – Padova, Italy

Introduction

Main target: interpolate (large) scattered data sets using efficient and accurate algorithms.

- [Allasia, Besenghi, Cavoretto, De Rossi (AMC, 2011)]: new 2D efficient implementation of the modified Shepard's algorithm using MLSAs and RBFs as nodal functions $\Rightarrow$ novelty of the partition of the domain in *strips* for constructing a strip-based searching procedure.
- [Cavoretto, De Rossi (JCAM, 2010)]: extension of the previous idea to the spherical case using ZBFs (instead of RBFs) $\Rightarrow$ partition of the domain in *spherical zones* $\Rightarrow$ spherical zone searching procedure.
- [Cavoretto, De Rossi (AML, 2012)]: the spherical zone algorithm has been generalized using the partition of unity method.
- [Cavoretto, De Rossi (CAMWA, 2014)]: new 2D partition of unity algorithm which is based on the partition of the domain in *cells* (squares) using a double structure of crossed-strips $\Rightarrow$ cell-based searching procedure.
- [Cavoretto, De Rossi (Submitted, 2014)]: extension of the 2D algorithm to the 3D case partitioning the domain in *cells* (cubes).
- [Cavoretto, De Rossi, Perracchione (Submitted, 2015)]: generalization for generic 2D and 3D domains using a new block-based searching procedure.

$$\Downarrow$$

BLOCK-BASED INTERPOLATION ALGORITHMS
FOR GENERIC DOMAINS

Scattered data interpolation problem

Since many applications either arise from a *function approximation problem*, or include *function interpolation* as a fundamental component, we consider the *scattered data interpolation problem* which requires the use of meshfree methods and algorithms.

$$\Downarrow$$

- $\mathcal{X}_n = \{\boldsymbol{x}_i, i = 1, \dots, n\}$, set of *data points* or *nodes*;
- $\mathcal{F}_n = \{f_i = f(\boldsymbol{x}_i), i = 1, \dots, n\}$, set of *data values* or *function values*.

Definition

Given a set $\mathcal{X}_n = \{\boldsymbol{x}_i, i = 1, \dots, n\}$ of n distinct data points on the domain $\Omega \subset \mathbb{R}^N$ and a set $\mathcal{F}_n = \{f_i, i = 1, \dots, n\}$ of the corresponding data values of an (unknown) continuous function $f : \Omega \to \mathbb{R}$, the *interpolation problem* is to find a continuous function $\mathcal{I} : \Omega \to \mathbb{R}$ which satisfies the interpolation conditions, i.e.

$$\mathcal{I}(\boldsymbol{x}_i) = f_i, \qquad i = 1, \dots, n.$$

References: [Buhmann (2003), Fasshauer (2007), Wendland (2005)]

RBF interpolation

- The standard *RBF interpolation problem* is to find an interpolant $R : \Omega \to \mathbb{R}$ of the form

$$R(\boldsymbol{x}) = \sum_{i=1}^{n} c_i \phi(||\boldsymbol{x} - \boldsymbol{x}_i||_2), \quad \boldsymbol{x} \in \Omega, \tag{1}$$

  where $||\cdot||_2$ is the Euclidean norm, and $\phi : [0, \infty) \to \mathbb{R}$ is a RBF. The coefficients $\{c_i\}_{i=1}^n$ are determined by enforcing the interpolation conditions

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

- Imposing the conditions (2) leads to a *symmetric* linear system of equations

$$\Phi \boldsymbol{c} = \boldsymbol{f}, \tag{3}$$

  where $\Phi_{ki} = \phi(||\boldsymbol{x}_k - \boldsymbol{x}_i||_2)$, $k, i = 1, \ldots, n$, $\boldsymbol{c} = [c_1, \ldots, c_n]^T$, and $\boldsymbol{f} = [f_1, \ldots, f_n]^T$. When $\boldsymbol{c}$ is found by solving the *system* (3), we can evaluate the RBF interpolant at a point $\boldsymbol{x}$ as

$$R(\boldsymbol{x}) = \bar{\phi}(\boldsymbol{x})\boldsymbol{c},$$

  where $\bar{\phi}(\boldsymbol{x}) = [\phi(||\boldsymbol{x} - \boldsymbol{x}_1||_2), \ldots, \phi(||\boldsymbol{x} - \boldsymbol{x}_n||_2)]$.

Examples of standard RBFs

RBF	$\phi(r)$
Gaussian C^∞ (G)	$e^{-\alpha^2 r^2}, \quad \alpha > 0$
Inverse MultiQuadric C^∞ (IMQ)	$(1+\gamma^2 r^2)^{-1/2}, \quad \gamma > 0$
MultiQuadric C^∞ (MQ)	$(1+\gamma^2 r^2)^{1/2}, \quad \gamma > 0$
Matérn C^4 (M4)	$e^{-\epsilon r}(\epsilon^2 r^2 + 3\epsilon r + 3), \quad \epsilon > 0$
Wendland C^4 (W4)	$(1-cr)_+^6 \left(35c^2r^2 + 18cr + 3\right), \quad c > 0$
Wendland C^2 (W2)	$(1-cr)_+^4 \left(4cr + 1\right), \quad c > 0$

Partition of unity interpolation

Definition

Given 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, the *partition of unity method* is obtained selecting a family of compactly supported, non-negative, continuous functions W_j with $\text{supp}(W_j) \subseteq \Omega_j$ such that

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

The *global approximant* $\mathcal{I} : \Omega \to \mathbb{R}$ takes the form

$$\mathcal{I}(\boldsymbol{x}) = \sum_{j=1}^{d} R_j(\boldsymbol{x}) W_j(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega, \tag{4}$$

where R_j are local approximants satisfying the interpolation conditions at nodes $\boldsymbol{x}_i$, $i = 1, \ldots, n$.

- *Shepard's weights:*

$$W_j = \bar{W}_j / \sum_k \bar{W}_k.$$

Local RBF interpolants

- Here $R_j : \Omega_j \to \mathbb{R}$ defines a *RBF interpolant* of the form

$$R_j(\boldsymbol{x}) = \sum_{i=1}^{\bar{n}_j} c_i^{(j)} \phi(||\boldsymbol{x} - \boldsymbol{x}_i^{(j)}||_2),$$

  where $\phi : [0, \infty) \to \mathbb{R}$ represents a *RBF*, $|| \cdot ||_2$ denotes the Euclidean norm, and $\bar{n}_j$ indicates the number of data points in Ω_j, i.e. the points $\boldsymbol{x}_i^{(j)} \in \mathcal{X}_j = \mathcal{X}_n \cap \Omega_j$.

- Furthermore, R_j satisfies the *interpolation conditions*

$$R_j(\boldsymbol{x}_i^{(j)}) = f_i^{(j)}, \quad i = 1, \ldots, \bar{n}_j. \tag{5}$$

Note: if the *local approximants* satisfy the interpolation conditions (5), then the global approximant also interpolates at this node, i.e.

$$\mathcal{I}(\boldsymbol{x}_i^{(j)}) = f_i^{(j)}, \quad i = 1, \ldots, \bar{n}_j.$$

Solving the *j-th interpolation problem* (5) leads to a system of linear equations of the form

$$\begin{bmatrix} \phi(||\boldsymbol{x}_1^{(j)} - \boldsymbol{x}_1^{(j)}||_2) & \phi(||\boldsymbol{x}_1^{(j)} - \boldsymbol{x}_2^{(j)}||_2) & \cdots & \phi(||\boldsymbol{x}_1^{(j)} - \boldsymbol{x}_{\bar{n}_j}^{(j)}||_2) \\ \phi(||\boldsymbol{x}_2^{(j)} - \boldsymbol{x}_1^{(j)}||_2) & \phi(||\boldsymbol{x}_2^{(j)} - \boldsymbol{x}_2^{(j)}||_2) & \cdots & \phi(||\boldsymbol{x}_2^{(j)} - \boldsymbol{x}_{\bar{n}_j}^{(j)}||_2) \\ \vdots & \vdots & \vdots & \vdots \\ \phi(||\boldsymbol{x}_{\bar{n}_j}^{(j)} - \boldsymbol{x}_1^{(j)}||_2) & \phi(||\boldsymbol{x}_{\bar{n}_j}^{(j)} - \boldsymbol{x}_2^{(j)}||_2) & \cdots & \phi(||\boldsymbol{x}_{\bar{n}_j}^{(j)} - \boldsymbol{x}_{\bar{n}_j}^{(j)}||_2) \end{bmatrix} \begin{bmatrix} c_1^{(j)} \\ c_2^{(j)} \\ \vdots \\ c_{\bar{n}_j}^{(j)} \end{bmatrix} = \begin{bmatrix} f_1^{(j)} \\ f_2^{(j)} \\ \vdots \\ f_{\bar{n}_j}^{(j)} \end{bmatrix},$$

or simply

$$\Phi^{(j)}\boldsymbol{c}^{(j)} = \boldsymbol{f}^{(j)}. \tag{6}$$

- In particular, the interpolation problem is *well-posed,* i.e., a solution to the problem exists and is unique, if and only if the matrix $\Phi^{(j)}$ is nonsingular.
- A sufficient condition to have nonsingularity is that the corresponding matrix is *positive definite.* In fact, if the matrix $\Phi^{(j)}$ is positive definite, then all its eigenvalues are positive and therefore $\Phi^{(j)}$ is nonsingular.

Definition

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 $\Omega \subseteq \bigcup_{j=1}^d \Omega_j$. Set $\delta_j = \mathrm{diam}(\Omega_j) = \sup_{\boldsymbol{x},\boldsymbol{y}\in\Omega_j} ||\boldsymbol{x} - \boldsymbol{y}||_2$. 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) $\mathrm{supp}(W_j) \subseteq \Omega_j$;
2) $\sum_{j=1}^d W_j(\boldsymbol{x}) \equiv 1$ on Ω;
3) for every $\beta \in \mathbb{N}_0^N$ with $|\beta| \leq k$ there exists a constant $C_\beta > 0$ such that

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

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

We require additional regularity assumptions on the *covering* $\{\Omega_j\}_{j=1}^d$.

Definition

Suppose that $\Omega \subseteq \mathbb{R}^N$ is bounded and $\mathcal{X}_n = \{\boldsymbol{x}_i, i = 1, \ldots, n\} \subseteq \Omega$ 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 $\boldsymbol{x} \in \Omega$, the number of subdomains Ω_j with $\boldsymbol{x} \in \Omega_j$ is bounded by a global constant K;

(b) each subdomain Ω_j satisfies an interior cone condition;

(c) the local fill distances $h_{\mathcal{X}_j,\Omega_j}$, where $\mathcal{X}_j = \mathcal{X}_n \cap \Omega_j$, are uniformly bounded by the global fill distance $h_{\mathcal{X}_n,\Omega}$, i.e.

$$h_{\mathcal{X}_n,\Omega} = \sup_{\boldsymbol{x}\in\Omega} \min_{\boldsymbol{x}_i\in\mathcal{X}_n} ||\boldsymbol{x} - \boldsymbol{x}_i||_2.$$

Theoretical result

Theorem

Let $\phi \in C_{\nu}^{k}(\mathbb{R}^N)$ be a strictly conditionally positive definite function of order m. 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 $f \in \mathcal{N}_{\phi}(\Omega)$ and its partition of unity interpolant (4) can be bounded by

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

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

[Wendland (2005)]

Remark

- *If we compare this result with the global error estimates, we can see that the partition of unity preserves the local approximation order for the global fit.*
- *This means that we can efficiently compute large RBF interpolants by solving small RBF interpolation problems and then glue them together with the global partition of unity $\{W_j\}_{j=1}^{d}$.*
- *The partition of unity approach is a simple and effective technique to decompose a large problem into many small problems while at the same time ensuring that the accuracy obtained for the local fits is carried over to the global one.*

PART I: block-based interpolation algorithms

Outline of block algorithms

The *interpolation algorithms* can be briefly described as follows:

1. Partition the domain Ω into a finite/suitable number of blocks.
2. Consider a *block-based searching procedure* that establishes the minimal number of blocks to be examined, in order to localize the set of nodes for each subdomain.
3. Apply the *Partition of Unity Method* (PUM) which uses RBFs as local approximants.

$$\Downarrow$$

Properties:

- efficiency $\rightarrow$ optimal searching procedure;
- accuracy $\rightarrow$ RBFs;
- high parallelism $\rightarrow$ PUM + block-based partition process.

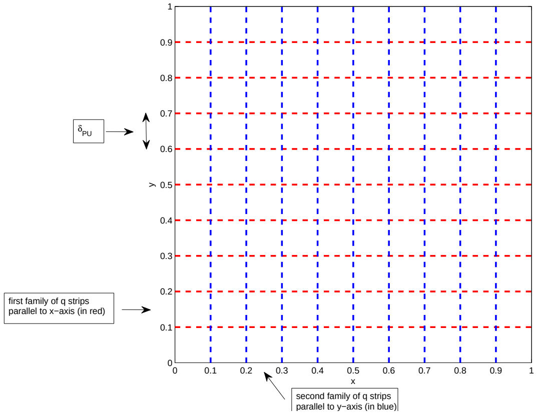
REMARK: in the following, for simplicity, we restrict our focus on the unit square (domain), BUT our software works with generic (convex) domains!!!

Basic idea - 2D case

- 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 in square blocks, which consists in generating two orthogonal families of parallel strips, where the original data set is suitably split up in ordered and well-organized data subsets.

- More precisely, to obtain the block-based partition structure/procedure, we act as follows:
  1. we organize all the data by a *quicksort$_y$ procedure* applied along the y-axis;
  2. 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*;
  3. we create a second family of q strips, parallel to the y-axis, which orthogonally intersect the first strip family $\Rightarrow$ partition of Ω in square blocks.

Families of crossed-strips



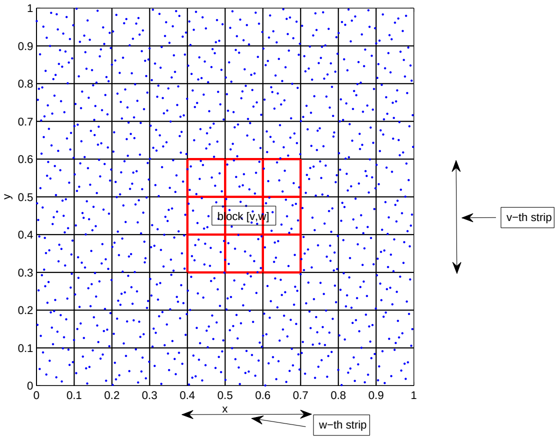
δ_{PU}
first family of q strips parallel to x−axis (in red)
second family of q strips parallel to y−axis (in blue)
y
x
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Block-based searching procedure

- 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 earlier described.
- An effective way to obtain an efficient searching technique is to connect the partition of unity method with the block-based partition structure, assuming that the block width/side δ_{block} is equal to the subdomain radius δ_{PU}, i.e.

$$\delta_{block} \equiv \delta_{PU}.$$

- Though this choice might seem to be trivial, in practice such an imposition means that the search of the nearby points is limited at most to nine blocks: the block on which the considered point lies, and the eight neighbouring blocks.
- The combination between block and subdomain sizes is 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 blocks) and *a priori* ignoring all the other points of Ω.
- Obviously, for all those points belonging to the first and last blocks, i.e. the ones close to the boundary of Ω, a reduction of the total number of blocks to be examined will be required.



1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
y
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
x
block [v,w]
v−th strip
w−th strip

Description of the 2D algorithm

The algorithm consists of three stages:

1. **Distribution phase**

- The nodes in the domain Ω are ordered with respect to a common direction (e.g. the y-axis), by applying a *quicksort$_y$ procedure*.
- For each subdomain point $(\bar{x}_i, \bar{y}_i)$, $i = 1, \dots, d$, a local circular subdomain is constructed, whose half-size (the radius) depends on the subdomain number d, that is

$$\delta_{PU} = \sqrt{\frac{2}{d}}.$$

- 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

$$q = \left\lceil \frac{1}{\delta_{PU}} \right\rceil,$$

  and a *quicksort$_x$ procedure* is applied to order the nodes of 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.

2. **Localization phase**

- The domain (unit square) is partitioned by a block-based structure consisted of q^2 square blocks, whose length of the sides is given by $\delta_{block} \equiv \delta_{PU}$. Then, the following structure is considered:
    - the sets $\mathcal{X}_n$ and $\mathcal{C}_d$ are partitioned by the block structure into q^2 subsets $\mathcal{X}_{m_k}$ and $\mathcal{C}_{d_k}$, $k = 1, \ldots, q^2$,

  where m_k and d_k are the number of points in the k-th block.
- After defining which and how many blocks are to be examined, a *block-based searching procedure* is applied for each subdomain point of $\mathcal{C}_{d_k}$, $k = 1, \ldots, q^2$, to determine all nodes belonging to a subdomain. The number of nodes of the j-th subdomain is counted and stored in n_j, $j = 1, \ldots, d$.
- Taking the $\bar{n}_j$ nodes of the j-th subdomain, a *local interpolant* R_j, $j = 1, \ldots, d$, is found for each subdomain point.

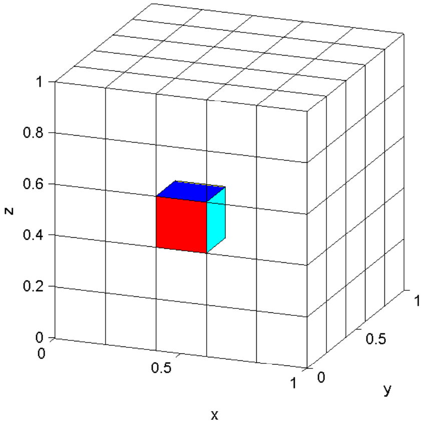
3. **Evaluation phase**

- The evaluation points are ordered with respect to a common direction (e.g. the y-axis), by applying a *quicksort$_y$ procedure*.
- Then, the set $\mathcal{E}_s$ is partitioned into q^2 subsets $\mathcal{E}_{p_k}$, $k = 1, \dots, q^2$, so that the evaluation points of $\mathcal{E}_{p_k}$ belong to the k-th block.
- A block-based searching procedure is applied for each evaluation point of $\mathcal{E}_s$, in order to find all those points belonging to a subdomain of centre $(\bar{x}_i, \bar{y}_i)$ and radius δ_{PU}. The number of subdomains containing the i-th evaluation point is counted and stored in r_i, $i = 1, \dots, s$.
- A local approximant $R_j(x, y)$ and a weight function $W_j(x, y)$, $j = 1, \dots, d$, are found for each evaluation point.
- Applying the PUM (4), the surface can be approximated at any evaluation point $(x, y) \in \mathcal{E}_s$.

Basic idea - 3D case

- The basic idea in constructing this searching procedure comes from the repeated use of a *quicksort* routine with respect to different directions (here, along the z-axis, the y-axis and the x-axis), passing from unordered to ordered data structures.
- This process is strictly related to the construction of a partition of the domain (cube) Ω in smaller cubes, which are obtained generating three orthogonal families of parallelepipeds, where the original data set is suitably split up in ordered and well-organized data subsets.

- More precisely, to obtain the cube-based structure/procedure, we act as follows:
  1. organize all the data by a *$quicksort_z$ procedure* applied along the z-axis;
  2. consider a first family of q parallelepipeds, parallel to the x-axis, and order the points of each parallelepiped by using a *$quicksort_x$ procedure*;
  3. create a second family of q parallelepipeds, parallel to the y-axis, which orthogonally intesect the first family, and order the points of each parallelepiped by using a *$quicksort_y$ procedure*;
  4. construct a third family of q parallelepipeds, parallel to the z-axis, which orthogonally intesect the two previous families $\Rightarrow$ partition of Ω in cubes.


1
0.8
0.6
0.4
0.2
0
z
0
0.5
1
x
0
0.5
1
y

Extension to the 3D case

- Following the same idea described in the 2D case we obtain that the search of the nearby points is limited at most to twenty-seven (3^3) cubes:
  - the cube on which the considered point lies,
  - and the twenty-six neighboring cubes.
- The combination between cube and subdomain sizes provides also here an *optimal* choice, allowing us to search the closest points only considering a very small number of them (only those belonging to one of the twenty-seven cubes) and *a priori* ignoring all the other points of Ω.
- For all those points belonging to cubes close to the boundary of Ω, it will be required a reduction of the total number of cubes to be examined.

$$\Downarrow$$

block-based searching procedure
for 3D interpolation

Complexity of block algorithms

Distribution phase: to build the data structure $\Rightarrow$ computational cost of order $\mathcal{O}(M \log M)$ (M = number of nodes to be sorted) due to the `quicksort` routine.

- $\mathcal{O}(n \log n)$ for the first sorting of all n nodes.

Localization phase: solution of d linear systems of size $\bar{n}_j$ to compute the RBF coefficients:

- $\mathcal{O}(\bar{n}_j^3)$ arithmetic operations to compute the local RBF interpolants.

Evaluation phase: computational cost of order

- $r_i \cdot \mathcal{O}(\bar{n}_j)$ to evaluate the global interpolant at the i-th evaluation point.

Storage locations:

- Nn, Nd and Ns for the data, and $\bar{n}_j$ for the coefficients of each local RBF interpolant.

Comparison: block v.s. kd-tree

N	Block-based structure	kd-tree structure	Block-based search	kd-tree search
2	$\mathcal{O}(3/2n\log n)+$ $\mathcal{O}(3/2s\log s)$	$\mathcal{O}(2n\log n)+$ $\mathcal{O}(2s\log s)$	$\mathcal{O}(1)$	$\mathcal{O}(\log n)+$ $\mathcal{O}(\log s)$
3	$\mathcal{O}(2n\log n)+$ $\mathcal{O}(2s\log s)$	$\mathcal{O}(3n\log n)+$ $\mathcal{O}(3s\log s)$	$\mathcal{O}(1)$	$\mathcal{O}(\log n)+$ $\mathcal{O}(\log s)$

Numerical experiments I

- Tests using the 2D Franke's function:

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

- RBFs with shape parameter $\epsilon > 0$:

$$\phi(r) = (1-\epsilon r)_+^4\,(4\epsilon r+1)\,, \qquad \text{Wendland } C^2 \text{ function (W2)},$$

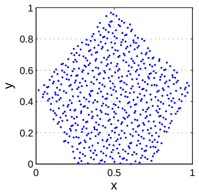
- Maximum Absolute Error (MAE) and Root Mean Square Error (RMSE):

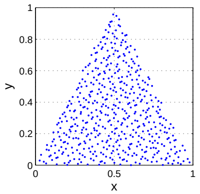
$$MAE = \max_{1\le i\le s} |f(\tilde{\boldsymbol{x}}_i) - \mathcal{I}(\tilde{\boldsymbol{x}}_i)|,$$

$$RMSE = \sqrt{\frac{1}{s}\sum_{i=1}^{s} |f(\tilde{\boldsymbol{x}}_i) - \mathcal{I}(\tilde{\boldsymbol{x}}_i)|^2}.$$

Errors and CPU times

- Interpolation nodes: sets of Halton points (scattered data) in convex domains like a polygon $\Omega \subseteq \mathbb{R}^2$ (e.g., triangle, hexagon, etc.).



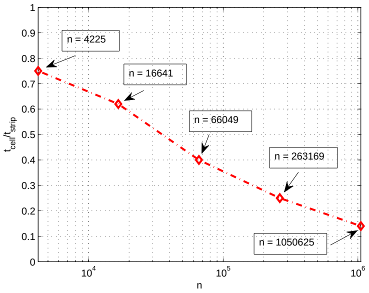


n	MAE	RMSE	t_{block}	t_{kdtree}
622	1.65E − 03	1.40E − 04	1.0	15.3
2499	5.02E − 04	3.30E − 05	3.7	42.3
9999	4.33E − 05	6.33E − 06	9.1	134.0
39991	9.86E − 06	1.25E − 06	34.1	494.1
159994	1.67E − 06	3.05E − 07	142.3	2013.88

Table: Errors and CPU times for pentagon, $\varepsilon = 0.5$.

Comparison of CPU times (in seconds)

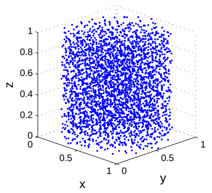
n	t_{cell}	t_{strip}	t_{basic}
4225	**0.3**	0.4	1.8
16641	**0.8**	1.3	14.2
66049	**2.6**	6.5	166.4
263169	**10.2**	41.2	2662.4

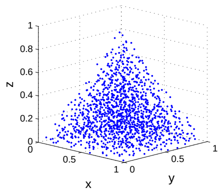


Numerical experiments II

- Tests using the 3D Franke's function.
- W2-RBF with shape parameter $\epsilon > 0$.
- Interpolation nodes: sets of Halton points (scattered data) in convex domains like a polyhedron $\Omega \subseteq \mathbb{R}^3$ (e.g., pyramid, cylinder, etc.).

n	d	q^3	t_{cube}	$t_{no-cube}$
4913	512	6^3	**2.2**	2.6
35937	4096	12^3	**16.6**	35.6
274625	32768	23^3	**138.2**	1241.0





n	MAE	RMSE	t_{block}	t_{kdtree}
3134	$5.94\text{E}-03$	$2.71\text{E}-04$	14.8	266.9
12551	$1.67\text{E}-03$	$6.00\text{E}-05$	53.1	892.7
50184	$4.67\text{E}-04$	$2.27\text{E}-05$	184.5	3141.4
200734	$1.22\text{E}-04$	$7.49\text{E}-06$	1758.1	14693.4
802865	$3.81\text{E}-05$	$2.91\text{E}-06$	-	-

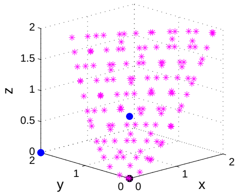
Table: Errors and CPU times for cylinder, $\varepsilon = 0.5$.

Applications to geometric modelling

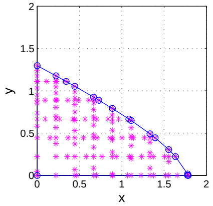
- **Surface approximation from biomathematics:**
  - In dynamical systems saddle points partition the domain into basins of attraction of the remaining locally stable equilibria.
  - This situation is rather common especially in population dynamics models, like competition systems. Trajectories with different initial conditions will possibly converge toward different equilibria, depending on the locations of their respective initial conditions.
  - The set of all points that taken as initial conditions will have trajectories all tending to the same equilibrium is called the basin of attraction of that equilibrium point.

Example of competition model:

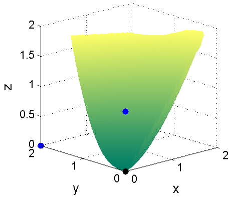
$$
\begin{aligned}
\frac{dx}{dt} &= p\left(1-\frac{x}{u}\right)x - axy - bxz, \\
\frac{dy}{dt} &= q\left(1-\frac{y}{v}\right)y - cxy - eyz, \\
\frac{dz}{dt} &= r\left(1-\frac{z}{w}\right)z - fxz - gyz.
\end{aligned} \tag{7}
$$



z
2
1.5
1
0.5
0
2
1
y
0
0
1
x
2



y
2
1.5
1
0.5
0
0
0.5
1
1.5
2
x



z
2
1.5
1
0.5
0
2
1
y
0
0
1
x
2

- **Reconstruction of 3D objects:**

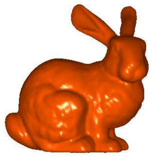
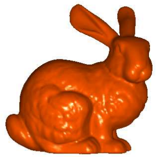



Figure: The Stanford Bunny with 8171 (left) and 35947 (right) data points.

PART II: software

MATLAB software

MATLAB codes free downloadable at:

http://hdl.handle.net/2318/158790

R. CAVORETTO, A. DE ROSSI, E. PERRACCHIONE, *Fast computation of partition of unity interpolants through block-based data structures*, submitted (2015).

Computational issues:

(i) *Range Search:* Given a set of data points $\boldsymbol{x}_i \in \mathcal{X}_n$ and a subdomain Ω_j, find all points situated in that subdomain, i.e. $\boldsymbol{x}_i \in \mathcal{X}_j = \mathcal{X}_n \cap \Omega_j$.

(ii) *Containing Query:* Given $\boldsymbol{x}_i \in \Omega$, return all subdomains Ω_j such that $\boldsymbol{x}_i \in \Omega_j$.

`PUM_2D_CSRBF.m` `PUM_3D_CSRBF.m`	scripts performing the partition of unity using CSRBFs
`BlockBased2D_Structure.m` `BlockBased3D_Structure.m`	scripts that store points into the different neighbourhoods
`BlockBased2D_ContainingQuery.m` `BlockBased3D_ContainingQuery.m`	scripts performing the containing query procedure
`BlockBased2D_RangeSearch.m` `BlockBased3D_RangeSearch.m`	scripts that perform the range search procedure
`BlockBased2D_DistanceMatrix.m` `BlockBased3D_DistanceMatrix.m`	scripts that form the distance matrix of two sets of points for CSRBFs
`inhull.m`	script that tests if a point belongs to the convex hull
`countingsort.m`	script that performs a sorting routine for integers
`haltonseq.m`	script that generates Halton data

Table: The MATLAB codes for the block-based partition of unity algorithms.

References

[1] 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.

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

[3] 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.

[4] 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.

[5] R. Cavoretto, A. De Rossi, *A meshless interpolation algorithm using a cell-based searching procedure*, Comput. Math. Appl. **67** (2014), 1024–1038.

[6] R. Cavoretto, A. De Rossi, *A trivariate interpolation algorithm using a cube-partition searching procedure*, submitted (2014).

[7] R. Cavoretto, A. De Rossi, E. Perracchione, *Fast computation of partition of unity interpolants through block-based data structures*, submitted (2015).

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

[9] 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.

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

Thank you!