A two-stage adaptive scheme based on RBF collocation for solving elliptic PDEs

A two-stage adaptive scheme based on RBF collocation for solving elliptic PDEs

R. Cavoretto[a,b], A. De Rossi[a,b]

[a] *Department of Mathematics "Giuseppe Peano", University of Torino, via Carlo Alberto 10, 10123 Torino, Italy*
[b] *Member of the INdAM Research group GNCS*

Abstract

In this paper we present a new adaptive two-stage algorithm for solving elliptic partial differential equations via a radial basis function collocation method. Our adaptive meshless scheme is based at first on the use of a leave-one-out cross validation technique, and then on a residual subsampling method. Each of phases is characterized by different error indicators and refinement strategies. The combination of these computational approaches allows us to detect the areas that need to be refined, also including the chance to further add or remove adaptively any points. The resulting algorithm turns out to be flexible and effective through a good interaction between error indicators and refinement procedures. Several numerical experiments support our study by illustrating the performance of our two-stage scheme. Finally, the latter is also compared with an efficient adaptive finite element method.

Keywords: meshfree approximation, adaptive algorithms, radial basis functions, collocation methods, partial differential equations
2010 MSC: 65D15, 65M70

1. Introduction

Radial basis function (RBF) methods are powerful and effective meshless techniques, which are commonly used for solving interpolation and approximation problems or modeling differential problems such as partial differential equations (PDEs). Meshless RBF methods have many remarkable advantages, since they allow us to get high convergence orders and great flexibility with respect to geometric problems. In addition, numerical algorithms are usually rather simple to implement, even when one works with complex geometries or in higher dimensions (see [2, 15, 18]).

In this paper we propose a new two-stage adaptive scheme for solving 2D elliptic PDEs. In order to test our adaptive algorithm, we consider the nonsymmetric RBF collocation method, also known as Kansa's method [24]. This collocation method was originally based on the use of multiquadric basis functions, but later it was applied more in general to different RBFs. We also observe that Kansa's method has spawned a large number of works, mainly by scientists from several different areas of science and engineering (see e.g. [1, 6, 15, 35] and references therein). Here, we construct an adaptive refinement scheme applied to RBF collocation, which consists basically of two computational phases. The first stage is dependent from a *leave-one-out cross validation* (LOOCV) method. This approach was proposed by Rippa [32] in the context of scattered data interpolation with RBFs. Nevertheless, as far as we know, Rippa's algorithm and some of its modifications have been used so far for choosing an optimal value of the RBF shape parameter (see e.g. [5, 16, 20, 33, 36]). On the contrary, here we use the LOOCV technique as an error indicator within an adaptive RBF collocation method. This choice allows us to detect which areas need to be refined by adding new discretization points. The second stage is instead based on an *adaptive residual subsampling* (ARS)

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

scheme, which is used to further refine some parts of the domain or coarsen such regions if an excessive number of points is present. In particular, we observe that in general the combination of these two phases results in an improvement of the method accuracy compared to when each method is used on its own. In the extensive numerical experiments we show the performance and the efficacy of our LOOCV-ARS algorithm, comparing it also with an efficient implementation of the adaptive finite element method, called P1-FEM [19]. Note that similar approaches consisting in the addition/removal of points can also be found in the works [3, 12]. In particular, in some previous papers we proposed the use of the RBF partition of unity (PU) collocation method [3, 4]. This approach provides in general good results but in dealing with PDE problems with solutions that present quite steep variations the RBF-PU method is not always able to achieve the expected accuracy. This fact motivated us to enhance further the current state of the art by proposing in this paper the LOOCV-ARS scheme, which is here presented for solving 2D elliptic PDE problems. However, it is important to remark that this method is not limited to two dimensions, but with suitable changes it could also be applied for solving PDE problems in higher dimensions. Finally, we observe that in literature other adaptive techniques have been studied for meshless RBF methods involving for instance weak form and finite difference methods, wavelet schemes and collocation multiscale methods (see e.g. [8, 10, 11, 13, 25, 29, 31]).

The paper is organized as follows. In Section 2 we recall the nonsymmetric Kansa's method, introducing some basic notions on the theory of RBFs. Section 3 describes the LOOCV technique applied to a RBF collocation method as an error indicator. In Section 4 we explain in detail the two stages of our adaptive refinement algorithm. In Section 5 we show numerical experiments devoted to illustrate the performance of our adaptive scheme on various types of elliptic PDE problems. Finally, Section 6 contains conclusions and future work.

2. Nonsymmetric RBF collocation

Given a domain $\Omega \subset \mathbb{R}^d$, we define a (time independent) elliptic PDE of the form

$$\mathcal{L}u(\boldsymbol{x}) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega, \tag{1}$$

with boundary conditions (BC)

$$\mathcal{B}u(\boldsymbol{x}) = g(\boldsymbol{x}), \qquad \boldsymbol{x} \in \partial\Omega, \tag{2}$$

where $\mathcal{L}$ is a linear elliptic partial differential operator and $\mathcal{B}$ is a linear boundary operator.

For Kansa's collocation method we choose to represent the approximate solution $\hat{u}$ by a RBF expansion analogous to that used in the field of RBF interpolation (see e.g. [2, 14]), i.e. $\hat{u}$ is expressed as a linear combination of basis functions

$$\hat{u}(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \phi_\varepsilon(||\boldsymbol{x} - \boldsymbol{z}_j||_2), \tag{3}$$

where c_j is an unknown real coefficient, $||\cdot||_2$ denotes the Euclidean norm, and $\phi_\varepsilon : \mathbb{R}_{\geq 0} \to \mathbb{R}$ is some RBF depending on a *shape parameter* $\varepsilon > 0$ such that

$$\phi_\varepsilon(||\boldsymbol{x} - \boldsymbol{z}||_2) = \phi(\varepsilon||\boldsymbol{x} - \boldsymbol{z}||_2), \qquad \forall\, \boldsymbol{x}, \boldsymbol{z} \in \Omega.$$

In Table 1 we list some examples of globally and compactly supported RBFs with their smoothness degrees (see [37] for details). These radial functions are commonly used for solving PDEs.

In (3) we distinguish between *centers* $Z_N = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\}$ and *collocation points* $X_N = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \Omega$. Even though such sets of points can formally be distinct, they can also coincide. As we will see in Section 4, both scenarios will be used in the construction of our adaptive refinement scheme. Further, in this discussion the set X_N is split into a set X_{N_I} of interior points and a set X_{N_B} of boundary points, so that $X_N = X_{N_I} \cup X_{N_B}$, where N_I and N_B denote the number of interior and boundary points, respectively.

RBF	$\phi_\varepsilon(r)$
Gaussian C^∞ (GA)	$\exp(-\varepsilon^2 r^2)$
Inverse MultiQuadric C^∞ (IMQ)	$(1+\varepsilon^2 r^2)^{-1/2}$
MultiQuadric C^∞ (MQ)	$(1+\varepsilon^2 r^2)^{1/2}$
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^6 (W6)	$\max(1-\varepsilon r, 0)^8 (32\varepsilon^3 r^3 + 25\varepsilon^2 r^2 + 8\varepsilon r + 1)$
Wendland C^4 (W4)	$\max(1-\varepsilon r, 0)^6 (35\varepsilon^2 r^2 + 18\varepsilon r + 3)$

Table 1: Some examples of commonly used RBFs.

Matching the PDE (1) and the BCs (2) at the collocation points X_N, we get

$$\begin{aligned} \mathcal{L}\hat{u}(\boldsymbol{x}_i) &= \sum_{j=1}^{N} c_j \mathcal{L}\phi_\varepsilon(||\boldsymbol{x}_i - \boldsymbol{z}_j||_2) = f(\boldsymbol{x}_i), \qquad \boldsymbol{x}_i \in \Omega, \\ \mathcal{B}\hat{u}(\boldsymbol{x}_i) &= \sum_{j=1}^{N} c_j \mathcal{B}\phi_\varepsilon(||\boldsymbol{x}_i - \boldsymbol{z}_j||_2) = g(\boldsymbol{x}_i), \qquad \boldsymbol{x}_i \in \partial\Omega, \end{aligned} \tag{4}$$

resulting in a linear system of equations

$$A\boldsymbol{c} = \boldsymbol{v}, \tag{5}$$

where $\boldsymbol{v}$ is the vector of entries

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

and A is the collocation matrix

$$A = \begin{bmatrix} A_{\mathcal{L}} \\ A_{\mathcal{B}} \end{bmatrix}. \tag{6}$$

The two blocks in (6) are thus defined as

$$\begin{aligned} (A_{\mathcal{L}})_{ij} &= \mathcal{L}\phi_\varepsilon(||\boldsymbol{x}_i - \boldsymbol{z}_j||_2), \qquad \boldsymbol{x}_i \in X_{N_I},\ \boldsymbol{z}_j \in Z_N, \\ (A_{\mathcal{B}})_{ij} &= \mathcal{B}\phi_\varepsilon(||\boldsymbol{x}_i - \boldsymbol{z}_j||_2), \qquad \boldsymbol{x}_i \in X_{N_B},\ \boldsymbol{z}_j \in Z_N. \end{aligned} \tag{7}$$

Since the collocation matrix (6) may be singular for certain (rare) configurations of the centers $\boldsymbol{z}_j$, it follows that the nonsymmetric collocation method cannot be well-posed for arbitrary center locations (see e.g. [14, 23]). However, it is possible to find sufficient conditions on the centers so that invertibility of Kansa's matrix is ensured. Hence, based on theoretical convergence analysis of Kansa's collocation method (and some variations thereof derived from applications) in [28, 34], this nonsymmetric approach has gained much popularity in spite of its potential failure when applied in its most straightforward and naive implementation [23].

3. LOOCV strategy for RBF collocation

A common strategy for selecting an optimal RBF shape parameter is to use a cross validation approach [7, 21]. The formula and the associated algorithm proposed by Rippa in [32] corresponds to one of its possible variants. The LOOCV technique is rather popular in statistical literature and, recently, it has also been

successful in the field of RBF approximation (see e.g. [5, 15, 33, 36, 39]). However, while in previous works the main focus is to find an optimal value of ε, here we apply the LOOCV strategy as an error indicator within an adaptive RBF collocation scheme. Indeed, as we will see in Section 4, this approach enables us to identify in which area of the domain we have to add some points by applying a local refinement. Notice that in this case the shape parameter ε is kept fixed.

3.1. Description of LOOCV

The idea behind LOOCV for the nonsymmetric RBF collocation can thus be described as follows. First of all, we split the data into two different sets:

- a *training set* $\{v(\boldsymbol{x}_1), \dots, v(\boldsymbol{x}_{k-1}), v(\boldsymbol{x}_{k+1}), \dots, v(\boldsymbol{x}_N)\}$,
- a *validation set* consisting of only the single value $v(\boldsymbol{x}_k)$ which was left out when creating the training set.

For a given $k \in \{1, \dots, N\}$ and fixed ε, we define the partial RBF approximant

$$\hat{u}^{[k]}(\boldsymbol{x}) = \sum_{j=1,\ j\neq k}^{N} c_j^{[k]} \phi_\varepsilon(||\boldsymbol{x} - \boldsymbol{z}_j||_2),$$

whose coefficients $c_j^{[k]}$ are determined by collocating the training data, i.e.

$$\begin{aligned} \mathcal{L}\hat{u}^{[k]}(\boldsymbol{x}_i) &= v(\boldsymbol{x}_i), \qquad && \boldsymbol{x}_i \in X_{N_I}, \\ \mathcal{B}\hat{u}^{[k]}(\boldsymbol{x}_i) &= v(\boldsymbol{x}_i), && \boldsymbol{x}_i \in X_{N_B}, \end{aligned} \qquad \text{for } i = 1, \dots, k-1, k+1, \dots, N.$$

This is equivalent to solving the $(N-1) \times (N-1)$ linear system

$$A^{[k]} \boldsymbol{c}^{[k]} = \boldsymbol{v}^{[k]}, \tag{8}$$

where $A^{[k]}$ is obtained from A by removing the k-th row and the k-th column, while $\boldsymbol{c}^{[k]} = (c_1^{[k]}, \dots, c_{k-1}^{[k]}, c_{k+1}^{[k]}, \dots, c_N^{[k]})^T$ and $\boldsymbol{v}^{[k]} = (v_1^{[k]}, \dots, v_{k-1}^{[k]}, v_{k+1}^{[k]}, \dots, v_N^{[k]})^T$.

Then, to have a measure of the quality of this attempt, we define the absolute error

$$e_k = |v(\boldsymbol{x}_k) - \hat{u}^{[k]}(\boldsymbol{x}_k)|, \tag{9}$$

at the one validation point $\boldsymbol{x}_k$ not used to determine the approximant. Hence, computing the error in (9), for all $k = 1, \dots, N$, we obtain a vector $\boldsymbol{e} = (e_1, \dots, e_N)^T$, which can be used as an error indicator to identify the regions that need a refinement, i.e. the addition of points, in the neighborhood of a collocation point.

However, the computational cost to find the error term (9) requires $\mathcal{O}(N^4)$ operations, which is quite expensive even for problems of modest size.

3.2. Construction of LOOCV as an error indicator

In order to reduce the complexity to $\mathcal{O}(N^3)$, a formula for computing the term e_k is derived to significantly speep up the computation process. This approach has originally been developed for the interpolation [32], while here we deal with the case of RBF collocation as follows:

i) If $\boldsymbol{t} = (t_1, \dots, t_N)^T \in \mathbb{R}^N$ is such that $t_k = 0$, then we have $A\boldsymbol{t} = \boldsymbol{w}$ for some $\boldsymbol{w}$, which implies that

$$A^{[k]}(t_1, \dots, t_{k-1}, t_{k+1}, \dots, t_N)^T = (w_1, \dots, w_{k-1}, w_{k+1}, \dots, w_N)^T. \tag{10}$$

ii) Denoting by $\boldsymbol{x}^{(k)}$ the solution to the system

$$A\boldsymbol{x}^{(k)} = \boldsymbol{e}^{(k)}, \tag{11}$$

where $\boldsymbol{e}^{(k)}$ is the k-th column of the $N \times N$ identity matrix, we deduce that $x_k^{(k)} \neq 0$. In fact, if $\boldsymbol{x}_k^{(k)} = 0$, from (10) and (11) we get

$$A^{[k]}(x_1^{(k)}, \ldots, x_{k-1}^{(k)}, x_{k+1}^{(k)}, \ldots, x_N^{(k)})^T = \mathbf{0}. \tag{12}$$

This leads to $\boldsymbol{x}^{(k)} = \mathbf{0}$, which is impossible since $x_k^{(k)}$ is the solution of (11).

iii) Defining the vector $\boldsymbol{b}^{(k)} \in \mathbb{R}^N$ such that

$$\boldsymbol{b}^{(k)} = \boldsymbol{c} - \frac{c_k}{x_k^{(k)}}\boldsymbol{x}^{(k)},$$

we obtain by (11)

$$A\boldsymbol{b}^{(k)} = A\boldsymbol{c} - \frac{c_k}{x_k^{(k)}}A\boldsymbol{x}^{(k)} = \boldsymbol{v} - \frac{c_k}{x_k^{(k)}}\boldsymbol{e}^{(k)} = \left(v_1, \ldots, v_{k-1}, v_k - \frac{c_k}{x_k^{(k)}}, v_{k+1}, \ldots, v_N\right)^T,$$

and, since $b_k^{(k)} = 0$, we use (10) once again to get

$$\boldsymbol{c}^{[k]} = (b_1^{(k)}, \ldots, b_{k-1}^{(k)}, b_{k+1}^{(k)}, \ldots, , b_N^{(k)})^T.$$

iv) Accordingly, we conclude that

$$\begin{aligned}\hat{u}^{[k]}(\boldsymbol{x}_k) &= \sum_{j=1,\ j\neq k}^{N} c_j^{[k]}\phi_\varepsilon(||\boldsymbol{x}_k - \boldsymbol{z}_j||_2) = \sum_{j=1,\ j\neq k}^{N} b_j^{(k)}\phi_\varepsilon(||\boldsymbol{x}_k - \boldsymbol{z}_j||_2) \\ &= \sum_{j=1}^{N} b_j^{(k)}\phi_\varepsilon(||\boldsymbol{x}_k - \boldsymbol{z}_j||_2) = (A\boldsymbol{b}^{(k)})_k = v_k - \frac{c_k}{x_k^{(k)}},\end{aligned}$$

v) The absolute error term e_k in (9) is thus given by

$$e_k = |v_k - \hat{u}^{[k]}(\boldsymbol{x}_k)| = \left|\frac{c_k}{x_k^{(k)}}\right|. \tag{13}$$

Therefore, instead of solving N collocation problems by inverting the matrix $A^{[k]}$ given in (8), it is evident that we can determine more easily the error component e_k. Finally, the rule (13) shows us that the computation of e_k can simply be rewritten as

$$e_k = \left|\frac{c_k}{A_{kk}^{-1}}\right|, \qquad k = 1, \ldots, N, \tag{14}$$

where c_k is the k-th coefficient of the full approximate solution (3) and A_{kk}^{-1} is the k-th diagonal element of the inverse of the corresponding $N \times N$ collocation matrix A in (6). Note that the term $x_k^{(k)}$ in (13) is exactly the diagonal element A_{kk}^{-1} in (14).

4. Two-stage adaptive scheme

In this section we describe in detail our new adaptive algorithm, which is based on two computational phases. The first stage consists in refining the areas of the domain Ω that have been identified to be not accurate enough by applying the LOOCV as an error indicator; the second stage is a further adaptive technique where the computational procedure based on ARS provides an error estimate, which allows us to refine or coarsen the distribution of collocation points.

4.1. Stage 1: Refinement technique based on LOOCV

First of all, we consider two initial sets $X^1_{N^{(1)}} \equiv X_N = \{\boldsymbol{x}_1^{(1)}, \ldots, \boldsymbol{x}_{N^{(1)}}^{(1)}\}$ and $Z^1_{N^{(1)}} \equiv Z_N = \{\boldsymbol{z}_1^{(1)}, \ldots, \boldsymbol{z}_{N^{(1)}}^{(1)}\}$ of collocation points and centers, respectively. Each of these sets are composed of two subsets which, in an adaptive scheme, for $j = 1, 2, \ldots$, can iteratively be defined as follows:

- $X^1_{N_I^{(j)}} = \{\boldsymbol{x}_1^{(j)}, \ldots, \boldsymbol{x}_{N_I^{(j)}}^{(j)}\}$ and $X^1_{N_B^{(j)}} = \{\boldsymbol{x}_1^{(j)}, \ldots, \boldsymbol{x}_{N_B^{(j)}}^{(j)}\}$

  are sets of interior and boundary collocation points, respectively;

- $Z^1_{N_I^{(j)}} = \{\boldsymbol{z}_1^{(j)}, \ldots, \boldsymbol{z}_{N_I^{(j)}}^{(j)}\}$ and $Z^1_{N_B^{(j)}} = \{\boldsymbol{z}_1^{(j)}, \ldots, \boldsymbol{z}_{N_B^{(j)}}^{(j)}\}$

  are sets of interior and additional boundary centers, respectively.

Here, we assume that $Z^1_{N_I^{(j)}} = X^1_{N_I^{(j)}}$ while the set $Z^1_{N_B^{(j)}}$ of centers is taken outside the domain Ω as suggested in [17] (see also [14, Chapter 39]). Note that using this technique the number of external centers, whose we refer to as additional boundary centers, is the same as boundary collocation points. For example, when the domain is given by $\Omega = [0, 1]^2$, we assume

$$N_B^{(j)} = 4 \left\lceil \sqrt{N_I^{(j)}} + 2 \right\rceil - 4.$$

This choice provides a uniform distribution of interior and boundary points, supposing the iterative algorithm starts from gridded points. Since we are designing an adaptive algorithm, it turns out to be particularly convenient to begin with a grid of interior and boundary points. This allows us to uniformly cover the domain Ω detecting in a better way which areas of Ω have to be refined. However, other choices of points could be considered. Note that in the above-mentioned sets the symbol ^{1} refers to the first stage of the adaptive scheme, while the index $_{N^{(j)}}$ denotes the number of points in the j-th iteration of that same stage.

At the first iteration, fixed a tolerance $\tau > 0$, from (14) we can define an error indicator via LOOCV as

$$e_k^{(1)} = \left| \frac{c_k}{A^{-1}_{kk}} \right|, \qquad k = 1, \ldots, N^{(1)}, \tag{15}$$

where in the absolute value argument any reference to the iteration is omitted to avoid confusion in the notation. If the error indicator (15) is such that $e_k^{(1)} > \tau$, then a refinement is applied in the neighborhood of $\boldsymbol{x}_k$ (and accordingly $\boldsymbol{z}_k$). In Figure 1 (top) we show some possible refinement strategies that consist in adding four or eight points located in different positions. However, when the point $\boldsymbol{x}_k$ is on or close to the boundary, the number of involved points is obviously reduced, as shown in Figure 1 (bottom). In doing that, to refine the data point distribution, we first compute the so-called *separation distance*

$$q_{X^1_{N^{(1)}}} = \frac{1}{2} \min_{u \neq v} ||\boldsymbol{x}_u^{(1)} - \boldsymbol{x}_v^{(1)}||_2, \qquad \boldsymbol{x}_u^{(1)}, \boldsymbol{x}_v^{(1)} \in X^1_{N^{(1)}}, \tag{16}$$

and then we sum up or subtract the quantity in (16) to one (both) coordinate(s) of the point $\boldsymbol{x}_k$. Notice that here $\boldsymbol{x}_k$ represents the point to be refined and it is marked by a blue dot in Figure 1, whereas the four/eight points in red denote the added ones. In particular, we remark that the refinement strategy is applied to both sets $X^1_{N^{(1)}}$ and $Z^1_{N^{(1)}}$.

Acting in this way, we can iterate the adaptive process for $j = 2, 3, \ldots$, thus determining new refined sets $X^1_{N^{(j)}}$ and $Z^1_{N^{(j)}}$ of collocation points and centers, respectively. These sets are then validated by simply updating the indicator (15) as

$$e_k^{(j)} = \left| \frac{c_k}{A^{-1}_{kk}} \right|, \qquad k = 1, \ldots, N^{(j)}, \tag{17}$$

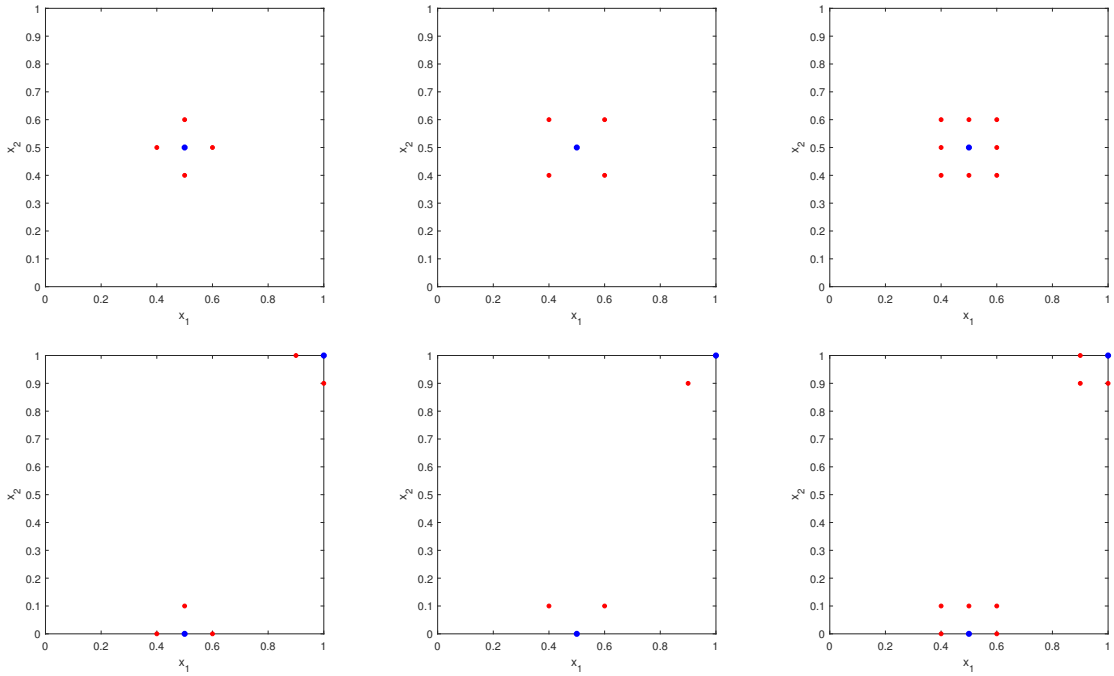


Figure 1: Examples of possible refinement strategies applied in the neighborhood of an interior (top) or a boundary (bottom) point $\boldsymbol{x}_k$ (blue dots) by adding points in different ways (red dots) when the error indicator $e_k^{(j)} > \tau$, with $j = 1, 2, \ldots$ and tolerance $\tau > 0$.

and accordingly renewing the value of the separation distance in (16) for the set $X^1_{N^{(j)}}$, for $j = 2, 3, \ldots$. Here, it is relevant to stress that the computation of vector $\boldsymbol{e}^{(j)} = \{\boldsymbol{e}_1^{(j)}, \ldots, \boldsymbol{e}_{N^{(j)}}^{(j)}\}$ requires the inversion of a matrix of the form (6) after each refinement (or iteration). The refinement procedure stops when all components of error terms in (17) are lower or equal to the tolerance τ. Finally, the computational procedure returns the sets $X^1_{N^{(j^*)}}$ and $Z^1_{N^{(j^*)}}$, where j^* denotes the final iteration of this first stage. A pseudo-code of this first phase is outlined in Algorithm 1.

4.2. Stage 2: Computational procedure based on ARS

In this second phase of the adaptive algorithm we present a computational technique that enables us to carry out a further refinement, adding or removing collocation points based on the information derived from a new error indicator. Albeit differently devised, the original idea of this adaptive meshless refinement is based on the residual subsampling method proposed in [12], which has suitably been modified for RBF-PU collocation [3, 9]. Here we further extend this procedure by applying a different error indicator. This process follows the common paradigm to solve, estimate and refine/coarsen till a stopping criterion is satisfied. Moreover, we observe that the ARS scheme can be used to solve PDE problems in science and engineering applications, since unlike [9] our indicator does not depend on information coming from the exact solution that in real-life situations is often not available.

In this stage, we begin by taking the initial sets $X^2_{N^{(1)}}$ and $Z^2_{N^{(1)}}$ obtained from the output of Algorithm 1, i.e. assuming that $X^2_{N^{(1)}} = X^1_{N^{(j^*)}}$ and $Z^2_{N^{(1)}} = Z^1_{N^{(j^*)}}$, where $X^1_{N^{(j^*)}} = X^1_{N_I^{(j^*)}} \cup X^1_{N_B^{(j^*)}}$ and $Z^1_{N^{(j^*)}} = Z^1_{N_I^{(j^*)}} \cup Z^1_{N_B^{(j^*)}}$. In terms of notation, the symbol ^{2} in the sets $X^2_{N^{(1)}}$ and $Z^2_{N^{(1)}}$ indicates the second stage of the adaptive process, whereas the index $_{N^{(1)}}$ is the number of points in the first iteration. Then we compute the two collocation solutions $\hat{u}$ and $\hat{u}^A$ of the form (3). More precisely, the two RBF approximants $\hat{u}$ and $\hat{u}^A$ are obtained by solving two linear systems of the form (5). In particular, the first system

Algorithm 1: LOOCV based refinement procedure

STEP 1	Generate two sets $X^1_{N^{(1)}}$ and $Z^1_{N^{(1)}}$ of (interior and boundary) collocation points and centers, assuming $X^1_{N^{(1)}} = X_N$ and $Z^1_{N^{(1)}} = Z_N$
STEP 2	Fix a tolerance $\tau > 0$
STEP 3	Compute the error term $e_k^{(1)}$ in (15) via LOOCV and use it as an error indicator
STEP 4	If the error indicator (15) is such that $e_k^{(1)} > \tau$, for any k then a refinement is applied in the neighborhood of $\boldsymbol{x}_k$ as shown in Figure 1 else return $X^1_{N^{(1)}}$ and $Z^1_{N^{(1)}}$
STEP 5	Iterate the STEPS 3-4, defining new sets $X^1_{N^{(j)}}$ and $Z^1_{N^{(j)}}$ till $e_k^{(j)} > \tau$, for any k where $j = 2, 3, \ldots$ denotes the iteration
STEP 6	Return the refined sets $X^1_{N^{(j^*)}}$ and $Z^1_{N^{(j^*)}}$ (j^* is the last iteration)

associated with $\hat{u}$ is constructed on coincident sets of collocation points and centers. In other words, the sets $X^2_{N^{(1)}} = X^2_{N_I^{(1)}} \cup X^2_{N_B^{(1)}}$ of collocation points and $Z^2_{N^{(1)}} = Z^2_{N_I^{(1)}} \cup Z^2_{N_B^{(1)}}$ of centers are such that $Z^2_{N^{(1)}} = X^2_{N^{(1)}}$. Thus, the collocation matrix A in (6) is defined as

$$A = \begin{bmatrix} A_{\mathcal{L}} \\ A_{\mathcal{B}} \end{bmatrix} = \begin{bmatrix} A_{\mathcal{L}_{II}} & A_{\mathcal{L}_{IB}} \\ A_{\mathcal{B}_{BI}} & A_{\mathcal{B}_{BB}} \end{bmatrix}, \tag{18}$$

where

$$\begin{aligned}
(A_{\mathcal{L}_{II}})_{ij} &= \mathcal{L}\phi_\varepsilon(||\boldsymbol{x}_i^{(1)} - \boldsymbol{z}_j^{(1)}||_2), && \boldsymbol{x}_i^{(1)} \in X^2_{N_I^{(1)}},\ \boldsymbol{z}_j^{(1)} \in Z^2_{N_I^{(1)}}, \\
(A_{\mathcal{L}_{IB}})_{ij} &= \mathcal{L}\phi_\varepsilon(||\boldsymbol{x}_i^{(1)} - \boldsymbol{z}_j^{(1)}||_2), && \boldsymbol{x}_i^{(1)} \in X^2_{N_I^{(1)}},\ \boldsymbol{z}_j^{(1)} \in Z^2_{N_B^{(1)}}, \\
(A_{\mathcal{B}_{BI}})_{ij} &= \mathcal{B}\phi_\varepsilon(||\boldsymbol{x}_i^{(1)} - \boldsymbol{z}_j^{(1)}||_2), && \boldsymbol{x}_i^{(1)} \in X^2_{N_B^{(1)}},\ \boldsymbol{z}_j^{(1)} \in Z^2_{N_I^{(1)}} \\
(A_{\mathcal{B}_{BB}})_{ij} &= \mathcal{B}\phi_\varepsilon(||\boldsymbol{x}_i^{(1)} - \boldsymbol{z}_j^{(1)}||_2), && \boldsymbol{x}_i^{(1)} \in X^2_{N_B^{(1)}},\ \boldsymbol{z}_j^{(1)} \in Z^2_{N_B^{(1)}}.
\end{aligned} \tag{19}$$

The second system associated with $\hat{u}^A$ is characterized by the same set $X^2_{N^{(1)}} = X^2_{N_I^{(1)}} \cup X^2_{N_B^{(1)}}$ of collocation points as the first system while the set of centers is split as $Z^2_{N^{(1)}} = Z^2_{N_I^{(1)}} \cup Z^{2,A}_{N_B^{(1)}}$, with $Z^{2,A}_{N_B^{(1)}}$ defining additional boundary centers taken outside the domain as in Stage 1. It follows that the resulting collocation matrix assumes the same form as in (18). However, while the matrix blocks $(A_{\mathcal{L}_{II}})_{ij}$ and $(A_{\mathcal{B}_{BI}})_{ij}$ are defined as in (19), the two remaining blocks $(A_{\mathcal{L}_{IB}})_{ij}$ and $(A_{\mathcal{B}_{BB}})_{ij}$ are given by

$$\begin{aligned}
(A_{\mathcal{L}_{IB}})_{ij} &= \mathcal{L}\phi_\varepsilon(||\boldsymbol{x}_i^{(1)} - \boldsymbol{z}_j^{(1)}||_2), && \boldsymbol{x}_i^{(1)} \in X^2_{N_I^{(1)}},\ \boldsymbol{z}_j^{(1)} \in Z^{2,A}_{N_B^{(1)}}, \\
(A_{\mathcal{B}_{BB}})_{ij} &= \mathcal{B}\phi_\varepsilon(||\boldsymbol{x}_i^{(1)} - \boldsymbol{z}_j^{(1)}||_2), && \boldsymbol{x}_i^{(1)} \in X^2_{N_B^{(1)}},\ \boldsymbol{z}_j^{(1)} \in Z^{2,A}_{N_B^{(1)}}.
\end{aligned}$$

Therefore, we observe that the solutions $\hat{u}$ and $\hat{u}^A$ are computed by considering the same sets of collocation points, whereas the sets of centers differ. Indeed, though the interior collocation points and interior centers coincide for both systems, in the first system we take a set $Z^2_{N_B^{(1)}}$ of boundary centers located on the boundary of Ω while in the second one we consider a set $Z^{2,A}_{N_B^{(1)}}$ of (additional) boundary centers lying

outside the domain. However, note that the number of additional boundary points used to compute $\hat{u}^A$ is the same as boundary ones considered to determine $\hat{u}$. In Figure 2 we show an example of these two sets of centers, which are used for finding the solutions $\hat{u}$ and $\hat{u}^A$. Both sets are overlapped with the corresponding sets of collocation points. It refers to the wave front problem that we will discuss later in Subsection 5.2.

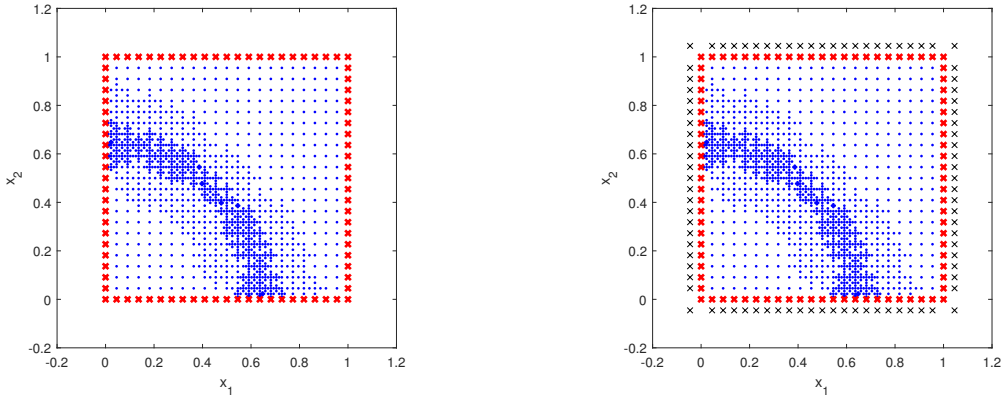


Figure 2: Example of a set $Z^2_{N^{(1)}} = Z^2_{N_I^{(1)}} \cup Z^2_{N_B^{(1)}}$ of interior and boundary centers (left) and a set $Z^2_{N^{(1)}} = Z^2_{N_I^{(1)}} \cup Z^{2,A}_{N_B^{(1)}}$ of interior and additional boundary centers outside the domain (right). Both sets are overlapped with the related sets of collocation points. Interior points are blue dots, boundary points are red crosses and additional boundary points are black crosses.

After defining a set $Y^{(1)} = \{\boldsymbol{y}_1^{(1)}, \ldots, \boldsymbol{y}_{N^{(1)}}^{(1)}\}$ of test points, we can thus evaluate the error via the indicator

$$E_i^{(1)} = |\hat{u}(\boldsymbol{y}_i^{(1)}) - \hat{u}^A(\boldsymbol{y}_i^{(1)})|, \qquad \boldsymbol{y}_i^{(1)} \in Y^{(1)}. \tag{20}$$

At the moment we fix two positive tolerances (or thresholds) τ_{low} and τ_{upp}, such that $\tau_{low} < \tau_{upp}$. Such thresholds allow us to refine the set of collocation points when the error indicator (20) does not predict an accurate enough result, or coarsen the set of collocation points if the level of precision achieved is excessive. In brief, this process results in an addition or removal of points, thus making this second phase of the scheme adaptive. Acting in this way, we can therefore define the sets $X^2_{N^{(2)}}$ and $Z^2_{N^{(2)}}$, and then iteratively the successive ones. Further details on the manner of addition/removal are given in Step 6 of Algorithm 2. The whole technique can then be iterated for $k = 2, 3, \ldots$ by updating the set $Y^{(k)} = \{\boldsymbol{y}_1^{(k)}, \ldots, \boldsymbol{y}_{N^{(k)}}^{(k)}\}$ of test points and validating the results through the error indicator defined by

$$E_i^{(k)} = |\hat{u}(\boldsymbol{y}_i^{(k)}) - \hat{u}^A(\boldsymbol{y}_i^{(k)})|, \qquad \boldsymbol{y}_i^{(k)} \in Y^{(k)}, \tag{21}$$

where $\hat{u}$ is the collocation solution computed on the set $Z^2_{N^{(k)}} = Z^2_{N_I^{(k)}} \cup Z^2_{N_B^{(k)}}$ of interior and boundary centers, and $\hat{u}^A$ is the collocation solution computed on the set $Z^2_{N^{(k)}} = Z^2_{N_I^{(k)}} \cup Z^{2,A}_{N_B^{(k)}}$ of interior centers with additional boundary points outside the domain Ω. The ARS concludes once the process of addition and/or removal was completed, returning the final sets $X^2_{N^{(k^*)}}$ and $Z^2_{N^{(k^*)}}$ (associated with $\hat{u}^A$), where k^* denotes the last iteration of the second phase. A pseudo-code of this second stage is sketched in Algorithm 2.

In Section 5 we will show how the two-stage adaptive scheme works. As an example, in order to provide some final configurations of discretization points returned by the algorithm, we can refer to Figure 4 and Figures 5–7 (right) contained in the next numerical section.

Remark 4.1. *Note that in general in a refinement strategy it could happen that duplicated points are included. However, in our technique a check is done in order to exclude multiple points.*

Algorithm 2: ARS based procedure

STEP 1	Take the sets $X^1_{N^{(j^*)}}$ and $Z^1_{N^{(j^*)}}$ of (interior and boundary) collocation points and centers from Algorithm 1, assuming $X^2_{N^{(1)}} = X^1_{N^{(j^*)}}$ and $Z^2_{N^{(1)}} = Z^1_{N^{(j^*)}}$
STEP 2	Fix two tolerances τ_{low} and τ_{upp}, such that $0 < \tau_{low} < \tau_{upp}$
STEP 3	Compute the approximate collocation solutions $\hat{u}$ and $\hat{u}^A$
STEP 4	Define a set $Y^{(1)}$ of test points
STEP 5	Evaluate at the test points the error via indicator (20)
STEP 6	Construct the refined set $X^2_{N^{(2)}} = X^2_{N_I^{(2)}} \cup X^2_{N_B^{(2)}}$, with $X^2_{N_I^{(2)}} = (X^2_{N_I^{(1)}} \cup S_{T^{(1)}_{upp}}) \backslash S_{T^{(1)}_{low}}$ where $S_{T^{(1)}_{upp}} = \{\boldsymbol{y}_i^{(1)} \in Y^{(1)} : E_i^{(1)} > \tau_{upp},\, i = 1, \ldots, T^{(1)}_{upp}\}$ $S_{T^{(1)}_{low}} = \{\bar{\boldsymbol{x}}_i^{(1)} \in X^2_{N^{(1)}} : E_i^{(1)} < \tau_{low},\, i = 1, \ldots, T^{(1)}_{low}\}$ and $\bar{\boldsymbol{x}}_i^{(1)}$ is the nearest point to $\boldsymbol{y}_i^{(1)}$; then, update the set $Z^2_{N^{(2)}}$
STEP 7	Repeat the STEPS 3-6, and for $k = 2, 3, \ldots$ define the new sets $Y^{(k)}$, $X^2_{N^{(k+1)}}$, $Z^2_{N^{(k+1)}}$, $S_{T^{(k)}_{upp}}$ and $S_{T^{(k)}_{low}}$ by using the error indicator (21)
STEP 8	Stop when the set $S_{T^{(k)}_{low}}$ is empty
STEP 9	Return the final sets $X^2_{N^{(k^*)}}$ and $Z^2_{N^{(k^*)}}$ associated with $\hat{u}^A$ (k^* is the last iteration)

Remark 4.2. *In this work we present an adaptive two-stage scheme, called LOOCV-ARS, which consists in the combination of LOOCV (Stage 1) and ARS (Stage 2). It is however possible that both adaptive refinement schemes can be used independently. Basically, the first stage is particularly useful to detect the areas that need to be refined via LOOCV indicator, while the second stage allows us to further refine the results of the previous phase via addition/removal of points. This approach produces in general an improvement in terms of accuracy. Nevertheless, our LOOCV-ARS algorithm also considers the chance of applying only the first stage without the second one intervenes. This fact generally occurs when a prescribed level of precision via LOOCV based refinement is achieved.*

Remark 4.3. *Another important aspect concerns the choice of test points to be used in the error indicator* (20) *(or* (21)*) of the second stage of the algorithm. Though the selection of such points is arbitrary, it is however advisable to use specific sets of points that can be generated sequentially, so that at each iteration the algorithm can be validated on different test points but, at the same time, every numerical experiment can be repeated. An example of sequence that meets the above-mentioned requirements is given by the van der Corput sequence, which is used to create the so-called Halton points [22, 38]. This is the choice we will do later in Section 5.*

4.3. Complexity analysis

In the first stage of our algorithm the computational cost is due to the iterated use of LOOCV technique as an error indicator. This implies that an inversion of collocation matrix in (6) needs to be done so as to compute all components of error terms in (15). The result is a $\mathcal{O}(N^3)$ computational complexity. This process is initially applied on the sets $X_N = X^1_{N^{(1)}}$ and $Z_N = Z^1_{N^{(1)}}$, but it have then to be iterated on the refined sets $X^1_{N^{(j)}}$ and $Z^1_{N^{(j)}}$, for $j = 2, 3, \ldots$, till the stop criterion is satisfied for all terms of error indicator (17).

The second stage of the scheme is characterized by the ARS procedure, which follows the paradigm to solve, estimate and refine or coarsen the discretization points. Also this process is an iterative approach, so the computational technique we are considering is generally repeated a few times by updating iteration-by-iteration the sets $X^2_{N^{(k)}}$ and $Z^2_{N^{(k)}}$, with $k = 1, 2, \ldots$. However, in this case the error indicator (20)–(21) is obtained by solving two collocation problems (i.e. with and without additional boundary points). This choice needs to invert two matrices of the form (18), thus requiring a complexity cost of $\mathcal{O}(M^3)$. Note that here, for an iteration k fixed, M denotes in both cases the generic number of discretization points.

Once the final discretization points were found, the two-stage algorithm stops and the approximate collocation solution can be evaluated on a set of evaluation points. This phase requires a vector-vector multiplication (for each evaluation point) with vectors of the same size of the final set of centers.

5. Numerical experiments and discussion

In this section we analyze the performance of the two-stage adaptive scheme described in Section 4. All the procedures are implemented in MATLAB, while the results are carried out on a laptop with an Intel(R) Core(TM) i7-4500U CPU 1.80 GHz processor and 4GB RAM.

In the following we focus on a large number of experiments, which consist in the solution of some 2D elliptic PDE problems via Kansa's method. More precisely, we deal with some types of benchmark Poisson problems with Dirichlet BCs. This offers the chance to show both flexibility and efficacy of our adaptive refinement strategies. Therefore, we discuss the numerical results aiming to test our two-stage scheme, which first is based on a LOOCV procedure used as error indicator (Stage 1) and then on the ARS technique (Stage 2). As regards the mode of point addition presented in Subsection 4.1, we choose to use the four-point approach shown in Figure 1 (left). In fact, after carrying out many experiments, that refinement strategy associated with the indicator (15) (or (17)) has turned out to be the most effective because it allows us to cover more uniformly the regions that need to be refined and further does not result in an excessive increase of collocation points. Moreover, we remark that in the second stage of the algorithm we used Halton points [14] as test points. As said in Remark 4.3, it is however obvious that the choice of these validation points is absolutely arbitrary and other possible distribution of test points can be used. In particular, we analyzed in some test examples the behavior of the algorithm when the sets $Y^{(k)}$ of test points are not Halton points, but for instance pseudorandom points generated by the MATLAB `rand` command. Basically, the use of different test points influences the algorithm performances. However, in general we obtained similar results (e.g. in terms of accuracy) to the Halton-based ones, so in this section we will only focus on the latter case.

In our tests we consider the nonsymmetric RBF collocation method by using basis functions of different smoothness as IMQ, M6 or M4 (see Table 1). The shape parameter ε is taken fixed, but in this work we also study the behavior of the numerical algorithm when ε varies. From a computational standpoint it is indeed relevant to analyze what is the effect on the code performance due to a change of ε. Although the main focus of this paper is devoted to see how the new adaptive scheme works (and therefore not to seek the best possible accuracy), from literature we know that the selection of the RBF associated with its shape parameter is a remarkable issue because this choice can influence (even significantly) the accuracy of the numerical method (see e.g. [14, 15]).

To investigate the accuracy of the adaptive scheme, we compute the Root Mean Square Error (RMSE) whose formula is given by

$$\text{RMSE} = \sqrt{\frac{1}{N_{eval}} \sum_{i=1}^{N_{eval}} |u(\boldsymbol{\xi}_i) - \hat{u}(\boldsymbol{\xi}_i)|^2} = \frac{1}{\sqrt{N_{eval}}} ||u - \hat{u}||_2,$$

where the $\boldsymbol{\xi}_i$, $i = 1, \ldots, N_{eval}$, are a grid of evaluation points and $N_{eval} = 40 \times 40$. Note that this value of N_{eval} refers to the case in which the considered domain is a square, while it is obviously reduced in case of a domain of generic (regular or irregular) shape because the latter can always be included in a square. Then, in order to analyze the stability of the numerical method, we evaluate the Condition Number (CN) of the

collocation matrix A in (6) by using the MATLAB command `cond`. As to the efficiency of our algorithm, we report the CPU times computed in seconds.

5.1. Poisson problem with Dirichlet BCs

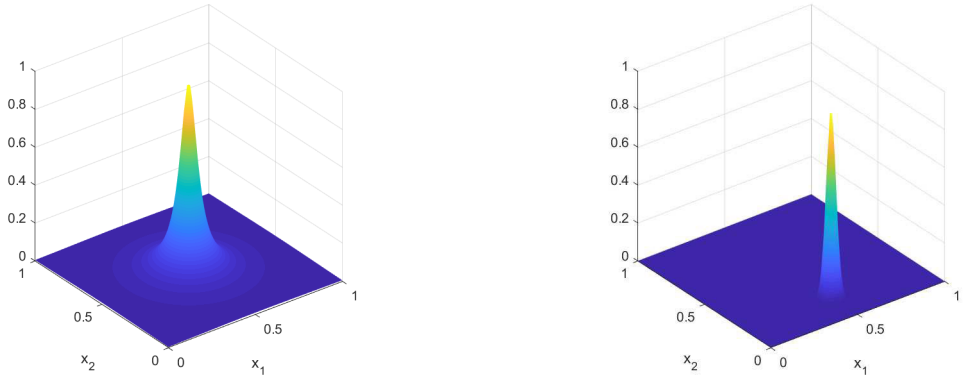
In this subsection we show numerical results acquired from experiments by solving some elliptic PDE problems via the nonsymmetric RBF collocation. In particular, we consider Poisson type problems, i.e. the elliptic operator in (1) is given by $\mathcal{L} = -\Delta$, assuming Dirichlet BCs in (2). Hence, the PDE problem in (1)-(2) can be defined as follows:

$$\begin{aligned} -\Delta u(\boldsymbol{x}) &= f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) &= g(\boldsymbol{x}), \qquad \boldsymbol{x} \in \partial\Omega. \end{aligned} \tag{22}$$

To analyze the performance of the two-stage adaptive algorithm, we take two test problems of the form (22) defined on the domain $\Omega = [0,1]^2$. The exact solutions of such Poisson problems are given by

$$\begin{aligned} \text{P1}: \quad & u_1(x_1,x_2) = \frac{1}{25(4x_1-2)^2 + 25(4x_2-2)^2 + 1}, \\ \text{P2}: \quad & u_2(x_1,x_2) = \exp(-1000((x_1-0.5)^2 + (x_2-0.2)^2)). \end{aligned}$$

In Figure 3 we show a graphical representation of such analytic solutions. Note that these problems are also studied in recent works (see e.g. [27, 30]), and they turn to be good examples for testing new adaptive refinement algorithms.



$u_1(x_1,x_2)$ $\qquad\qquad$ $u_2(x_1,x_2)$

Figure 3: Exact solutions of problems studied in Subsection 5.1.

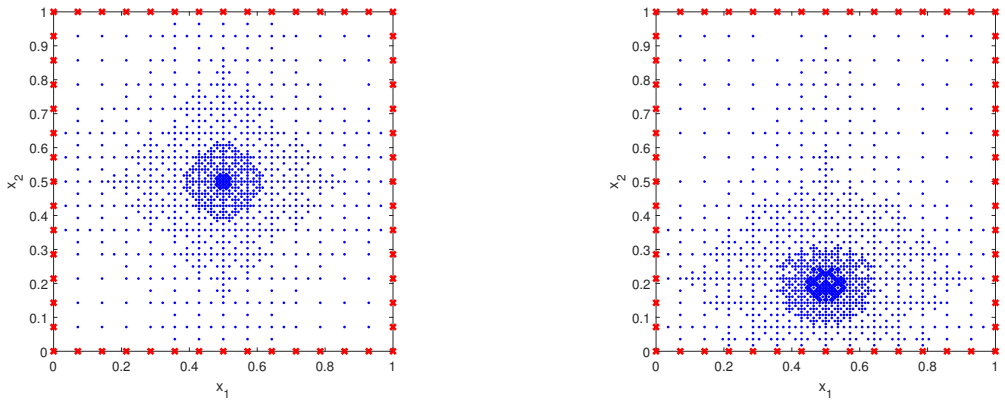
The main scope of this analysis is therefore to point out the performance of our two-stage adaptive scheme. In this series of numerical tests we begin the iterative process by taking $N = 225$ grid collocation points, consisting of $N_I = 169$ interior points and $N_B = 56$ boundary points. In Table 2 we show a summary of all results obtained by using M6 with $\varepsilon = 4$, also indicating the total number N_{fin} of collocation points obtained to achieve the final result. Moreover, in Table 3 we provide the results derived from the use of IMQ with $\varepsilon = 12$, but starting from $N = 625$ grid collocation points (i.e., with $N_I = 529$ and $N_B = 96$). Note that the value of τ regarding the LOOCV is provided case-by-case, while the thresholds associated with ARS are given by $(\tau_{low}, \tau_{upp}) = (10^{-8}, 10^{-4})$. We further observe that in these examples the level of precision achieved by means of LOOCV is very accurate, so the ARS phase does not intervene. Then, in Figure 4 we plot the final discretization point sets in case of M6.

Problem	N_{fin}	RMSE	CN	time	τ
P1	1114	7.98e−7	9.25e+12	3.8	0.5
P2	1342	7.66e−7	1.30e+13	5.3	0.9

Table 2: Results for problems studied in Subsection 5.1 by starting from $N = 225$ using M6 with $\varepsilon = 4$ and $(\tau_{low}, \tau_{upp}) = (10^{-8}, 10^{-4})$.

Problem	N_{fin}	RMSE	CN	time	τ
P1	1285	1.29e−5	4.44e+17	5.3	0.5
P2	1221	1.11e−6	2.93e+09	2.6	0.9

Table 3: Results for problems studied in Subsection 5.1 by starting from $N = 625$ using IMQ with $\varepsilon = 12$ and $(\tau_{low}, \tau_{upp}) = (10^{-8}, 10^{-4})$.



P1: final discretization points

P2: final discretization points

Figure 4: Final discretization point configurations for problems studied in Subsection 5.1 using M6 with $\varepsilon = 4$.

Analyzing the numerical results, we can observe that the adaptive algorithm works well in both of different situations. It allows us to detect the regions of Ω characterized by significant variations, increasing the number of points only in those areas where the accuracy is not enough. In general, also considering the values of tolerances used, the two-stage adaptive scheme provides accurate results. In fact, when solving the problems P1 and P2, we get remarkable errors with a relatively small number of points. In this case, we further remark that the latter problems are quite hard examples to handle, since often – as happens for instance in [3, 4] – an adaptive algorithm can undersample the peak present in the domain making quite difficult or impossible to get a high level of precision. From Figure 4 it is however evident that our adaptive scheme identifies very clearly what are the areas that need to be refined. Then, focusing on the stability, we report the CN that in all problems faced has a order of magnitude around 10^{+12}–10^{+13}. Though the nature of RBF methods influences in terms of conditioning these results, we consider worthwhile to note as a good refinement strategy enables us to keep under control even the ill-conditioning avoiding the number of collocation points excessively grows. As to efficiency Table 2 highlights that the two-stage algorithm completes the refinement process in few seconds only. For both problems we have then analyzed what

happens if the LOOCV is applied with values of τ smaller than 0.5 for P1 and 0.9 for P2 (cf. Tables 2–3), trying to obtain higher accuracy. Basically, the algorithm behavior is similar even if we have to distinguish the two studied cases. For problem P1 we can get similar errors till $\tau \approx 0.2$ for both M6 and IMQ, but then for $\tau < 0.15$ the method does not converge anymore. For problem P2, instead, with M6 numerical convergence is achieved till $\tau = 0.5$ (without being remarkable changes in terms of accuracy), while with IMQ we observe an accuracy improvement (and so convergence) till $\tau = 0.01$ that is results in a RMSE = 1.65e−7, with $N_{fin} = 2728$ points and CN around 10^{+16}. Moreover, in order to demonstrate that our adaptive scheme is helpful, in Table 4 we report the results obtained by applying the non-adaptive Kansa method [17] on uniform/gridded point sets. From a comparison with Table 2, on the one hand we can observe that for both problems P1 and P2 a similar level of accuracy can be achieved only when we use about 7000 collocation points, even if this leads to significantly higher condition numbers (around 10^{+17}). On the other hand, we remark as with a similar number of points the accuracy shown in Table 4 is much lower than the one obtained by using our adaptive scheme (cf. Table 2).

Problem	N	RMSE	CN
P1	1089	4.73e−3	3.68e+13
	1156	3.52e−3	4.85e+13
	3600	1.68e−5	9.14e+15
	7056	8.84e−7	2.43e+17
	7225	6.75e−7	2.59e+17
P2	1296	1.57e−3	8.26e+13
	1369	1.31e−3	1.07e+14
	3600	2.02e−5	9.14e+15
	6889	8.28e−7	2.02e+17
	7056	6.35e−7	2.43e+17

Table 4: Results for problems studied in Subsection 5.1 obtained by applying the non-adaptive Kansa method [17] on uniform/gridded points and using M6 with $\varepsilon = 4$.

Finally, as a comparison, we conclude this subsection by reporting the results obtained with the adaptive P1-FEM algorithm, which is efficiently implemented in MATLAB and can be freely online downloaded (see [19]). Although a reliable comparison between conceptually different methods is quite difficult to do, in Table 5 we show the numerical results obtained by applying the P1-FEM with $\theta = 0.25$ and $nEmax = 10^{+5}$. Notice that the parameter θ can assume values in the interval $(0, 1)$: $\theta \to 1$ corresponds to almost uniform mesh-refinement, whereas $\theta \to 0$ leads to highly adapted meshes (see [19] for details). In particular, from Tables 2–3 and 5 we can observe as our adaptive scheme is more accurate than P1-FEM. Additionally, even though CPU times of two algorithms are similar, the P1-FEM scheme needs to generate a (final) number of discretization points that is considerably larger than those of our algorithm.

Problem	N_{fin}	RMSE	time
P1	63248	3.77e−5	2.7
P2	59042	5.70e−4	2.5

Table 5: Results for problems studied in Subsection 5.1 and obtained via P1-FEM [19] with $\theta = 0.25$ and $nEmax = 1.0$e+5.

5.2. Modeling a wave front problem

In this subsection we consider a wave front problem [30] involving a Poisson problem with Dirichlet BCs of the form (22). This problem is solved on two different domains:

(a) the square domain $\Omega = [0,1]^2$;
(b) the starfish like shape domain $\Omega \subset [0,1]^2$ bounded by the parametric curve [26]

$$r(\theta) = 0.8 + 0.1(\sin(6\theta) + \sin(3\theta)), \qquad \theta \in [0, 2\pi).$$

The exact solution is given by

$$u(x_1, x_2) = \arctan(\alpha(\sqrt{(x_1 - \tilde{x}_1)^2 + (x_2 - \tilde{x}_2)^2} - r_0)), \tag{23}$$

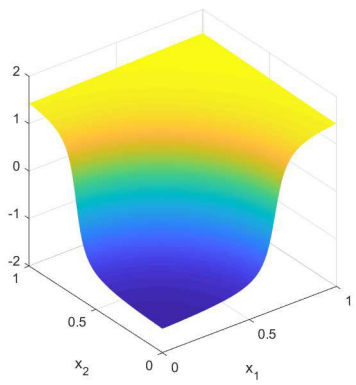
where parameters in (23) are defined as follows:

- $(\tilde{x}_1, \tilde{x}_2) = (-0.05, -0.05)$ is the center of the circular wave front;
- $r_0 = 0.7$ is the distance from the wave front to the center of the circle;
- $\alpha = 20$ gives the steepness of the wave front.

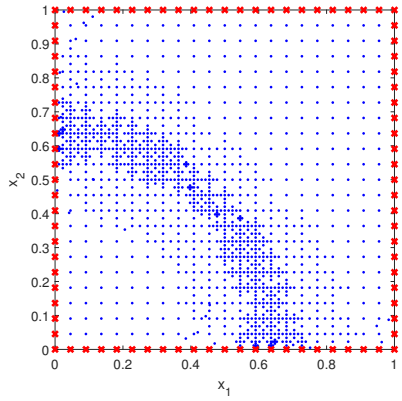
Such parameters determine the steepness and location of the wave front. We observe that the presence of the arctangent in (23) generates a mild singularity at the center of the circle. However, we choose the center of the circle to be outside the domain so as to examine the performance on the wave front, not the singularity. The solution is thus characterized by a mild wave front that is not symmetric around the origin [30].

In the tests we compute the approximate solution by applying our two-stage adaptive scheme. The plots of exact solution (23) given in both cases (a) and (b) are reported in Figures 5–6 (left). To analyze the behavior of the algorithm, we use the functions M4, M6 and IMQ with various values of ε. Therefore, in Tables 6–7 we show the output of the LOOCV-ARS algorithm (in comparison with the use of the LOOCV one alone described in Algorithm 1) obtained by starting in case (a) of the square domain from $N = 529$ grid points, with $N_I = 441$ and $N_B = 88$, and in case (b) of the starfish domain from $N = 305$ points, with $N_I = 245$ and $N_B = 60$. Note that the value $N_I = 245$ has been obtained by reducing the $N_I = 441$ grid interior points contained in the square domain. Moreover, in both situations we fix the same tolerances as follows: $(\tau; \tau_{low}, \tau_{upp}) = (4; 10^{-8}, 10^{-4})$. Finally, from results in Tables 6–7 we can observe relatively small variations in terms of accuracy (RMSE), stability (CN) and efficiency (time). In particular, from a comparison between Table 6 and Table 7 we can only observe a reduction of CPU time for the starfish example (while RMSE and CN are rather similar to the square domain case), but that is essentially due to the smaller number of points present in the starfish like shape domain. Nevertheless, while for M6 the number of final discretization points is similar, for M4 and IMQ the increase of ε results in a larger number of collocation points in the final configuration. This can be attributable to the greater effort required to M4 and IMQ to achieve the prefixed thresholds. Furthermore, from the comparison between LOOCV and LOOCV-ARS we can remark a general improvement in terms of accuracy when the two-stage algorithm is applied except for the case of M4 with $\varepsilon = 4$ in Table 6, where the ARS phase does not intervene. An example of final discretization point configuration for LOOCV-ARS is given in Figures 5–6 (right).

Then, in order to illustrate that combining LOOCV and ARS is advantageous, in Tables 8–9 we compare the three approaches, i.e. the LOOCV alone (Stage 1), the ARS alone (Stage 2) and the LOOCV-ARS scheme (Stage 1 + Stage 2). In doing that, we consider both cases (a) and (b) above discussed but now we show the results by starting from $N = 441$ grid points, with $N_I = 361$ and $N_B = 80$, for case (a) and from $N = 261$ points, with $N_I = 201$ and $N_B = 60$, for case (b). The parameters used in these tests are $\tau = 3$ and $(\tau_{low}, \tau_{upp}) = (10^{-8}, 10^{-4})$. From these experiments we can observe as the LOOCV scheme generally yields more accurate results than the ARS one. However, the two-stage method (LOOCV-ARS) outperforms both LOOCV and ARS applied alone. Additionally, although the LOOCV appears to be more effective in terms of accuracy than the ARS technique, the latter applied as second stage of the algorithm turns out to be useful to further reduce the computational error.

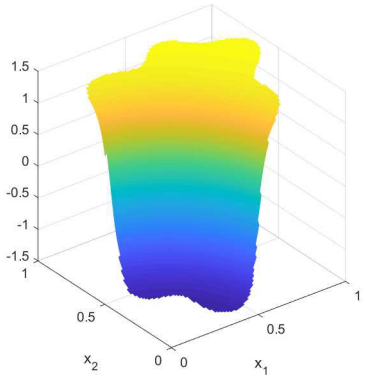


$u(x_1, x_2)$ on square domain

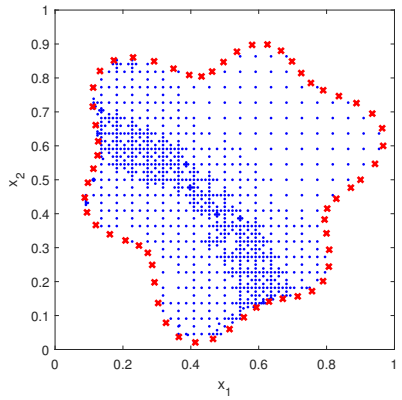


final discretization points

Figure 5: Exact solution of wave front problem in (a) defined on the square domain (left) and final discretization point configuration for LOOCV-ARS using M6 with $\varepsilon = 4$ (right).



$u(x_1, x_2)$ on starfish domain



final discretization points

Figure 6: Exact solution of wave front problem in (b) defined on the starfish like shape domain (left) and final discretization point configuration for LOOCV-ARS using M6 with $\varepsilon = 4$ (right).

Finally, we show the results obtained by using the adaptive P1-FEM scheme [19], which is applied by taking the parameters $\theta = 0.25$ and $nEmax = 10^{+5}$. Comparing the numerical results given in Tables 6 and 10 we can remark that our adaptive algorithm is comparable in terms of accurate with the adaptive P1-FEM scheme. However, the P1-FEM needs a much larger amount of points than LOOCV-ARS.

5.3. Approximation of a boundary layer problem

In this subsection we deal with a boundary layer problem [29] involving a Poisson type PDE along with Dirichlet BCs in (22) defined in the domain $\Omega = [0, 1]^2$. The solution of this problem is

$$u(x_1, x_2) = \exp(-200x_1^2) + \exp(-200x_2^2).$$

As shown in Figure 7 (left), we observe that there are two sharp regions near the boundaries $x_1 = 0$ and $x_2 = 0$ [29]. The point distribution must be refined near the boundaries with the sharp gradient so that the localized features can be captured properly. From a graphical inspection of Figure 7 (right) this

RBF	ε	LOOCV				LOOCV-ARS			
		N_{fin}	RMSE	CN	time	N_{fin}	RMSE	CN	time
M4	4	1469	3.24e−4	1.54e+10	7.1	1469	3.24e−4	1.54e+10	7.1
	5	1464	3.90e−4	6.18e+09	8.0	1553	3.22e−4	6.29e+09	10.9
	6	1461	3.95e−4	2.88e+09	8.9	1688	3.33e−4	3.00e+09	11.7
M6	4	1399	1.83e−4	5.23e+13	10.1	1420	1.59e−4	5.25e+13	13.0
	5	1401	1.86e−4	2.79e+13	12.0	1425	1.61e−4	2.80e+13	16.2
	6	1395	1.88e−4	1.42e+12	10.2	1411	1.63e−4	1.54e+12	16.9
IMQ	10	1475	1.57e−3	7.60e+09	9.4	1954	5.93e−4	2.41e+11	15.1
	11	1345	1.63e−3	6.04e+09	8.6	1831	7.16e−4	1.47e+10	13.9
	12	1283	1.39e−3	1.42e+08	5.8	1778	4.50e−4	5.37e+09	9.9

Table 6: Comparison between results obtained by applying the LOOCV scheme (Algorithm 1 only) with $\tau = 4$ and the LOOCV-ARS scheme (Algorithm 1 + 2) with $(\tau; \tau_{low}, \tau_{upp}) = (4; 10^{-8}, 10^{-4})$. Tests have been carried out by starting from $N = 529$ for wave front problem in (a) defined on the square domain.

RBF	ε	LOOCV				LOOCV-ARS			
		N_{fin}	RMSE	CN	time	N_{fin}	RMSE	CN	time
M4	4	1031	3.27e−4	3.38e+10	2.7	1038	3.26e−4	3.41e+10	3.0
	5	1024	3.27e−4	1.40e+10	2.8	1032	3.26e−4	1.41e+10	3.0
	6	1022	3.28e−4	6.69e+09	2.8	1030	3.27e−4	6.74e+09	2.9
M6	4	996	1.97e−4	4.45e+13	4.4	1003	1.96e−4	4.63e+13	5.0
	5	989	2.20e−4	8.09e+12	3.3	995	1.99e−4	8.12e+12	3.6
	6	983	2.21e−4	2.71e+12	3.0	989	2.00e−4	2.72e+12	3.5
IMQ	10	1157	2.53e−4	1.26e+10	5.1	1402	1.94e−4	4.84e+11	9.0
	11	1174	3.10e−4	6.27e+09	5.0	1407	2.35e−4	1.78e+11	8.9
	12	1137	4.15e−4	3.61e+08	4.8	1369	3.43e−4	3.75e+09	7.6

Table 7: Comparison between results obtained by applying the LOOCV scheme (Algorithm 1 only) with $\tau = 4$ and the LOOCV-ARS scheme (Algorithm 1 + 2) with $(\tau; \tau_{low}, \tau_{upp}) = (4; 10^{-8}, 10^{-4})$. Tests have been carried out by starting $N = 305$ for wave front problem in (b) defined on the starfish like shape domain.

target is achieved since our adaptive algorithm refine exactly the areas close to above-mentioned boundaries. Moreover, in Table 11 we summarize all essential information about the performance of our LOOCV-ARS algorithm in comparison with the use of the LOOCV scheme only, i.e. applying the first stage described in Algorithm 1. Both codes were performed by starting from a grid of $N = 625$ collocation points with $N_I = 529$ and $N_B = 96$ and using the IMQ-RBF. These results confirm once more the efficacy of our two-stage refinement scheme.

Finally, to conclude this subsection, we compare the previous results with the ones obtained with the adaptive P1-FEM algorithm [19]. In particular, in Table 12 we report the numerical results given by the P1-FEM with $\theta = 0.25$ and $nEmax = 10^{+5}$. Also in this situation we see as our LOOCV-ARS scheme is at

		LOOCV			ARS			LOOCV-ARS		
RBF	ε	N_{fin}	RMSE	time	N_{fin}	RMSE	time	N_{fin}	RMSE	time
M4	4	1677	3.83e−4	11.3	1304	7.65e−3	6.1	1713	3.67e−4	14.5
M6	4	1399	2.79e−4	9.8	1291	2.13e−3	9.4	1490	2.53e−4	14.4
IMQ	10	1451	1.57e−3	8.7	1839	1.49e−3	15.1	1939	6.46e−4	16.1

Table 8: Comparison between results obtained by applying the LOOCV only with $\tau = 3$, the ARS only with $(\tau_{low}, \tau_{upp}) = (10^{-8}, 10^{-4})$ and the LOOCV-ARS with $(\tau; \tau_{low}, \tau_{upp}) = (3; 10^{-8}, 10^{-4})$. Tests have been carried out by starting from $N = 441$ for wave front problem in (a) defined on the square domain.

		LOOCV			ARS			LOOCV-ARS		
RBF	ε	N_{fin}	RMSE	time	N_{fin}	RMSE	time	N_{fin}	RMSE	time
M4	4	1277	1.88e−4	5.4	949	2.29e−3	2.5	1512	1.86e−4	7.3
M6	4	998	1.76e−4	5.6	812	4.89e−3	3.8	1145	1.68e−4	8.2
IMQ	10	1197	2.33e−4	6.1	1022	3.95e−3	4.6	1446	1.95e−4	9.8

Table 9: Comparison between results obtained by applying the LOOCV only with $\tau = 3$, the ARS only with $(\tau_{low}, \tau_{upp}) = (10^{-8}, 10^{-4})$ and the LOOCV-ARS with $(\tau; \tau_{low}, \tau_{upp}) = (3; 10^{-8}, 10^{-4})$. Tests have been carried out by starting $N = 261$ for wave front problem in (b) defined on the starfish like shape domain.

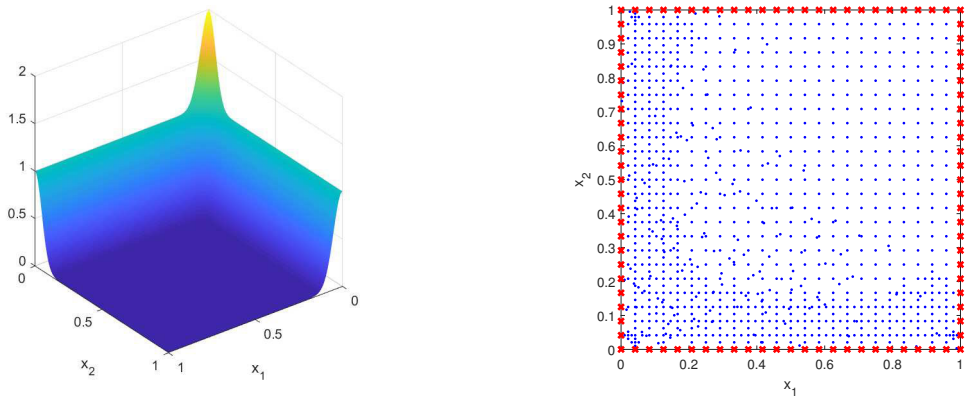
N_{fin}	RMSE	time
63847	1.44e−4	3.0

Table 10: Results for problem studied in Subsection 5.2 on the square domain and obtained via P1-FEM [19] with $\theta = 0.25$ and $nEmax = 1.0e{+}5$.

	LOOCV				LOOCV-ARS			
ε	N_{fin}	RMSE	CN	time	N_{fin}	RMSE	CN	time
4.0	915	2.04e−4	2.02e+18	2.4	984	6.76e−5	3.98e+18	4.3
4.5	733	1.10e−3	4.80e+17	1.7	1025	3.18e−4	1.97e+17	3.6
5.0	781	2.27e−3	2.53e+14	1.9	1083	6.22e−4	3.24e+14	4.1
5.5	939	1.09e−4	3.06e+13	3.7	1129	1.68e−4	1.61e+14	5.5
6.0	979	2.90e−4	2.16e+10	3.2	1108	1.33e−4	1.01e+11	5.1

Table 11: Comparison between results obtained by applying the LOOCV scheme (Algorithm 1 only) with $\tau = 2$ and the LOOCV-ARS scheme (Algorithm 1 + 2) with $(\tau; \tau_{low}, \tau_{upp}) = (2; 10^{-8}, 10^{-4})$. Tests have been carried out by starting from $N = 625$ and using IMQ for boundary layer problem.

least comparable in terms of accurate and efficiency with the P1-FEM, but the latter requires a considerably larger number of points than LOOCV-ARS.



$u(x_1, x_2)$ final discretization points

Figure 7: Exact solution of boundary layer problem (left) and final discretization point configuration for LOOCV-ARS using IMQ with $\varepsilon = 6$ (right).

N_{fin}	RMSE	time
72750	3.49e−4	4.4

Table 12: Results for problem studied in Subsection 5.3 and obtained via P1-FEM [19] with $\theta = 0.25$ and $nEmax = 1.0\text{e}{+5}$.

Remark 5.1. *The role of the parameters τ, τ_{low} and τ_{upp} is important for the algorithm performances. In a preliminary study, we analysed the problem of how to choose such parameters. In particular, in Stage 2 τ_{upp} defines the accuracy expected on all domain, while τ_{low} is used to reduce the number of points when the precision is excessive. In this phase the role of τ_{upp} highly influences the performance: for instance, a larger (smaller) value of τ_{upp} reduces (improves) the entire accuracy of the adaptive method. On the contrary, τ_{low} is used for the removal of points and influences less the final result. It is however important that the two tolerances are not too close, in order to avoid a simultaneous effect of addition and removal of points in the same iteration. The choice of τ in Stage 1 turned out to be more difficult to do, since it seems to be influenced by several aspects depending on the PDE problem, the RBF used, the value of ε taken etc. For this reason the selection of τ was done experimentally.*

6. Conclusions and future work

In this paper we proposed an adaptive algorithm for solving elliptic PDEs via Kansa's method. The scheme is based on two computational phases, which involve the LOOCV technique used as an error indicator and the ARS procedure applied as a further refinement strategy. This two-stage approach enabled us to identify via LOOCV the domain regions that need to be refined, then giving the possibility of an adaptive addition/removal of points via ARS. Extensive numerical results showed the good performance of our new LOOCV-ARS scheme on different PDE problems.

As future work we propose to extend our adaptive algorithm for solving other types of differential equations. This could include the modeling of both time-independent and time-dependent PDEs. Another important aspect to investigate concerns then the study of new refinement techniques that could be tailored on the PDE model. Moreover, as the method has a computational cost of $\mathcal{O}(N^3)$, which makes its use prohibitive in large scale applications, additional investigations are needed to reduce it. Finally, we are

interested in further analyzing how this adaptive method behaves when the problem solution is not isotropic, since in such cases the use of non-uniform points could be useful to improve the refinement process.

Acknowledgments

The authors acknowledge support from the Department of Mathematics "Giuseppe Peano" of the University of Torino via 2016-2017 projects "Numerical and computational methods for applied sciences" and "Multivariate approximation and efficient algorithms with applications to algebraic, differential and integral problems". Moreover, this work was partially supported by GNCS–INdAM 2018 project "Methods, algorithms and applications of multivariate approximation". This research has been accomplished within RITA (Research ITalian network on Approximation).

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] 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, Adaptive meshless refinement schemes for RBF-PUM collocation, Appl. Math. Lett. 90 (2019) 131–138.
[4] R. Cavoretto, A. De Rossi, Error indicators and refinement strategies for solving Poisson problems through a RBF partition of unity collocation scheme, Appl. Math. Comput. (2019), in press.
[5] R. Cavoretto, T. Schneider, P. Zulian, OpenCL based parallel algorithm for RBF-PUM interpolation, J. Sci. Comput. 74 (2018) 267–289.
[6] W. Chen, Z.-J. Fu, C.S. Chen, Recent Advances on Radial Basis Function Collocation Methods, Springer Briefs in Applied Science and Technology, Springer, Heidelberg, 2014.
[7] P. Craven, G. Wahba, Smoothing noisy data with spline functions, Numer. Math. 31 (1979) 377–403.
[8] O. Davydov, D.T. Oanh, Adaptive meshless centres and RBF stencils for Poisson equation, J. Comput. Phys. 230 (2011) 287–304.
[9] S. De Marchi, A. Martínez, E. Perracchione, M. Rossini, RBF-based partition of unity methods for elliptic PDEs: Adaptivity and stability issues via variably scaled kernels, J. Sci. Comput. 79 (2019) 321–344.
[10] A. De Rossi, Hierarchical Numerical Methods for Numerical Approximation of Partial Differential Equations (in italian), PhD thesis, University of Milano, 1997.
[11] A. De Rossi, A posteriori error estimates for hierarchical methods, Riv. Mat. Univ. Parma (6) 1 (1998), 53–69.
[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, Interdisciplinary Mathematical Sciences, vol. 6, World Scientific Publishing Co., Singapore, 2007.
[15] G.E. Fasshauer, M.J. McCourt, Kernel-based Approximation Methods using Matlab, Interdisciplinary Mathematical Sciences, Vol. 19, World Scientific Publishing Co., Singapore, 2015.
[16] G.E. Fasshauer, J.G. Zhang, On choosing "optimal" shape parameters for RBF approximation, Numer. Algorithms 45 (2007), 345–368.
[17] A.I. Fedoseyev, M.J. Friedman, E.J. Kansa, Improved multiquadric method for elliptic partial differential equations via PDE collocation on the boundary, Comput. Math. Appl. 43 (2002) 439–455.
[18] B. Fornberg, N. Flyer, A Primer on Radial Basis Functions with Applications to the Geosciences, SIAM, Philadelphia, 2015.
[19] S. Funken, D. Praetorius, P. Wissgott, Efficient implementation of adaptive P1-FEM in Matlab, Comput. Methods Appl. Math. 11 (2011) 460–490.
[20] A. Golbabai, E. Mohebianfar, H. Rabiei, On the new variable shape parameter strategies for radial basis functions, Comput. Appl. Math. 34 (2015) 691–704.
[21] G.H. Golub, M. Heath, G. Wahba, Generalized cross-validation as a method for choosing a good ridge parameter, Technometrics 21 (1979) 215–223.
[22] J.H. Halton, On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals, Numer. Math. 2 (1960) 84–90.
[23] Y.C. Hon, R. Schaback, On unsymmetric collocation by radial basis functions, Appl. Math. Comput. 119 (2001) 177–186.
[24] 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.
[25] G. Kosec, B. Šarler, H-adaptive local radial basis function collocation meshless method, CMC 26 (2011), 227–253.

[26] E. Larsson, E. Lehto, A. Heryudono, B. Fornberg, Stable computation of differentiation matrices and scattered node stencils based on Gaussian radial basis functions, SIAM J. Sci. Comput. 35 (2013) A2096–A2119.
[27] 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.
[28] C.-F. Lee, L. Ling, R. Schaback, On convergent numerical algorithms for unsymmetric collocation, Adv. Comput. Math. 30 (2009) 339–354.
[29] N.A. Libre, A. Emdadi, E.J. Kansa, M. Shekarchi, M. Rahimian, A fast adaptive wavelet scheme in RBF collocation for nearly singular potential PDEs, CMES Comput. Model. Eng. Sci. 38 (2008) 263–284.
[30] W. F. Mitchell, A collection of 2D elliptic problems for testing adaptive grid refinement algorithms, Appl. Math. Comput. 220 (2013) 350–364.
[31] 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.
[32] S. Rippa, An algorithm for selecting a good value for the parameter c in radial basis function interpolation, Adv. Comput. Math. 11 (1999) 193–210.
[33] Y.V.S.S. Sanyasiraju, C. Satyanarayana, On optimization of the RBF shape parameter in a grid-free local scheme for convection dominated problems over non-uniform centers, Appl. Math. Model. 37 (2013) 7245–7272.
[34] R. Schaback, Convergence of unsymmetric kernel-based meshless collocation methods, SIAM J. Numer. Anal. 45 (2007) 333–351.
[35] S.A. Sarra, E.J. Kansa, Multiquadric Radial Basis Function Approximation Methods for the Numerical Solution of Partial Differential Equations, Advances in Computational Mechanics, vol. 2, Tech Science Press, Encino, CA, 2009.
[36] M. Uddin, On the selection of a good value of shape parameter in solving time-dependent partial differential equations using RBF approximation method, Appl. Math. Model. 38 (2014) 135–144.
[37] H. Wendland, Scattered Data Approximation, Cambridge Monogr. Appl. Comput. Math., vol. 17, Cambridge Univ. Press, Cambridge, 2005.
[38] T.-T. Wong, W.-S Luk, P.-A. Heng, Sampling with Hammersley and Halton points, J. Graphics Tools 2 (1997) 9–24.
[39] G. Yao, J. Duo, C. S. Chen, L. H. Shen, Implicit local radial basis function interpolations based on function values, Appl. Math. Comput. 265 (2015) 91–102.