Error indicators and refinement strategies for solving Poisson problems through a RBF partition of unity collocation scheme

Error indicators and refinement strategies for solving Poisson problems through a RBF partition of unity collocation scheme

Roberto Cavoretto[a], Alessandra De Rossi[a]

[a]*Department of Mathematics "Giuseppe Peano", University of Turin, Italy*

Abstract

In this article adaptive refinement algorithms are presented to solve Poisson problems by a radial basis function partition of unity (RBF-PU) collocation scheme. Since in this context the problem of constructing an adaptive discretization method to be really effective is still open, we propose some error indicators and refinement strategies, so that each of these two essential ingredients takes advantage of the potentiality of the other one. More precisely, the refinement techniques coupled with a local error indicator is an ad-hoc strategy for the RBF-PU method. The resulting scheme turns out to be flexible and the use of efficient searching procedures enables us a fast detection of the regions that adaptively need the addition/removal of points. Several numerical experiments and applications support our study by illustrating the performance of our adaptive algorithms.

Keywords: meshfree methods, radial basis function collocation, refinement schemes, adaptive algorithms, elliptic partial differential equations
2010 MSC: 65D15, 65M70

1. Introduction

Radial basis function (RBF) approximation methods are effective techniques, which are commonly applied to numerically solve high dimensional interpolation and differential problems. The use of RBF methods has several important advantages. For example, RBF approximation enables us to reach a high convergence order, providing a great flexibility with respect to geometric problem. Due to the meshless nature of RBF approximation schemes, it is then quite easy to formulate and implement the algorithms in higher dimensions (see [5, 15, 17, 30]).

In this article we propose new adaptive refinement algorithms for solving elliptic partial differential equations (PDEs) such as Poisson type problems. Since some traditional RBF collocation schemes such as Kansa's method [20, 21] are generally quite ill-conditioned, we construct our adaptive refinement strategies on the radial basis function partition of unity (RBF-PU) method. This approach is based on a decomposition of the domain into a number of patches or subdomains, which form a cover of the underlying domain. The PU scheme is then combined with a localized RBF approximant for every patch. As shown from numerical experiments and applications, the use of the RBF-PU method allows us a significant reduction of the numerical instability generated by RBFs. However, we remark that the original and more general idea of PU was introduced by Babuška and Melenk [3, 23] in the context of PDEs. Recently, this method combined with RBFs has gained popularity in many fields of applied mathematics, scientific computing and engineering, since it was used to effectively solve large scale approximation and PDE problems (see e.g. [7, 18, 22, 16, 24, 26, 27, 28]). Nevertheless, as far as we know, in the current literature the adaptivity problem via a RBF-PU scheme has not properly been addressed yet. For this reason, in this work we aim to construct

Email addresses: roberto.cavoretto@unito.it (Roberto Cavoretto), alessandra.derossi@unito.it (Alessandra De Rossi)

some adaptive refinement algorithms based on RBF-PU collocation, so that it is possible to determine some (good) error indicator that can be integrated with an effective refinement strategy. In so doing, we propose to use a local error indicator, which turns out to be an ad-hoc strategy for a PU scheme. Associated with this error estimate we suggest two possible variants of a refinement technique that is based on the paradigm to solve, estimate and refine/coarsen till a stop criterion is satisfied. This article extends and completes our earlier work in [9], generalizing the original idea of adaptivity. This approach makes the RBF-PU scheme much more flexible, and accordingly more effective for solving PDE problems in applications. Moreover, we also remark that the work in [11] faces the adaptivity issue in a PU framework, but the error estimates therein are not local and in general even depend on the exact (or true) solution. Nevertheless the latter in real-world applications is not generally available, and so cannot be used at all. Finally, we observe that other adaptive procedures have recently been studied for solving elliptic problems via meshfree RBF techniques, which involve either collocation multiscale methods or finite difference methods (see e.g. [10, 13, 25]).

The article is organized as follows. In Section 2 we first give some basic information on the theory of RBF-PU interpolation, focusing then on the related collocation method used to solve Poisson type problems. Section 3 analyzes the data structures used for the localization and search of points within the PU framework. In Section 4 we describe all various stages of adaptive algorithms, also discussing about different refinement strategies. In Section 5 we show numerical results oriented to illustrate the algorithm performances. Section 6 deals with a few applications. Section 7 contains conclusions and future work.

2. RBF-PU interpolation and collocation

In this section we recall some of the main theoretical results of RBF and RBF-PU interpolation, considering then the RBF-PU collocation for solving Poisson problems.

2.1. RBF interpolation

Given a set $X_N = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ of N distinct *nodes* or *data points* defined in a domain $\Omega \subseteq \mathbb{R}^s$ and the corresponding set of *data* or *function values* $u(\boldsymbol{x}_1), \ldots, u(\boldsymbol{x}_N)$ obtained by possibly sampling any (unknown) function $u : \Omega \to \mathbb{R}$, a standard RBF interpolant $\hat{u} : \Omega \to \mathbb{R}$ is a linear combination of RBFs of the form

$$\hat{u}(\boldsymbol{x}) = \sum_{i=1}^{N} c_i \varphi_\varepsilon(||\boldsymbol{x} - \boldsymbol{x}_i||_2), \quad \boldsymbol{x} \in \Omega, \tag{1}$$

where the coefficient c_i is a real number, $||\cdot||_2$ specifies the so-called Euclidean norm, and $\varphi : \mathbb{R}_{\geq 0} \to \mathbb{R}$ identifies a strictly positive definite (SPD) RBF, which depends on a positive *shape parameter* ε, i.e.

$$\varphi_\varepsilon(||\boldsymbol{x} - \boldsymbol{z}||_2) = \varphi(\varepsilon||\boldsymbol{x} - \boldsymbol{z}||_2), \quad \forall \boldsymbol{x}, \boldsymbol{z} \in \Omega.$$

For the sake of brevity, in the following we will refer to φ_ε as φ. In Table 1 we list some examples of popular SPD RBFs. In particular, we remark that Gaussian, Inverse MultiQuadric and Matérn functions are globally supported and SPD in $\mathbb{R}^s$ for any s, whereas Wendland functions are compactly supported – with support $[0, 1/\varepsilon]$ – and SPD in $\mathbb{R}^s$ for $s \leq 3$ [30].

The coefficients $c_1, \ldots, c_N$ in (1) are determined by enforcing the interpolation conditions $\hat{u}(\boldsymbol{x}_i) = u(\boldsymbol{x}_i)$, $i = 1, \ldots, N$. Consequently, we get a system of linear equations

$$A\boldsymbol{c} = \boldsymbol{u}, \tag{2}$$

where $\boldsymbol{c} = (c_1, \ldots, c_N)^T$, $\boldsymbol{u} = (u_1, \ldots, u_N)^T$, and $A \in \mathbb{R}^{N \times N}$ is a symmetric matrix whose entries are defined as $A_{ki} = \varphi(||\boldsymbol{x}_k - \boldsymbol{x}_i||_2)$, $k, i = 1, \ldots, N$. Since φ is a SPD function, the interpolation matrix A is invertible and so the resulting problem is well-posed. It follows that a solution to the problem exists uniquely [14].

So, once the vector $\boldsymbol{c}$ was found, the RBF interpolant can be evaluated at some point $\boldsymbol{x}$ as follows

$$\hat{u}(\boldsymbol{x}) = \boldsymbol{\varphi}^T(\boldsymbol{x})\boldsymbol{c}, \tag{3}$$

RBF	$\varphi_\varepsilon(r)$
Gaussian C^∞ (GA)	$\exp(-\varepsilon^2 r^2)$
Inverse MultiQuadric C^∞ (IMQ)	$\frac{1}{\sqrt{1+\varepsilon^2 r^2}}$
Matérn C^6 (M6)	$\exp(-\varepsilon r)(\varepsilon^3 r^3 + 6\varepsilon^2 r^2 + 15\varepsilon r + 15)$
Matérn C^4 (M4)	$\exp(-\varepsilon r)(\varepsilon^2 r^2 + 3\varepsilon r + 3)$
Wendland C^4 (W4)	$(1-\varepsilon r)_+^6 (35\varepsilon^2 r^2 + 18\varepsilon r + 3)$
Wendland C^2 (W2)	$(1-\varepsilon r)_+^4 (4\varepsilon r + 1)$

Table 1: Some examples of popular SPD RBFs. Here r denotes the Euclidean distance (or norm) while $(\cdot)_+$ defines the truncated power function.

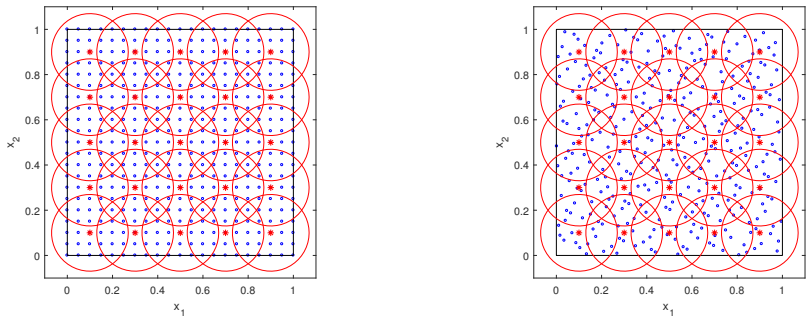
where $\boldsymbol{\varphi}^T(\boldsymbol{x}) = (\varphi(||\boldsymbol{x}-\boldsymbol{x}_1||_2), \ldots, \varphi(||\boldsymbol{x}-\boldsymbol{x}_N||_2))$. Further, as known, the interpolant $\hat{u}$ in (1) (or (3)) turns out to be a function of the so-called *native space* $\mathcal{N}_\varphi(\Omega)$, which is uniquely associated with the RBF. Moreover, if $u \in \mathcal{N}_\varphi$, $\hat{u}$ is the $\mathcal{N}_\varphi$-projection of u into the subspace $\mathcal{N}_\varphi(X_N) = \{\varphi(||\boldsymbol{x}-\boldsymbol{x}_i||_2), \boldsymbol{x}_i \in X_N\}$ [30].

2.2. RBF-PU interpolation

Let $\Omega \subseteq \mathbb{R}^s$ be an open bounded domain, and let $\{\Omega_j\}_{j=1}^d$ be an open bounded cover of Ω that fulfills some mild overlap condition among the patches (or subdomains) Ω_j. In fact, the patches Ω_j need to form a cover of the domain such that

$$\bigcup_{j=1}^{d} \Omega_j \supseteq \Omega,$$

and the number of patches that overlap at one given point $\boldsymbol{x} \in \Omega$ must be bounded by a constant K (independent of d). This overlap must also be sufficient so that each interior point $\boldsymbol{x} \in \Omega$ is located in the interior of at least one patch Ω_j [18]. A typical example of PU patches using two different point distributions in $\mathbb{R}^2$ is shown in Figure 1; the choice of taking circular patches is not unique [27] but, being not relevant for the purposes of this work, it is the one we will later use in our numerical examples.



Grid points Halton points

Figure 1: Example in $\mathbb{R}^2$ of PU patches in red with grid (left) and Halton (right) points in blue.

Given such patches, a partition of unity $\{w_j\}_{j=1}^d$ subordinated to the cover $\{\Omega_j\}_{j=1}^d$ can be defined as

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

where the weight $w_j : \Omega_j \to \mathbb{R}$ is a continuous, nonnegative and compactly supported function with $supp(w_j) \subseteq \Omega_j$. Therefore, for instance we can consider the well-known *Shepard's weight*

$$w_j(\boldsymbol{x}) = \frac{\varphi_j(\boldsymbol{x})}{\sum_{k=1}^{d} \varphi_k(\boldsymbol{x})}, \quad j = 1, \ldots, d, \tag{4}$$

where $\varphi_j(\boldsymbol{x})$ is a compactly supported function on Ω_j such as W2 function shown in Table 1. Such a function is scaled with a shape parameter σ so that $\varphi_j(\boldsymbol{x}) = \varphi(\sigma||\boldsymbol{x}-\boldsymbol{\xi}_j||_2)$, $\boldsymbol{\xi}_j$ being the centre of the weight function.

For each patch, similarly to (1), a localized RBF interpolant $\hat{u}_j : \Omega_j \to \mathbb{R}$ can be written as follows

$$\hat{u}_j(\boldsymbol{x}) = \sum_{i=1}^{N_j} c_i^j \varphi(||\boldsymbol{x} - \boldsymbol{x}_i^j||_2), \tag{5}$$

where N_j is the number of points in Ω_j, i.e. the latter represents the nodes $\boldsymbol{x}_i^j \in X_{N_j} = X_N \cap \Omega_j$. Then, we define the global RBF-PU interpolant

$$\hat{u}(\boldsymbol{x}) = \sum_{j=1}^{d} w_j(\boldsymbol{x}) \hat{u}_j(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega. \tag{6}$$

If the functions $\hat{u}_j$, $j = 1, \ldots, d$, fulfill the following interpolation conditions

$$\hat{u}_j(\boldsymbol{x}_i^j) = u(\boldsymbol{x}_i^j), \quad \boldsymbol{x}_i^j \in \Omega_j, \quad i = 1, \ldots, N_j, \tag{7}$$

the RBF-PU interpolant (6) inherits the interpolation property of the localized RBF interpolants, namely

$$\hat{u}(\boldsymbol{x}_i^j) = \sum_{j=1}^{d} w_j(\boldsymbol{x}_i^j) \hat{u}_j(\boldsymbol{x}_i^j) = \sum_{j=1}^{d} w_j(\boldsymbol{x}_i^j) u(\boldsymbol{x}_i^j) = u(\boldsymbol{x}_i^j).$$

Solving the j-th interpolation problem (7) results in the linear system generated by local RBFs, i.e.

$$\begin{pmatrix} \varphi(||\boldsymbol{x}_1^j - \boldsymbol{x}_1^j||_2) & \cdots & \varphi(||\boldsymbol{x}_1^j - \boldsymbol{x}_{N_j}^j||_2) \\ \vdots & \vdots & \vdots \\ \varphi(||\boldsymbol{x}_{N_j}^j - \boldsymbol{x}_1^j||_2) & \cdots & \varphi(||\boldsymbol{x}_{N_j}^j - \boldsymbol{x}_{N_j}^j||_2) \end{pmatrix} \begin{pmatrix} c_1^j \\ \vdots \\ c_{N_j}^j \end{pmatrix} = \begin{pmatrix} u_1^j \\ \vdots \\ u_{N_j}^j \end{pmatrix},$$

or simply

$$A_j \boldsymbol{c}_j = \boldsymbol{u}_j. \tag{8}$$

Specifically, we notice that the use of SPD functions φ provides (also in the local case) existence and uniqueness of the solution, since the local matrix A_j turns out to be nonsingular [14].

To be able to formulate error bounds, we first consider some technical conditions and then define a few assumptions on regularity of Ω_j [29]. We thus require the partition of unity functions w_j to be *k-stable*. This condition implies that each $w_j \in C^k(\mathbb{R}^s)$ satisfies the inequality

$$\|D^{\boldsymbol{\alpha}} w_j\|_{L_\infty(\Omega_j)} \leq \frac{C_{\boldsymbol{\alpha}}}{\delta_j^{|\boldsymbol{\alpha}|}}, \quad \forall \boldsymbol{\alpha} \in \mathbb{N}_0^s, \text{ with } |\boldsymbol{\alpha}| \leq k, \quad j = 1, \ldots, d,$$

where $C_{\boldsymbol{\alpha}}$ is some positive constant, and $\delta_j = diam(\Omega_j) = \sup_{\boldsymbol{x},\boldsymbol{z}\in\Omega_j} ||\boldsymbol{x}-\boldsymbol{z}||_2$.

Then, we define two common indicators of data regularity: the *separation distance* and the *fill distance*. The former is defined as

$$q_{X_N} = \frac{1}{2}\min_{k\neq i} ||\boldsymbol{x}_k - \boldsymbol{x}_i||_2, \tag{9}$$

while the latter is given by

$$h_{X_N,\Omega} = \sup_{\boldsymbol{x}\in\Omega}\min_{\boldsymbol{x}_i\in X_N} ||\boldsymbol{x}-\boldsymbol{x}_i||_2. \tag{10}$$

Notice that (9) represents the radius of the largest ball that can be centred at every point on X_N such that no two balls overlap, whereas (10) indicates how well the data fill out the domain Ω.

Definition 2.1. *Suppose that $\Omega \subseteq \mathbb{R}^s$ is bounded and $X_N = \{\boldsymbol{x}_1,\dots,\boldsymbol{x}_N\} \subseteq \Omega$. An open and bounded covering $\{\Omega_j\}_{j=1}^d$ is called regular for (Ω, X_N) if the following conditions hold:*

- *$\forall \boldsymbol{x} \in \Omega$, the number of patches Ω_j with $\boldsymbol{x} \in \Omega_j$ is bounded by a global constant K;*
- *every patch Ω_j fulfills an interior cone condition [30];*
- *the local fill distances $h_{X_{N_j},\Omega_j}$ are uniformly bounded by the global fill distance* (10).

After taking the space $C_\nu^k(\mathbb{R}^s)$ of all functions $f \in C^k$ whose derivatives of order $|\boldsymbol{\alpha}| = k$ satisfy $D^\alpha u(\boldsymbol{x}) = \mathcal{O}(||\boldsymbol{x}||_2^\nu)$ for $||\boldsymbol{x}||_2 \to 0$, it is possible to refer to the following convergence result (see [14, Theorem 29.1] and [30, Theorem 15.9]).

Theorem 2.1. *Suppose that $\Omega \subseteq \mathbb{R}^s$ is open and bounded and $X_N = \{\boldsymbol{x}_1,\dots,\boldsymbol{x}_N\} \subseteq \Omega$. Let $\varphi \in C_\nu^k(\mathbb{R}^s)$ be a strictly conditionally positive definite function of order m. If $\{\Omega_j\}_{j=1}^d$ is a regular covering for (Ω, X_N) and $\{w_j\}_{j=1}^d$ is k-stable for $\{\Omega_j\}_{j=1}^d$, then the error between $u \in \mathcal{N}_\varphi(\Omega)$ and its PU interpolant* (6) *is bounded by*

$$|D^{\boldsymbol{\alpha}}u(\boldsymbol{x}) - D^{\boldsymbol{\alpha}}\hat{u}(\boldsymbol{x})| \leq C h_{X_N,\Omega}^{(k+\nu)/2-|\boldsymbol{\alpha}|}|u|_{\mathcal{N}_\varphi(\Omega)}, \quad \forall \boldsymbol{x}\in\Omega,\ |\boldsymbol{\alpha}| \leq k/2.$$

Comparing the convergence result given in Theorem 2.1 with the global error estimates in [30], it turns out to be evident that the PU interpolation guarantees the preservation of the local approximation order for the global fit. Hence a big problem can suitably be decomposed by solving small interpolation problems and then glue them together with the partition of unity weights. As a result, the PU approach can be viewed as a simple and efficient tool to decompose a large interpolation problem into many small problems, simultaneously ensuring that the accuracy obtained for the local fits is carried over to the global one.

2.3. RBF-PU collocation

In this subsection we consider the RBF-PU method for solving differential problems via a collocation scheme. In so doing, we focus on elliptic PDEs such as Poisson type problems. Therefore, for the elliptic operator $\mathcal{L} = -\Delta$, the Poisson equation is defined as

$$-\Delta u(\boldsymbol{x}) = f(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega, \tag{11}$$

together with Dirichlet boundary conditions

$$u(\boldsymbol{x}) = g(\boldsymbol{x}), \quad \boldsymbol{x} \in \partial\Omega. \tag{12}$$

This problem can then be discretized on a set X_N of collocation nodes, which can be split into the two subsets X_{N_i} and X_{N_b} of interior and boundary points, i.e. $X_N = X_{N_i} \cup X_{N_b}$, where N_i and N_b represent

respectively the number of interior and boundary points. Thus, supposing that φ and w_j is at least C^2 and the PDE problem can be approximated by (6), we obtain

$$\begin{aligned} -\Delta\hat{u}(\boldsymbol{x}_i) = -\sum_{j=1}^{d} \Delta\left(w_j(\boldsymbol{x}_i)\hat{u}_j(\boldsymbol{x}_i)\right) = f(\boldsymbol{x}_i), \qquad & \boldsymbol{x}_i \in \Omega \\ \hat{u}(\boldsymbol{x}_i) = \sum_{j=1}^{d} w_j(\boldsymbol{x}_i)\hat{u}_j(\boldsymbol{x}_i) = g(\boldsymbol{x}_i), \qquad & \boldsymbol{x}_i \in \partial\Omega. \end{aligned} \tag{13}$$

In particular, we can expand the (Laplace) elliptic operator Δ as follows

$$-\Delta\left(w_j(\boldsymbol{x}_i)\hat{u}_j(\boldsymbol{x}_i)\right) = -\Delta w_j(\boldsymbol{x}_i)\hat{u}_j(\boldsymbol{x}_i) - 2\nabla w_j(\boldsymbol{x}_i)\cdot\nabla\hat{u}_j(\boldsymbol{x}_i) - w_j(\boldsymbol{x}_i)\Delta\hat{u}_j(\boldsymbol{x}_i), \quad \boldsymbol{x}_i \in \Omega, \tag{14}$$

and define the local vectors $\hat{\boldsymbol{u}}_j = (\hat{u}_j(\boldsymbol{x}_1^j), \ldots, \hat{u}_j(\boldsymbol{x}_{N_j}^j))^T$ and $\boldsymbol{c}_j = (c_1^j, \ldots, c_{N_j}^j)^T$, where $\boldsymbol{c}_j = A_j^{-1}\hat{\boldsymbol{u}}_j$ (see [11]). From (5), we have then the relations

$$\Delta\hat{\boldsymbol{u}}_j = A_j^{\Delta}A_j^{-1}\hat{\boldsymbol{u}}_j, \qquad \nabla\hat{\boldsymbol{u}}_j = A_j^{\nabla}A_j^{-1}\hat{\boldsymbol{u}}_j, \tag{15}$$

A_j^{Δ} and A_j^{∇} being matrices whose entries are given by

$$(A_j^{\Delta})_{ki} = \Delta\varphi(||\boldsymbol{x}_k^j - \boldsymbol{x}_i^j||_2),$$

and

$$(A_j^{\nabla})_{ki} = \nabla\varphi(||\boldsymbol{x}_k^j - \boldsymbol{x}_i^j||_2).$$

In addition, the diagonal matrices W_j^{Δ}, W_j^{∇} and W_j are defined as

$$\begin{aligned} W_j^{\Delta} &= diag\left(\Delta w_j(\boldsymbol{x}_1^j), \ldots, \Delta w_j(\boldsymbol{x}_{N_j}^j)\right), \\ W_j^{\nabla} &= diag\left(\nabla w_j(\boldsymbol{x}_1^j), \ldots, \nabla w_j(\boldsymbol{x}_{N_j}^j)\right), \\ W_j &= diag\left(w_j(\boldsymbol{x}_1^j), \ldots, w_j(\boldsymbol{x}_{N_j}^j)\right). \end{aligned}$$

The discrete operator Q_j is obtained by differentiating (13) by means of a product derivative rule and of the relations in (15). Now, taking into account the expansion in (14), the discrete local Laplacian operators can be expressed as

$$\bar{Q}_j = \left(W_j^{\Delta}A_j + 2W_j^{\nabla}A_j^{\nabla} + W_jA_j^{\Delta}\right)A_j^{-1}.$$

To derive the discrete local PDE operator, including the boundary conditions, we get

$$(Q_j)_{ki} = \begin{cases} (\bar{Q}_j)_{ki}, & \boldsymbol{x}_i^j \in \Omega, \\ \delta_{ki}, & \boldsymbol{x}_i^j \in \partial\Omega, \end{cases}$$

δ_{ki} being the Kronecker delta. As a result, the global discrete operator is obtained by assembling the local matrices Q_j into the global sparse matrix Q. Finally, the numerical solution can be found by solving the linear system

$$Q\boldsymbol{z} = \boldsymbol{v}, \tag{16}$$

where $\boldsymbol{z} = (\hat{u}(\boldsymbol{x}_1), \ldots, \hat{u}(\boldsymbol{x}_N))^T$ and $\boldsymbol{v} = (v_1, \ldots, v_N)^T$ is defined by

$$v_i = \begin{cases} f(\boldsymbol{x}_i), & \boldsymbol{x}_i \in \Omega, \\ g(\boldsymbol{x}_i), & \boldsymbol{x}_i \in \partial\Omega. \end{cases}$$

As for the RBF collocation scheme [19], also in the RBF-PU collocation we have no theoretical proof that proves the sparse collocation matrix Q in (16) is invertible. This implies that the matrix Q could be singular for some configurations of points. However, the fact of constructing an adaptive scheme allows us to search for a "good" final configuration of data for which matrix singularity is avoided. So, here, we deal with the problem essentially from a numerical point of view, whereas the treatment of more theoretical issues such as well-posedness will be considered in future works.

3. Fast algorithms for localization and search of points

In the PU framework, a remarkable task regards the selection of the collocation nodes in the various patches Ω_j, $j = 1, \ldots, d$. From a computational standpoint efficiency assumes in fact an important role to fast assembly the collocation matrix and, as a result, determine the PDE solution by RBF-PU method. Thus, this section describes the searching technique used to localize and determine all the points belonging to a given patch. Such procedure is related to the construction of a partitioning structure used in [7, 8] to efficiently solve two- and three-dimensional interpolation problems, and here suitably adapted for solving PDE problems by a collocation method. Although these algorithms can be applied to generic domains, for the sake of brevity and clarity we now focus on the domain $\Omega = [0, 1]^2 \subseteq \mathbb{R}^2$.

3.1. Description of the algorithms

`Stage 1: localization phase.` At first, we define a cover of the unit square domain Ω consisting of patches of radius $\delta = \sqrt{2/d}$, where each patch has a center in Ω. We remark that the larger (smaller) the value of d is, the finer (coarser) the structure becomes. The number d of patches is selected by suitably adapting the definition given in [7] for the purposes of this work, i.e. developing an adaptive refinement scheme for RBF-PU collocation.

`Stage 2: partitioning phase.` In order to find all collocation nodes that belong to a given patch Ω_j and then apply our RBF-PU scheme, we construct a structure that partitions the domain Ω in blocks of squared shape. Such process leads to an effective searching technique which is quite efficient from a computational standpoint. More precisely, it consists in partitioning the unit square Ω with b^2 square blocks, where b is the number of blocks along one side of the domain defined by $b = \lceil 1/\delta \rceil$. Hence the side of each block of squared shape is slightly less or equal to the patch radius. This choice enables us to examine in the searching procedure only a small number of blocks, thus significantly reducing the computational effort with respect to the most advanced searching procedures such as kd-trees [2, 30]. In the partitioning process we then number the blocks of square shape from 1 to b^2, following the lexicographic order "bottom to top, left to right". By repeatedly using a *quicksort* routine, we can thus split by the block-based structure the set of (interior and boundary) collocation points in the b^2 square blocks, as to easily be able to find the b^2 subsets containing the points belonging to nine blocks: the k-th block and its eight neighboring blocks, see Figure 2 (left). In this way, we get a fast searching procedure to detect all points belonging to each patch Ω_j.

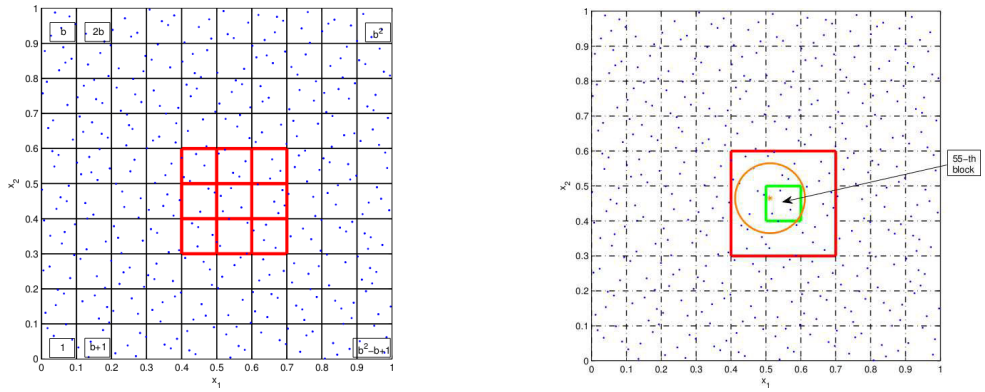


Figure 2: Example of block-based structure applied to interior collocation points (blue): to left the partitioning structure with the k-th block and its eight neighboring blocks (red); to right the k-th block (green, with $k = 55$) and a neighborhood associated with a point denoted by $\star$ (orange) belonging to the k-th block.

`Stage 3: searching phase.` After partitioning the collocation points in the b^2 square blocks, we need to answer the following queries, known respectively as *containing query* and *range search*, i.e., i) given a

patch centre belonging to Ω, return the k-th square block containing that centre; ii) given a set of collocation nodes and a patch Ω_j, find all nodes located in that patch. Thus, given a patch centre, the block-based containing query provides the index of the k-block containing such centre, i.e. $k = (k_1 - 1)\, b + k_2$. An example of this strategy is shown in Figure 2 (right). Then, after answering the first query, given a patch Ω_j the search routine enables to determine all points lying in the j-th patch. Specifically, assuming that the j-th patch centre belongs to the k-th block, the block-based search process examines all data lying in the k-th block and in its eight neighboring blocks, see Figure 2 (bottom). Obviously, if a block lies on the boundary of the domain Ω, the block-based structure and the search process allow to further reduce the number of neighboring blocks that need to be examined.

3.2. Complexity analysis

The localization phase in `Stage 1` is essentially a sort of data pre-processing which is not involved in complexity cost. So we focus on `Stage 2` used to partition the N collocation points in blocks. In the assessment of the total complexity we should also consider the cost which derives from the storing of other points used in the numerical scheme (i.e., test points, evaluation points, etc.). However, for simplicity, here we explicitly refer only to collocation points, further supposing to apply our localization and search procedures at the initial set of points, i.e. $X_{N^{(1)}} = X_N$. Then, as we will see in Section 4, in our adaptive scheme this process is iterated to the set $X_{N^{(i)}}$, $i = 2, 3, \ldots$. Since adaptivity, in particular at the final phase, may create distribution of points not much uniform in the domain Ω, we give a sketch of complexity assuming to have some data regularity. As stated in `Stage 2`, the partitioning structure is based on the use of a quicksort routine, i.e. recursive calls to the MATLAB `sortrows.m` routine, which requires $\mathcal{O}(N \log N)$ time complexity and $\mathcal{O}(\log N)$ space, where N is the number of points to be sorted. This approach allows us to partition the domain, organizing the points in a block-based structure. So we can estimate that the cost of this second phase is about $\mathcal{O}(\frac{3}{2} N \log N + \frac{N}{2})$. In regards to the search process of `Stage 3` we need further to apply a quicksort procedure to sort distances of points within each patch. As the data point distribution is supposed to be quite uniform (and accordingly is in the nine neighboring blocks), the complexity of this search phase is estimated by $\mathcal{O}(1)$.

4. Adaptive refinement schemes

As known for instance from [4], when one seeks to get an adaptive technique, it is important to devise some error indicator that can integrate perfectly with a refinement algorithm. Therefore, in order to derive an adaptive algorithm to be really effective, a good combination between error indicator and refinement strategy has to be found so that each of these two essential ingredients can benefit from the efficacy of the other one. This aspect is of particular relevance in this context, because it allows us to increase the benefits when one adaptively applies a numerical method to solve PDEs. With this scope, referring to [12], the authors in [11] have presented a few techniques for PU collocation to carry out a refinement of the collocation nodes, which essentially consists in the addition/removal of any points. However, such refinement procedures are based on the use of a true solution (although in real-world problems it is usually unknown), or of any indicator that in some practical situations cannot give satisfactory results leading to low accuracy or creation of huge number of points.

4.1. Error indicators

Based on previous considerations we briefly recall two error indicators presented for the first time in [9], whose use is now coupled to a new refinement algorithm. Since we suppose the exact solution is not available, we define these indicators so that they only depend on the approximate solution coming from the RBF-PU collocation or some its related information. Therefore, focusing on the k-th iteration, the error indicators are defined as follows.

`Indicator 1.` We compare two RBF-PU collocation solutions computed on two different sets of nodes, namely a coarser set $X_{N_c^{(k)}}$ and a finer one $X_{N_f^{(k)}}$, with $X_{N_c^{(k)}} \subseteq X_{N_f^{(k)}}$. Then, for $k = 1, 2, \ldots$, providing an

estimate on the set $X_{N_c^{(k)}}$, the error indicator is given by

$$E(\boldsymbol{x}_i^{(k)}) = \left| \hat{u}_{X_{N_f^{(k)}}}(\boldsymbol{x}_i^{(k)}) - \hat{u}_{X_{N_c^{(k)}}}(\boldsymbol{x}_i^{(k)}) \right|, \qquad \boldsymbol{x}_i \in X_{N_c^{(k)}}. \tag{17}$$

Note that indicator (17) is based on the assumption that a finer set gives us a greater accuracy than the solution computed on a coarser one.

`Indicator 2.` It is obtained by making a comparison between the RBF-PU collocation solution $\hat{u}$ computed on $X_{N_j{}^{(k)}}$ and the localized RBF interpolant $(\mathcal{I}\hat{u})_{\Omega_j}$ constructed on the $N_j^{(k)}$ collocation points of Ω_j by using the collocation solution found. Therefore, we can evaluate the error on a set $T^{(k)} = \{\boldsymbol{t}_1^{(k)}, \ldots, \boldsymbol{t}_{n^{(k)}}^{(k)}\}$ of test points, thus deriving the indicator

$$E(\boldsymbol{x}_i^{(k)}) = \left| \hat{u}(\boldsymbol{t}_i^{(k)}) - (\mathcal{I}\hat{u})_{\Omega_j}(\boldsymbol{t}_i^{(k)}) \right|, \qquad \boldsymbol{t}_i^{(k)} \in T^{(k)}. \tag{18}$$

Note that the use of indicator (18) is justified by the fact that if the collocation solution is not accurate then a significant difference should also be detected by the local interpolant of data coming from collocation.

Remark 4.1. *Since the indicator* (18) *enables us to work locally on each patch, it turns out to be more efficient than* (17) *in terms of computational cost. For this reason, in the experiments we will restrict our attention only on the use of* (18).

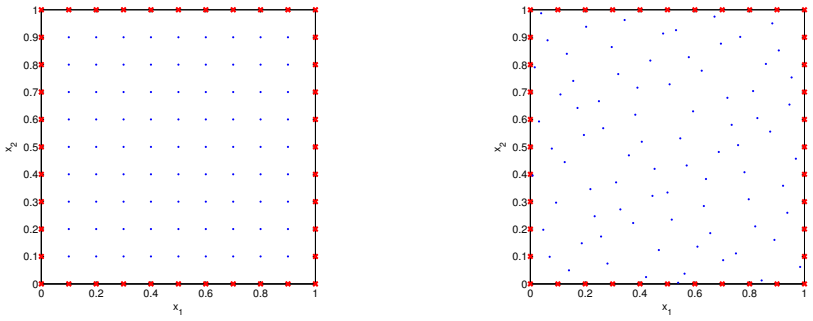
4.2. Refinement strategies

Associated with any error indicator, we have to define some refinement strategies that enable the adaptive scheme to be effective. In particular, in this work, we propose a more general refinement algorithm than the one given in [9]. In fact, even though the structure of these computational procedures is quite similar, the new version guarantees a greater flexibility allowing us to add and/or remove points, in an adaptive way, by using different data distributions. As we will see in Section 5, this choice (i.e. the mode of adding/removing points) can significantly influence the numerical results. For this reason, we give a general description of the refinement algorithm, then specifying which are the possible changes that can be carried out. Moreover, although these techniques can be applied to generic (regular or irregular) domains, for the sake of simplicity in the following we discuss in detail the case in which the domain is a square.

First of all, we take a set $X_{N^{(1)}} \equiv X_N = \{\boldsymbol{x}_1^{(1)}, \ldots, \boldsymbol{x}_{N^{(1)}}^{(1)}\}$ of collocation nodes, which is split into the sets $X_{N_i^{(1)}}$ and $X_{N_b^{(1)}}$ of interior and boundary nodes, respectively. Obviously, the adaptive process is then iterated and the sets $X_{N^{(k)}} = X_{N_i^{(k)}} \cup X_{N_b^{(k)}}$ are updated, for $k = 1, 2, \ldots$. In Figure 3 we give some possible examples of initial configuration of collocation points, using grid points on the boundary and grid (left) or uniformly random Halton (right) points in the interior of the domain Ω [14]. Both of these configurations will be used in our numerical experiments.

Basically, the refinement method we discuss here has some connections with the so-called residual subsampling technique studied in [12], and then used in [11]. Nevertheless, in particular in the latter paper the authors construct their refinement strategy assuming that the true solution of the PDE problem is given. As a result, our technique points out a greater strength and chance of use than previous method since in real-life situations, we need an adaptive scheme where both error estimate and refinement strategy can exploit at best each other all own potentialities. So, we propose a new refinement technique for solving Poisson problems by an adaptive RBF-PU collocation scheme. This procedure follows the common paradigm to solve, estimate and refine/coarsen till a stop criterion is satisfied. It can therefore be sketched as shown in Procedure 1.

Remark 4.2. *In the refinement algorithm we update step-by-step the set* $X_{N^{(k)}}$*, for* $k = 1, 2, \ldots$*. It is therefore evident that at each iteration the error indicator* (18) *needs to solve a collocation problem and an interpolation one. The former is characterized by computation of the approximate solution* $\hat{u}$ *via RBF-PU method, while the latter is based on the computation of* d *local RBF interpolants* $(\mathcal{I}\hat{u})_{\Omega_j}$ *constructed on*



Interior grid points Interior Halton points

Figure 3: Examples of starting sets of (interior and boundary) collocation nodes with $N_i = 81$ and $N_b = 40$.

Procedure 1: Refinement algorithm

STEP 1 Create a set $X_{N^{(1)}}$ of (initial) collocation nodes

STEP 2 Fix two positive tolerances (or thresholds) $\tau_{\min}$ and $\tau_{\max}$, such that $0 < \tau_{\min} < \tau_{\max}$

STEP 3 For $k = 1, 2, \ldots$ compute the k-th approximate collocation solution on $X_{N^{(k)}}$

STEP 4 Define a set $T^{(k)}$ of test points

STEP 5 Evaluate the error at the test point $\boldsymbol{t}_i^{(k)} \in T^{(k)}$ via an (error) indicator

STEP 6 If the error indicator

`i)` $E(\boldsymbol{t}_i^{(1)}) > \tau_{\max}$, add the test point $\boldsymbol{t}_i^{(k)}$ among the collocation points

`ii)` $E(\boldsymbol{t}_i^{(1)}) < \tau_{\min}$, remove the collocation node $\bar{\boldsymbol{x}}_i^{(k)}$ closest to $\boldsymbol{t}_i^{(k)}$ from the collocation points

Define the sets

$$Z_{T_{\max}^{(k)}} = \{\boldsymbol{t}_i^{(k)} \in T^{(k)} \,:\, E(\boldsymbol{t}_i^{(k)}) > \tau_{\max},\, i = 1, \ldots, T_{\max}^{(k)}\}$$
$$Z_{T_{\min}^{(k)}} = \{\bar{\boldsymbol{x}}_i^{(k)} \in X_{N^{(k)}} \,:\, E(\boldsymbol{t}_i^{(k)}) < \tau_{\min},\, i = 1, \ldots, T_{\min}^{(k)}\}$$

and construct the refined set

$$X_{N^{(k+1)}} = X_{N_i^{(k+1)}} \cup X_{N_b^{(k+1)}}, \qquad \text{with } X_{N_i^{(k+1)}} = (X_{N_i^{(k)}} \cup Z_{T_{\max}^{(k)}}) \backslash Z_{T_{\min}^{(k)}}$$

STEP 7 Stop when $Z_{T_{\min}^{(k)}} = \emptyset$

the $N_j^{(k)}$ collocation points lying in Ω_j, $j = 1, \ldots, d$. This approach requires firstly to invert the collocation matrix given in the linear system (16), whose cost is $O((N^{(k)})^3)$, whereas the inversion of a local interpolation matrix needs only $O((N_j^{(k)})^3)$ operations, with $N_j^{(k)} << N^{(k)}$. Finally, once the final collocation points were determined, the adaptive refinement algorithm stops and the resulting approximate solution can be evaluated on a set of evaluation points. The cost of this final step is $O(1)$. Note that here we use a direct solver to find the solutions of linear systems, because the target of this work is devoted to the construction and validation of new adaptive refinement strategies. It is therefore out of our scopes, for instance, the application of new or sophisticated iterative solvers from numerical linear algebra.

4.3. Discussion about the refinement algorithms

After considering the generic description of the refinement algorithm given in the previous subsection, we discuss about possible variants that can result in different numerical results. In fact, as evident from Figure 3, various initial (and even successive) configurations of points can be considered. For instance, at the initial phase, in `Step 1` we offer the chance to start with interior grid or quasi random Halton points. The choice is of great importance in an adaptive scheme, because it defines the refinement strategy and, in practice, how the points are added or removed within the iterative process. It is thus reasonable to think that this selection influences the approximation method in terms of both computational efficiency and numerical accuracy. In what follows, we provide two alternative refinements that we will then use in our tests:

- `RefAlg-1` consists in taking the Halton points as test points starting from `Step 2`;
- `RefAlg-2` imposes to start with an initial inner node grid of stepsize h_1, selecting then in the successive phase from `Step 2` further grids whose stepsize is given by $h_j = h_{j-1}/2$, for $j = 2, 3, \ldots$.

We remark that in this second case from a computational standpoint it is convenient to iterate this process only a few times (for instance, till $j = 3$), then completing the refinement with Halton points as test points. As an example, to compare how the two refinement strategies `RefAlg-1` and `RefAlg-2` work, one can refer to Figures 5 and 7, respectively. Finally, we observe that our refinement techniques are based on the use of Halton points, but in practice the choice of other point distributions is absolutely arbitrary.

5. Numerical results

In this numerical section we analyze the algorithm performances. All the routines were implemented in MATLAB environment, and the experiments were done on a computer with processor Intel(R) Core(TM) i7-4500U CPU 1.80 GHz and RAM 4GB.

In the following we show the results, which refer to numerical solution of a few Poisson type problems via RBF-PU collocation as described in Subsection 2.3. Specifically, we present some numerical results that aim to test our adaptive refinement algorithms, which make use of estimate (18) with the related refinement strategies, as stressed in Section 4. As regards, instead, the phases of localization and search of collocation points regarding the RBF-PU collocation, details are given in Section 3.

Therefore we focus on the experiments done by using in the PU method the localized M6 approximant. As shape parameter we take the value of $\varepsilon = 3$ while the W2 function is used in (4) within the Shepard weight. Though we restrict our attention on a single RBF with fixed shape parameter, in our study we analyzed the algorithm behavior taking various localized RBF approximants and different values of ε. However, even if the choice of the RBF and the role of the shape parameter ε are known to be relevant for the accuracy of the whole approximation process (see e.g. [14]), we do not care of it because in this work our main interest is addressed to adaptivity (and not to seek the best possible accuracy). Additionally, we remark that either in applications or in particular situations in which adaptivity is required, the use of very smooth local kernels is not usually suggested since they turn out to be quite ill-conditioned [15].

In the various experiments we thus discuss the algorithm performances associated with the RBF-PU method by referring to some benchmark test problems of Poisson type, which are defined on the unit square

domain $\Omega = [0,1]^2$ (and numbered by the symbol #). The true solutions of these problems are

$$\begin{aligned}
&\#1: \quad u_1(x_1,x_2) = \sin(x_1 + 2x_2^2) - \sin(2x_1^2 + (x_2 - 0.5)^2),\\
&\#2: \quad u_2(x_1,x_2) = \frac{1}{20}\exp(4x_1)\cos(2x_1 + x_2),\\
&\#3: \quad u_3(x_1,x_2) = \frac{1}{2}x_2\left[\cos(4x_1^2 + x_2^2 - 1)\right]^4 + \frac{1}{4}x_1,\\
&\#4: \quad u_4(x_1,x_2) = \exp(-8((x_1 - 0.5)^2 + (x_2 - 0.05)^2)),\\
&\#5: \quad u_5(x_1,x_2) = \sinh(0.3(4x_1 - 4)\sin(8x_2 - 4)\exp(-(4x_1 - 2.1)^4)),\\
&\#6: \quad u_6(x_1,x_2) = \frac{1}{25(4x_1 - 2)^2 + 25(4x_2 - 2)^2 + 1}.
\end{aligned}$$

In Figure 4 we show a graphical representation of such analytic solutions. Note that Poisson problems #1–#4 are also studied in [11], while problems #5–#6 (suitably rescaled) come from [22].

To measure the precision of our numerical solutions, we report the ∞-norm or Maximum Absolute (MA) error defined by

$$\text{MA error} = \max_{1\le i\le N_e} |u(\boldsymbol{z}_i) - \hat{u}(\boldsymbol{z}_i)|,$$

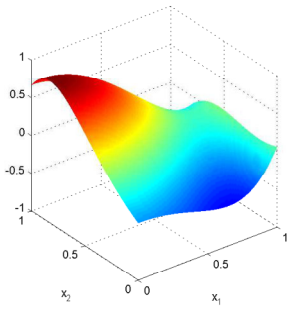
and the Root Mean Square (RMS) error

$$\text{RMS error} = \left(\frac{1}{N_e}\sum_{i=1}^{N_e} |u(\boldsymbol{z}_i) - \hat{u}(\boldsymbol{z}_i)|^2\right)^{1/2},$$

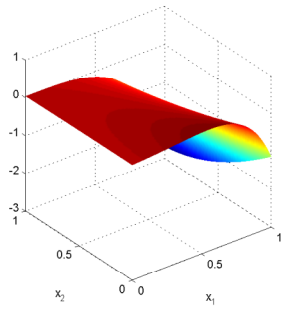
which are both computed on a set of N_e gridded evaluation points. Further, by making use of the MATLAB `condest` command we also give a Condition Number (CN) estimate of the collocation matrix Q in (16), and show the execution (or CPU) times expressed in seconds.

In the tests we start by taking $N = 121$ collocation points, consisting of $N_i = 81$ interior points and $N_b = 40$ boundary points. Precisely, we consider two different initial configurations of interior points, i.e. grid and Halton points, as depicted in Figure 3, by making use of the refinement strategy `RefAlg-1`. Further, to evaluate the adaptive refinement algorithm, we show the numerical results found by setting $(\tau_{\min}, \tau_{\max}) = (10^{-8}, 10^{-5})$ as lower and upper thresholds, respectively. At the initial phase, the iterative process begins with $N = 121$ collocation points, which consist of $N_b = 40$ grid points and $N_i = 81$: (a) grid points; (b) Halton points. In Tables 2–3 we show a summary of all results obtained, also indicating the final number N_{tot} of interior and boundary collocation points necessary to achieve the minimum/maximum tolerances. Then, in Figure 5 the final distribution of points is plotted even if here we only focus, for the sake of brevity, on the specific case of grid data as starting interior/boundary points. Moreover, always referring to the same test examples, in Figure 6 we graphically represent the absolute error $|u(\boldsymbol{z}_i) - \hat{u}(\boldsymbol{z}_i)|$, $i = 1, \ldots, N_e$, computed on the grid of $N_e = 40^2$ evaluation points, as above mentioned.

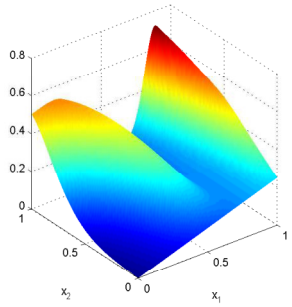
From an analysis of our tests, it is possible to remark that the adaptive algorithm allows to add points in the domain areas in which the solution highlights some significant variation. In general, taking into account the couple of tolerances used, the adaptive refinement algorithm results in getting similar MA and RMS errors for Poisson problems #1–#5, while for the elliptic problem #6 we remark the algorithm undersamples the peak present in the central area of the domain Ω. This fact reflects on the final accuracy, since error indicator and refinement process turn out to be less sensible than in the previous tests. It is however evident that – as shown in Figure 4 (bottom, right) – the last example is quite hard problem to deal with and, accordingly, high precision is difficult to get. Considering then the CN of the collocation matrix, we can see that in all problems faced it assumes a value between 10^{+6} and 10^{+9}. Such conditioning is much smaller than the one present in the commonly used RBF collocation methods (cf. [14]). Additionally, from Tables 2–3 we can observe as the adaptive algorithm converges to the solution in few seconds, except for the problem #5 in which the addition/removal process results in a less computational efficiency.



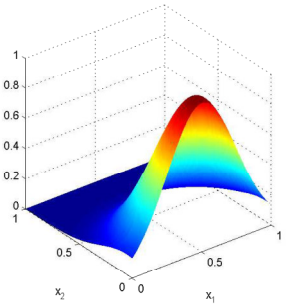
$u_1(x_1, x_2)$



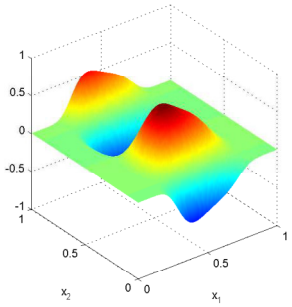
$u_2(x_1, x_2)$



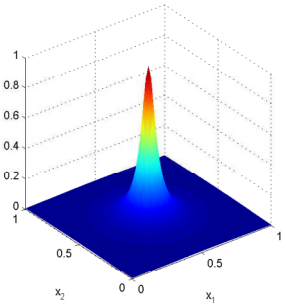
$u_3(x_1, x_2)$



$u_4(x_1, x_2)$



$u_5(x_1, x_2)$



$u_6(x_1, x_2)$

Figure 4: Exact solutions of Poisson problems.

Finally, we conclude this section showing some numerical results obtained by using the refinement strategy `RefAlg-2`, with $(\tau_{\min}, \tau_{\max}) = (10^{-8}, 10^{-5})$. As Table 4 highlights, this adaptive approach gives for Poisson problems #1–#5 quite similar results to those reported in Table 2, whereas for problem #6 we get an improvement of about one order of magnitude better in terms of accuracy. In fact, in this particularly hard situation the use of a refinement technique as `RefAlg-2`, along with a local error indicator, allows us to give a better error estimate. Figure 7 shows the final configuration of points once the adaptive refinement was completed. A visual inspection makes evident how the two refinement techniques generate quite different point distributions (cf. Figures 5 and 7).

Poisson	N_{tot}	MA error	RMS error	CN	time
#1	709	1.09e−4	3.10e−5	7.34e+06	3.5
#2	755	1.00e−4	3.46e−5	6.58e+06	4.6
#3	1411	1.58e−4	1.71e−5	4.04e+08	8.6
#4	834	4.64e−5	1.56e−5	4.04e+07	3.9
#5	2096	5.82e−5	1.73e−5	1.13e+08	13.9
#6	1197	3.44e−2	5.26e−3	1.78e+08	7.9

Table 2: Results in (a) obtained via `RefAlg-1`.

Poisson	N_{tot}	MA error	RMS error	CN	time
#1	712	7.04e−5	1.99e−5	7.59e+06	4.6
#2	746	1.21e−4	3.45e−5	9.56e+06	4.0
#3	1452	1.13e−4	1.72e−5	1.31e+08	8.8
#4	818	5.00e−5	1.49e−5	4.87e+06	4.4
#5	2094	5.48e−5	1.49e−5	8.85e+07	15.3
#6	1071	2.20e−2	2.84e−3	1.71e+07	6.2

Table 3: Results in (b) obtained via `RefAlg-1`.

Poisson	N_{tot}	MA error	RMS error	CN	time
#1	920	6.33e−5	1.64e−5	4.76e+06	3.2
#2	1003	1.66e−4	6.44e−5	1.41e+07	2.7
#3	1635	1.82e−4	2.08e−5	5.50e+07	13.1
#4	1079	5.06e−5	1.19e−5	9.12e+06	4.4
#5	2104	4.94e−5	1.57e−5	7.90e+07	13.6
#6	1265	4.96e−3	7.56e−4	1.87e+08	3.4

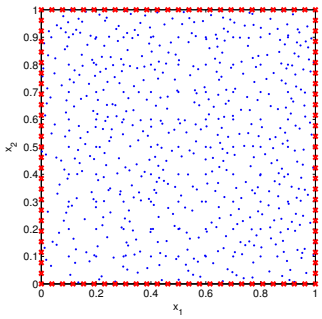
Table 4: Results obtained via `RefAlg-2`.

6. Applications

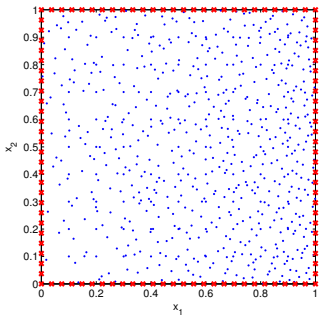
In this section we focus on two applications. Firstly, we numerically solve a Poisson problem whose solution is expressed by the first modes of a truncated series expansion; then, we deal with a Laplace problem which models the steady state temperature distribution in a thin plate.

6.1. Approximation of a truncated series

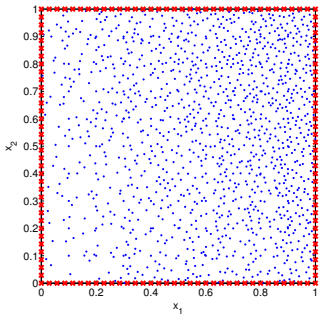
In this application we consider a Poisson problem of the form (11)–(12) defined on $\Omega = [0,1]^2$, whose solution for $j \to \infty$ becomes nonanalytic (see [22]). Here we assume as exact solution the first $p+1$ modes



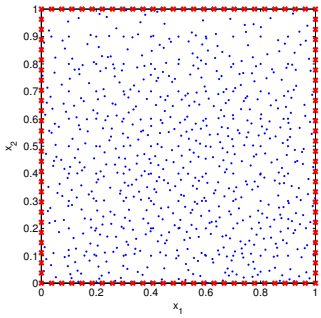
Poisson #1



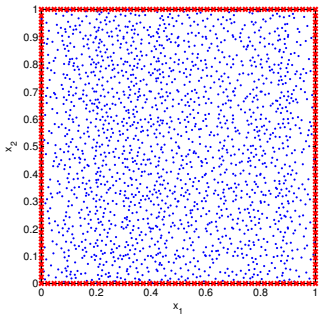
Poisson #2



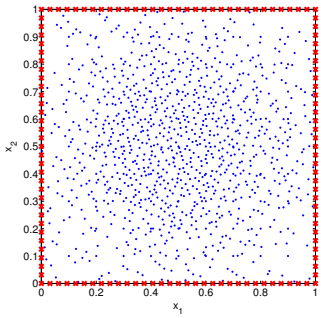
Poisson #3



Poisson #4



Poisson #5



Poisson #6

Figure 5: Final data distribution in (a) obtained via `RefAlg-1`.

of the expansion

$$u(x_1, x_2) = \sum_{j=0}^{p} \exp(-\sqrt{2^j})(\cos(2^j(4x_1 - 2)) + \cos(2^j(4x_2 - 2))).$$

In these tests we then compute the approximate solution by applying the adaptive algorithm with refinement `RefAlg-1` for $p = 1, 2, 3$, evaluating the results on a set of $N_e = 40^2$ grid points. In Figure 8 we plot solutions and Laplacians of the truncated series for various p. We thus report the collocation results in a summarizing table, where the RBF-PU method is run using a local M6 approximant with shape parameter $\varepsilon = 3$. Precisely, Table 5 shows the output of the algorithm obtained by starting from $N = 121$ grid points, composed of $N_i = 81$ interior points and $N_b = 40$ boundary points. Moreover, as in previous

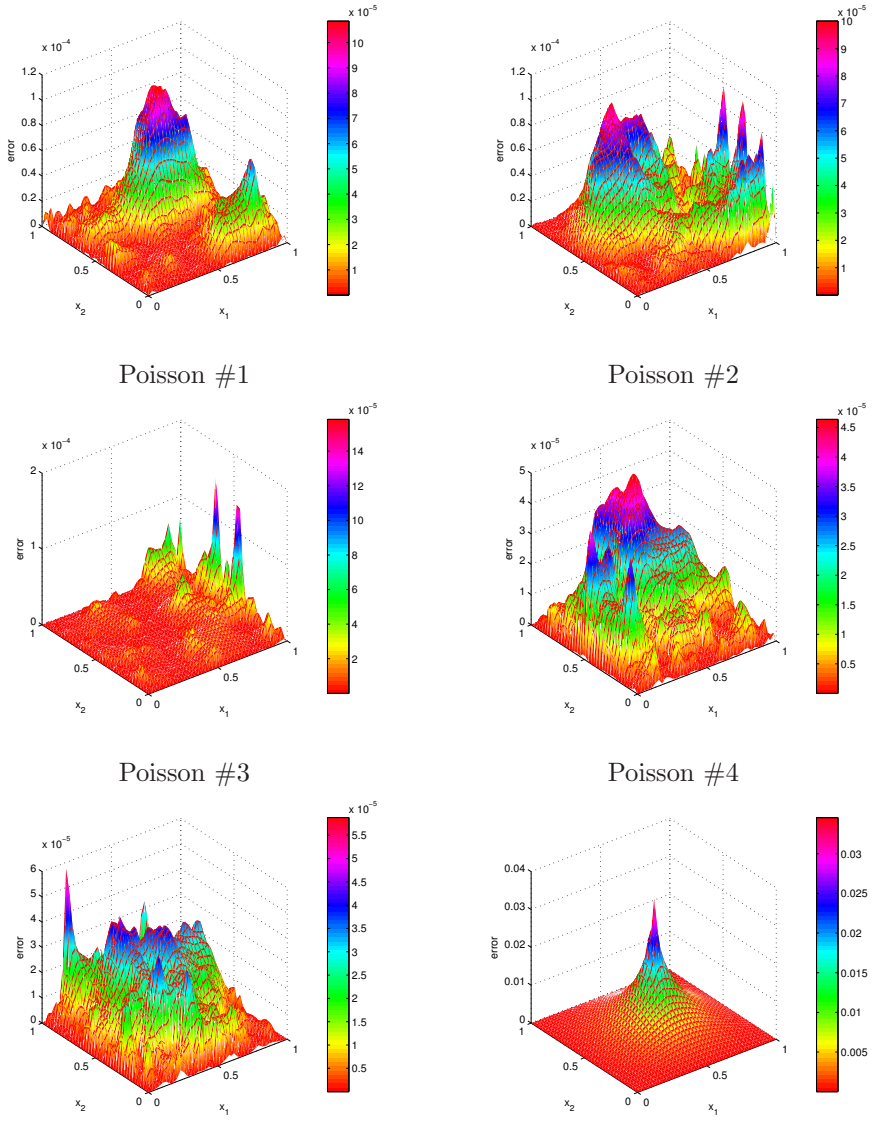
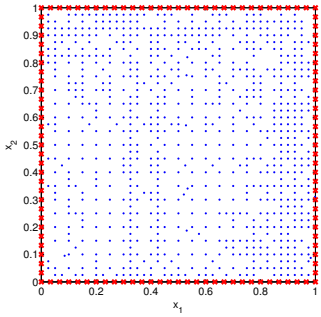
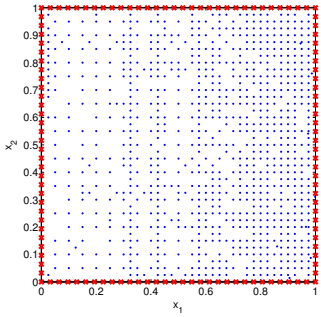


Figure 6: Absolute error $|u(x_1, x_2) - \hat{u}(x_1, x_2)|$ in (a) obtained via `RefAlg-1`.

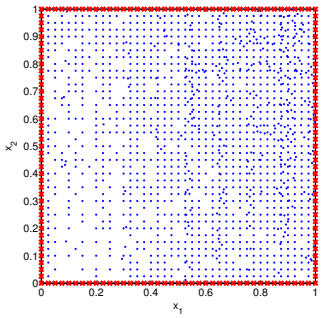
section, we focus our attention on accuracy, stability and efficiency of our collocation scheme. In doing that, we fix as tolerances in the refinement phase $(\tau_{\min}, \tau_{\max}) = (10^{-7}, 10^{-4})$. As evident from Figure 8, when the value of p increases the complexity of the problem grows. This fact happens since the solution becomes more and more difficult to be approximated and, at the same time, the Laplacian significantly increases the level of oscillations. Accordingly, growing p leads to a lager number of collocation points to get the desired thresholds. Finally, analyzing errors and conditioning, we can observe a similar behavior to that shown in the several test examples of Section 5. For brevity, we omit to report the results obtained with `RefAlg-2` as numerically similar to the previous ones.



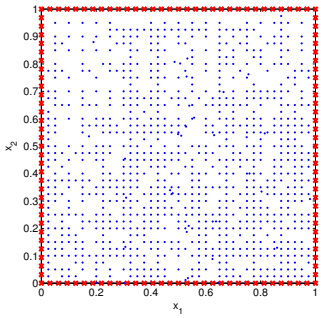
Poisson #1



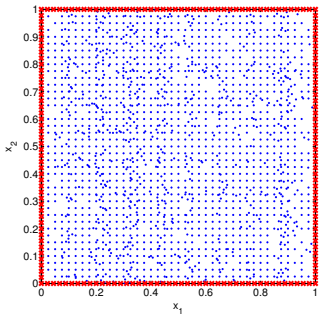
Poisson #2



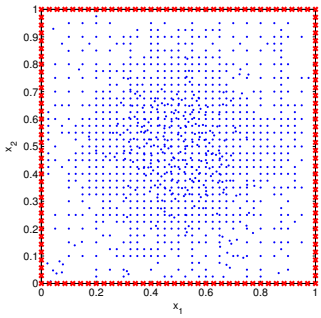
Poisson #3



Poisson #4



Poisson #5

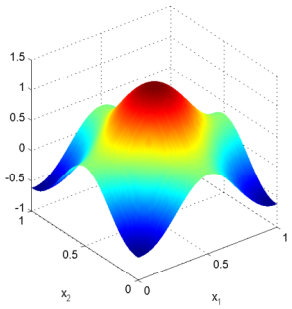


Poisson #6

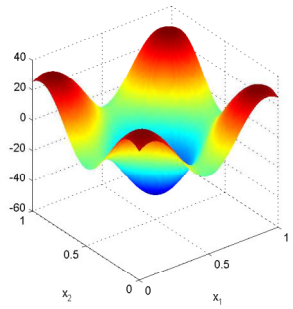
Figure 7: Final data distribution obtained via `RefAlg-2`.

p	N_{tot}	MA error	RMS error	CN	time
1	812	1.26e−3	6.36e−4	9.93e+06	5.3
2	1422	3.01e−4	1.10e−4	2.75e+07	8.2
3	2585	7.15e−4	1.75e−4	3.48e+08	17.5

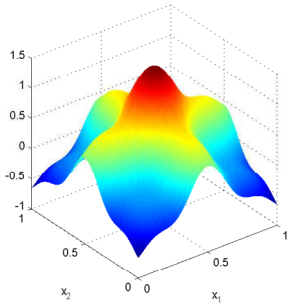
Table 5: Approximation of the truncated series by varying p via `RefAlg-1`.



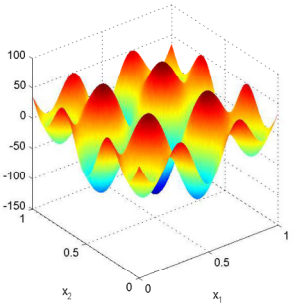
solution: $p = 1$



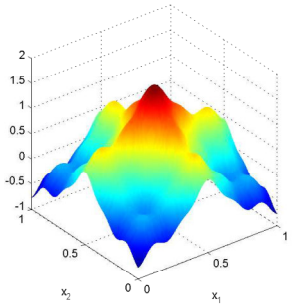
Laplacian: $p = 1$



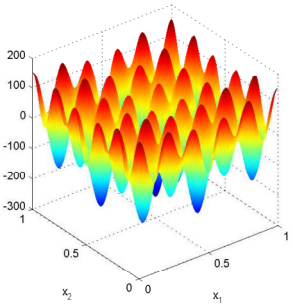
solution: $p = 2$



Laplacian: $p = 2$



solution: $p = 3$



Laplacian: $p = 3$

Figure 8: Truncated sums of exact solution (left) and Laplacian (right).

6.2. Modeling temperature distribution

In this application we consider the Laplace equation which models the steady state temperature distribution in a thin plate [1, 6]

$$\Delta u(x_1, x_2) = 0, \qquad (x_1, x_2) \in (0, 1)^2$$

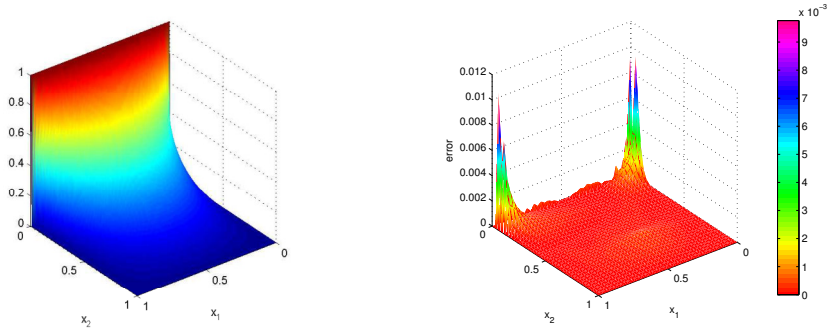
with Dirichlet boundary conditions

$$u(x_1, 0) = 1, \quad u(x_1, 1) = 0, \quad u(0, x_2) = 0, \quad u(1, x_2) = 0.$$

The analytic solution of this problem is given by

$$u(x_1, x_2) = \left(\frac{4}{\pi}\right) \sum_{i=0}^{\infty} \left\{ \frac{1}{2i+1} \sin[(2i+1)\pi x_1] \sinh[(1-x_2)(2i+1)\pi] \operatorname{csch}[(2i+1)\pi] \right\}.$$

We remark that this solution has a behavior difficult to be captured close to the boundaries, where the values of the solution are prescribed equal to zero and one by the Dirichlet conditions (see Figure 9, left). Hence, considered the difficulty in dealing with a problem of this type, we select $N = 225$ grid points, taking $N_i = 169$ interior and $N_b = 56$ boundary points, respectively. We then apply the adaptive RBF-PU method using the M6 as local approximant, with $\varepsilon = 11$. Fixed $(\tau_{\min}, \tau_{\max}) = (10^{-7}, 10^{-4})$, our algorithm with `RefAlg-2` gives us back a RMS error = 6.88e−4 selecting a number of $N_{tot} = 1546$ collocation points. Finally, Figure 9 (right) shows the absolute errors. The worst results are obtained near the boundary $x_2 = 0$ when the series goes to one very quickly, whereas in all other parts of the domain errors are very low. Note that overall such accuracy is comparable with the one achieved by the method proposed in [1] but, as evident from that work, the RBF-PU approach enables us to get much better results than RBF collocation does.



temperature distribution absolute error

Figure 9: Temperature distribution (left) and absolute error obtained by applying the adaptive scheme via `RefAlg-2` (right).

7. Conclusions and future work

In this paper we presented some adaptive refinement techniques to solve elliptic PDEs and in particular Poisson problems via a RBF-PU collocation scheme. More precisely, we proposed the use of some flexible refinement strategies that coupled with an appropriate error indicator enabled the addition and/or the removal of points. The final result was the creation of an ad-hoc approach in the RBF-PU framework. Numerical tests and applications showed the performance of our adaptive refinement algorithms.

As future work we propose to extend our adaptive algorithms modelling other types of PDE problems. This could include the numerical solution of time-independent and time-dependent differential equations. Another important aspect that deserves to be investigated concerns the design of different (or variable) cover structures for PU patches, whose size and shape could be related to the adaptive refinement scheme.

Acknowledgments

The authors thank the Department of Mathematics "Giuseppe Peano" of the University of Turin for its financial support via Project 2018 "Mathematics for applications". In addition, we acknowledge support by INdAM – GNCS Project 2019 "Kernel-based approximation, multiresolution and subdivision methods and related applications". This research has been accomplished within RITA (Rete ITaliana di Approssimazione).

References

[1] G. Allasia, A. De Rossi, Application of cardinal radial basis interpolation operators to numerical solution of the Poisson equation, Rend. Mat. Appl. 24 (2004) 281–301.

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

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

[4] I. Babuška, T. Strouboulis, The Finite Element Method and its Reliability, Oxford Univ. Press, London, 2001.

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

[6] H.S. Carslaw, J.C. Jaeger, Conduction of Heat in Solids, 2nd ed., Clarendon Press, Oxford, 1959.

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

[8] R. Cavoretto, A. De Rossi, F. Dell'Accio, F. Di Tommaso, Fast computation of triangular Shepard interpolants, J. Comput. Appl. Math. 354 (2019) 457–470.

[9] R. Cavoretto, A. De Rossi, Adaptive meshless refinement schemes for RBF-PUM collocation, Appl. Math. Lett. 90 (2019) 131–138.

[10] O. Davydov, D.T. Oanh, Adaptive meshless centres and RBF stencils for Poisson equation, J. Comput. Phys. 230 (2011) 287–304.

[11] S. De Marchi, A. Martínez, E. Perracchione, M. Rossini, RBF-based partition of unity method for elliptic PDEs: Adaptivity and stability issues via VSKs, J. Sci. Comput. 79 (2019) 321–344.

[12] T.A. Driscoll, A.R.H. Heryudono, Adaptive residual subsampling methods for radial basis function interpolation and collocation problems, Comput. Math. Appl. 53 (2007) 927–939.

[13] P. Farrell, H. Wendland, RBF multiscale collocation for second order elliptic boundary value problems, SIAM J. Numer. Anal. 51 (2013) 2403–2425.

[14] G.E. Fasshauer, Meshfree Approximation 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] G. Garmanjani, R. Cavoretto, M. Esmaeilbeigi, A RBF partition of unity collocation method based on finite difference for initial-boundary value problems, Comput. Math. Appl. 75 (2018) 4066–4090.

[17] B. Fornberg, N. Flyer, A Primer on Radial Basis Functions with Applications to the Geosciences, SIAM, Philadelphia, 2015.

[18] A. Heryudono, E. Larsson, A. Ramage, L.V. Sydow, Preconditioning for radial basis function partition of unity methods, J. Sci. Comput. 67 (2016) 1089–1109.

[19] Y.C. Hon, R. Schaback, On unsymmetric collocation by radial basis functions, Appl. Math. Comput. 119 (2001) 177–186.

[20] E.J. Kansa, Multiquadrics–A scattered data approximation scheme with applications to computational fluid-dynamics–I surface approximations and partial derivative estimates, Comput. Math. Appl. 19 (1990) 127–145.

[21] E.J. Kansa, Multiquadrics–A scattered data approximation scheme with applications to computational fluid-dynamics–II solutions to parabolic, hyperbolic and elliptic partial differential equations, Comput. Math. Appl. 19 (1990) 147–161.

[22] E. Larsson, V. Shcherbakov, A. Heryudono, A least squares radial basis function partition of unity method for solving PDEs, SIAM J. Sci. Comput. 39 (2017) A2538–A2563.

[23] 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.

[24] R. Mollapourasl, A. Fereshtian, H. Li, X. Lu, RBF-PU method for pricing options under the jump-diffusion model with local volatility, J. Comput. Appl. Math. 337 (2018) 98–118.

[25] D.T. Oanh, O. Davydov, H.X. Phu, Adaptive RBF-FD method for elliptic problems with point singularities in 2D, Appl. Math. Comput. 313 (2017) 474–497.

[26] M. Sadik, E. Hassan Ben-Ahmed, M. Wakrim, RBFPUM with QR factorization for solving water flow problem in multi-layered soil, Int. J. Nonlinear Sci. Numer. Simul. 19 (2018) 397–408.

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

[28] V. Shankar, G.B. Wright, Mesh-free semi-Lagrangian methods for transport on a sphere using radial basis functions, J. Comput. Phys. 366 (2018) 170–190.

[29] H. Wendland, Fast evaluation of radial basis functions: Methods based on partition of unity, in: Approximation Theory X: Wavelets, Splines, and Applications, C.K. Chui, L.L. Schumaker, J. Stöckler, eds., Vanderbilt Univ. Press, Nashville, TN, 2002, pp. 473–483.

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