Journal of Scientific Computing manuscript No.
(will be inserted by the editor)

RBF-based partition of unity methods for elliptic PDEs: Adaptivity and stability issues via variably scaled kernels

S. De Marchi · A. Martínez · E. Perracchione · M. Rossini

Received: date / Accepted: date

Abstract We investigate adaptivity issues for the approximation of Poisson equations via radial basis function-based partition of unity collocation. The adaptive residual subsampling approach is performed with quasi-uniform node sequences leading to a flexible tool which however might suffer from numerical instability due to ill-conditioning of the collocation matrices. We thus develop a hybrid method which makes use of the so-called variably scaled kernels. The proposed algorithm numerically ensures the convergence of the adaptive procedure.

Keywords Partition of unity method · Radial basis functions · Meshfree approximation · Elliptic PDEs · Variably scaled kernels

Mathematics Subject Classification (2000) 65D05 · 65D15 · 65N99

1 Introduction

The Partition of Unity (PU) scheme has been used for interpolation from the sixties when D. Shepard introduced, as undergraduate student at Harvard University, what are now called the *Shepard's weights* [42]. Later, this local approach based on decomposing the original reconstruction domain into many *subdomains* or *patches* has been coupled with Radial Basis Functions (RBFs), see e.g. [17,46]. Moreover, among several applications (see e.g. [7,21]), the RBF-PU method for

S. De Marchi
Dipartimento di Matematica "Tullio Levi-Civita", Università di Padova, Italia
E-mail: demarchi@math.unipd.it

A. Martínez
Dipartimento di Matematica "Tullio Levi-Civita", Università di Padova, Italia
E-mail: acalomar@math.unipd.it

E. Perracchione
Dipartimento di Matematica "Tullio Levi-Civita", Università di Padova, Italia
E-mail: emma.perracchione@math.unipd.it

M. Rossini
Dipartimento di Matematica e Applicazioni, Università di Milano Bicocca, Italia
E-mail: milvia.rossini@unimib.it

solving Partial Differential Equations (PDEs), first introduced in the mid nineties [33], is nowadays a popular and well developed technique (refer to [23,38,44]). The importance of investigating new robust tools for solving PDEs easily follows from the fact that they govern many multivariate physical phenomena, such as for instance the distribution of heat, the propagation of sound or light waves and fluid dynamics.

Here, our goal consists in developing an adaptive PU meshfree collocation method for Poisson equations independent of the problem geometry. In the current literature, except for particular applications (see e.g. [27]), most papers about adaptive RBF collocation and multiscale methods only consider global approximation methods or RBF-Finite Difference (FD) local approaches (see [9,15,19, 34]). In [19], the approximate solution is constructed with a multilevel approach in which Compactly Supported RBFs (CSRBFs) of smaller support are used on an increasingly finer mesh, similarly as done also in [25]. In the cited papers the approximation is found by adaptively selecting points so that the sampling density follows the regions of high variation of the solution. Such approach is also adopted here and, thanks to the use of the so-called *geometric greedy points*, introduced in [10] and recently analyzed in [39], the Adaptive Residual Subsampling (ARS) scheme shows to work quite well with such distribution of nodes, without computing grid data as outlined in [15].

When using RBF-based methods, the oversampling induced by adaptivity and the *shape parameter* as well (see e.g. [18]) may lead to ill-conditioning of the collocation matrices and thus a stable approximation of the solution is crucial. For the Gaussian kernel there are well-established tools, such as RBF-QR methods, that allow stable computations of the solutions, see e.g. [20,28,29]. More recently, the Hilbert-Schmidt Singular Value Decomposition (HS-SVD) has been developed [8]. Such technique in principle can be applied to any kernel, provided that the Hilbert-Schmidt eigendecomposition is known. However, the eigenvalues and eigenvectors are far from being easy to compute and in practice are known only for the Gaussian kernel. We finally remark that there are two other classes of stable methods, namely the Weighted SVD (WSVD) and the rescaled-method that properly work with any RBF. The WSVD has the purpose of finding a stable subspace for a given kernel [13,35], while the rescaled-method is based on a proper selection of the supports of CSRBFs so that the ill-conditioning is kept under control [11].

In this paper, in order to guarantee the stability of the solution, as suggested by [40], we carry out Tikhonov regularization [4,45] and preconditioning techniques [16,23,31]. Furthermore, we develop a stable method for the solution of elliptic Boundary Value Problems (BVPs) via Variably Scaled Kernels (VSKs), recently introduced in [2] and further developed in [36,37]. The VSKs depend on a scale function that usually enhances stability and work for any kernel. However, in several cases, the standard scaling gives more accurate results. Thus, taking advantage of the PU scheme, we develop a Hybrid technique (HVSK) such that, on a given subdomain, we use of the standard scaling as long as the conditioning is acceptable, otherwise we switch to VSKs.

The outline of the paper is as follows. In Section 2, we briefly review the main theoretical aspects of the RBF-PU collocation method. Then in Section 3 we present the main computational issues for stably computing the solution of the Poisson problem. In Section 4, we propose an adaptive scheme based on the HVSK

approach. In Section 5 we provide extensive numerical experiments and in the last section we make some conclusions and outline future developments.

2 Elliptic PDEs via RBF-PU collocation methods

For large scale problems, the global RBF-based approach has prohibitive computational costs. Fortunately, the PU method, which leads to moderately sparse matrices, partially overcomes these computational cost issues.

This scheme for computing the solution of elliptic PDEs is reviewed in this section. It is essentially based on Kansa's collocation method, which was introduced by E. Kansa in [26]. Originally it consisted of an unsymmetric scheme, based on multiquadrics, whose convergence properties in the global case were studied only later by R. Schaback (see e.g. [43]).

2.1 RBF-PU method: interpolation and partition of unity

Let Ω be a bounded domain on $\mathbb{R}^M$ and $f : \Omega \to \mathbb{R}$. Given a set of N distinct points $\mathcal{X}_N = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \Omega$, and function values $\mathcal{F}_N = \{f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)\}$, the usual goal in the approximation framework is that of recovering f from the values $\mathcal{F}_N$. To this end, we consider a positive definite and symmetric kernel $\Phi : \Omega \times \Omega \longrightarrow \mathbb{R}$ and define the interpolant $R \in \text{span}\{\Phi(\cdot, \boldsymbol{x}_i), i = 1, \ldots, N\}$ as

$$R(\boldsymbol{x}) = \sum_{k=1}^{N} c_k \Phi(\boldsymbol{x}, \boldsymbol{x}_k), \quad \boldsymbol{x} \in \Omega. \tag{2.1}$$

We take radial kernels and thus we suppose that there exists a function $\phi : [0, \infty) \to \mathbb{R}$ such that for all $\boldsymbol{x}, \boldsymbol{y} \in \Omega$,

$$\Phi(\boldsymbol{x}, \boldsymbol{y}) = \phi(||\boldsymbol{x} - \boldsymbol{y}||_2) := \phi(r).$$

Moreover, the function ϕ may depend on a positive *shape parameter* $\varepsilon > 0$. The role of this parameter is relevant for the accuracy of the whole reconstruction process (see e.g. [18]).

The coefficients $\boldsymbol{c} = (c_1, \ldots, c_N)^T$ in (2.1) are determined by solving the linear system $A\boldsymbol{c} = \boldsymbol{f}$, where $\boldsymbol{f} = (f_1, \ldots, f_N)^T$, $f_i = f(\boldsymbol{x}_i)$, and $A \in \mathbb{R}^{N \times N}$ is the interpolation (or kernel) matrix of entries

$$(A)_{ik} = \Phi(\boldsymbol{x}_i, \boldsymbol{x}_k), \quad i, k = 1, \ldots, N.$$

In the sequel we only focus on strictly positive definite functions. For such functions the interpolation system admits a unique solution.

On real applications we often deal with large data sets and the computational cost of constructing the interpolant via (2.1) becomes prohibitive. Such drawback can be overcome by introducing the PU method [46]. At first, we consider a partition of the open and bounded domain Ω into d subdomains or patches Ω_j, such that $\Omega \subseteq \cup_{j=1}^{d} \Omega_j$, with some mild overlaps among them. In what follows, as patches, we take balls on $\mathbb{R}^M$ of a certain radius δ. The radius is chosen so that the covering property is satisfied.

Together with these subdomains we take a family of compactly supported, non-negative and continuous functions W_j, with $\operatorname{supp}(W_j) \subseteq \Omega_j$ and such that

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

A possible choice is given by the *Shepard's weights* (see [42])

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

where $\tilde{W}_j$ are compactly supported functions on Ω_j.

Once we choose $\{W_j\}_{j=1}^{d}$, the global interpolant is formed by the weighted sum of d local approximants R_j (see e.g. [46])

$$\mathcal{I}(\boldsymbol{x}) = \sum_{j=1}^{d} W_j(\boldsymbol{x}) R_j(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega, \tag{2.2}$$

with

$$R_j(\boldsymbol{x}) = \sum_{k=1}^{N_j} c_k^j \Phi(\boldsymbol{x}, \boldsymbol{x}_k^j),$$

where N_j indicates the number of points on Ω_j and $\boldsymbol{x}_k^j \in \mathcal{X}_{N_j} = \mathcal{X}_N \cap \Omega_j$, with $k = 1, \dots, N_j$.

Hence, the problem of finding the global PU approximant in (2.2) reduces to solving for each Ω_j a linear system with local kernel matrix $A_j \in \mathbb{R}^{N_j \times N_j}$ of entries $(A_j)_{ik} = \Phi(\boldsymbol{x}_i^j, \boldsymbol{x}_k^j)$, $i, k = 1, \dots, N_j$.

Remark 1 In the PU framework, an important computational issue consists in organizing points among the subdomains. To achieve this we use the so-called block-based data structure, refer to [6] or see also [5,14] for further details. Furthermore, we remark that other partitioning data structures, such as kd-trees, are available in literature, see e.g. [17].

To make the paper self-contained we report the convergence theorem for the PU interpolant and later we will discuss the differences arising when the PU is used for collocation. At first, we define two common indicators of data regularity: the *separation distance* and the *fill distance*.

The separation distance is defined as

$$q_{\mathcal{X}_N} := \frac{1}{2} \min_{i \neq k} \|\boldsymbol{x}_i - \boldsymbol{x}_k\|_2,$$

and represents the radius of the largest ball that can be centred at every point on $\mathcal{X}_N = \{\boldsymbol{x}_i, \ i = 1, \dots, N\}$ such that no two balls overlap.

The fill distance is defined as

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

and indicates how well the data fill out the domain Ω. A geometric interpretation of the fill distance is given by the radius of the largest possible empty ball that can be placed among the data locations inside Ω. In particular, it is used in pointwise error bounds like the following one (cf. [47, Theorem 15.19, p. 277]). Such statement is enunciated for strictly positive definite functions, but it can be generalized to the conditionally positive definite case. For further details, the reader can also refer to [46]. Here the aim is the one of stressing the dependence of the interpolation error on the fill distance.

Theorem 1 *Let $C^k_\nu(\mathbb{R}^M)$ be the space of all functions $f \in C^k$ whose derivatives of order $|\boldsymbol{\mu}| = k$ satisfy $D^{\boldsymbol{\mu}} f(\boldsymbol{x}) = \mathcal{O}(||\boldsymbol{x}||_2^\nu)$ for $||\boldsymbol{x}||_2 \longrightarrow 0$. Let $\Omega \subseteq \mathbb{R}^M$ be open and bounded and suppose that $\mathcal{X}_N = \{\boldsymbol{x}_i, i = 1, \dots, N\} \subseteq \Omega$. Let $\phi \in C^k_\nu(\mathbb{R}^M)$ be a strictly positive definite function. Let $\{\Omega_j\}_{j=1}^d$ be a regular covering for $(\Omega, \mathcal{X}_N)$ and let $\{W_j\}_{j=1}^d$ be k-stable for $\{\Omega_j\}_{j=1}^d$. Then the error between $f \in \mathcal{N}_\phi(\Omega)$, where $\mathcal{N}_\phi$ is the native space of ϕ, and its PU interpolant can be bounded by*

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

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

Remark 2 By comparing the results reported in Theorem 1 with the global error estimates shown in [47], one can easily realize that the PU interpolant preserves the local approximation order for the global fit.

2.2 RBF-PU method: PDEs collocation

Given a linear elliptic differential operator $\mathscr{L}$, the goal consists in finding an approximate solution of the BVP problem (Dirichlet boundary conditions)

$$\begin{aligned} \mathscr{L} f(\boldsymbol{x}) &= g_1(\boldsymbol{x}), \quad \text{for } \boldsymbol{x} \in \Omega, \\ f(\boldsymbol{x}) &= g_2(\boldsymbol{x}), \quad \text{for } \boldsymbol{x} \in \partial\Omega. \end{aligned} \tag{2.3}$$

The problem (2.3) is then discretized on a global set of collocation points $\mathcal{X}_N = \mathcal{X}_{N_b} \cup \mathcal{X}_{N_c} = \{\boldsymbol{x}_i,\ i = 1, \dots, N\}$, where N_b and N_c are the number of nodes on $\partial\Omega$ and $\Omega \backslash \partial\Omega$, respectively. Precisely, as done in the majority of papers dealing with elliptic operators on bounded domains, we consider uniformly spaced data on $\partial\Omega$.

Once we assume that (2.3) admits a solution of the form (2.2) then (see e.g. [38, 44]),

$$\begin{aligned} \mathscr{L}\mathcal{I}(\boldsymbol{x}_i) &= \textstyle\sum_{j=1}^d \mathscr{L}(W_j(\boldsymbol{x}_i) R_j(\boldsymbol{x}_i)) = g_1(\boldsymbol{x}_i),\ \boldsymbol{x}_i \in \Omega \backslash \partial\Omega, \\ \mathcal{I}(\boldsymbol{x}_i) &= \textstyle\sum_{j=1}^d W_j(\boldsymbol{x}_i) R_j(\boldsymbol{x}_i) \quad = g_2(\boldsymbol{x}_i),\ \boldsymbol{x}_i \in \partial\Omega. \end{aligned} \tag{2.4}$$

Let $\boldsymbol{R}_j = (R_j(\boldsymbol{x}_1^j), \dots, R_j(\boldsymbol{x}_{N_j}^j))^T$ be the vector of local nodal values. Since the local coefficients $\boldsymbol{c}_j = (c_1^j, \dots, c_{N_j}^j)^T$ are so that $\boldsymbol{c}_j = A_j^{-1} \boldsymbol{R}_j$, we get

$$\mathscr{L}\boldsymbol{R}_j = A_j^{\mathscr{L}} A_j^{-1} \boldsymbol{R}_j, \tag{2.5}$$

where $A_j^{\mathscr{L}}$ is the matrix

$$(A_j^{\mathscr{L}})_{ik} := \mathscr{L}\Phi(\boldsymbol{x}_i^j, \boldsymbol{x}_k^j), \quad i,k = 1,\dots,N_j.$$

To obtain a discrete local operator L_j, we have to differentiate (2.4) by applying a product derivative rule and then use the relation (2.5).

To fix things, consider the Poisson problem, i.e. $\mathscr{L} = -\Delta$. The elliptic operator $\mathscr{L}$ can be expanded to get [23]

$$\begin{aligned}\mathscr{L}(W_j(\boldsymbol{x}_i)R_j(\boldsymbol{x}_i)) = &- \Delta W_j(\boldsymbol{x}_i)R_j(\boldsymbol{x}_i) - 2\nabla W_j(\boldsymbol{x}_i)\cdot\nabla R_j(\boldsymbol{x}_i)\\ &- W_j(\boldsymbol{x}_i)\Delta R_j(\boldsymbol{x}_i), \quad \boldsymbol{x}_i \in \Omega\backslash\partial\Omega,\end{aligned}$$

where the scalar product is applied to the components of the gradients. Let A_j^{Δ} and A_j^{∇} be the matrices with entries

$$(A_j^{\Delta})_{ik} = \Delta\Phi(\boldsymbol{x}_i^j, \boldsymbol{x}_k^j), \quad i,k = 1,\dots,N_j,$$

and

$$(A_j^{\nabla})_{ik} = \nabla\Phi(\boldsymbol{x}_i^j, \boldsymbol{x}_k^j), \quad i,k = 1,\dots,N_j,$$

we have

$$\Delta\boldsymbol{R}_j = A_j^{\Delta}\boldsymbol{c}_j = A_j^{\Delta}A_j^{-1}\boldsymbol{R}_j.$$

Furthermore we consider the matrix

$$\bar{W}_j^{\Delta} = \operatorname{diag}\left(\Delta W_j(\boldsymbol{x}_1^j),\dots,\Delta W_j(\boldsymbol{x}_{N_j}^j)\right),$$

and similarly we define $\bar{W}_j^{\nabla}$ and $\bar{W}_j$. Finally, by including the boundary conditions, we can express the discrete local Laplacian as

$$(L_j)_{ik} = \begin{cases}(\bar{L}_j)_{ik}, & \text{for } \boldsymbol{x}_i^j \in \Omega\backslash\partial\Omega,\\ \delta_{ik}, & \text{for } \boldsymbol{x}_i^j \in \partial\Omega,\end{cases}$$

where δ_{ik} denotes the Kronecker delta and

$$\bar{L}_j = \left(\bar{W}_j^{\Delta}A_j + 2\bar{W}_j^{\nabla}\cdot A_j^{\nabla} + \bar{W}_j A_j^{\Delta}\right)A_j^{-1}. \tag{2.6}$$

In what follows we will refer to the collocation method described in this section as the RBF Standard approach (RBF-S), meaning that the standard basis is used to approximate the solution.

Note that, since we use the Laplacian, we require that both the kernel and the weight functions are at least twice differentiable. Let $\boldsymbol{x}_{\zeta_{kj}} \in \mathcal{X}_{N_j}$ be the node corresponding to $\boldsymbol{x}_k \in \mathcal{X}_N$. In order to obtain the global discrete PDE operator, we need to assemble the local ones into a global matrix L as follows

$$(L)_{ik} = \sum_{j=1}^{d}(L_j)_{\zeta_{ij}.\zeta_{kj}}, \quad i,k = 1,\dots,N.$$

Then, we simply have to solve the (possibly ill-conditioned) system

$$L\boldsymbol{z} = \boldsymbol{f}, \tag{2.7}$$

where $\boldsymbol{z} = (\mathcal{I}(\boldsymbol{x}_1),\dots,\mathcal{I}(\boldsymbol{x}_N))^T$ and $\boldsymbol{f} = (f_1,\dots,f_N)^T$, with

$$f_i = \begin{cases}g_1(\boldsymbol{x}_i), & \text{for } \boldsymbol{x}_i \in \Omega\backslash\partial\Omega,\\ g_2(\boldsymbol{x}_i), & \text{for } \boldsymbol{x}_i \in \partial\Omega,\end{cases} \quad i = 1,\dots,N.$$

Remark 3 The main advantage of PU collocation is the computational efficiency in constructing the collocation matrix. However, we have to discuss several drawbacks concerning its well-posedness. In general, among meshfree global collocation methods, the symmetric ones should be preferred because they guarantee the existence and uniqueness of the solution. For Kansa's collocation approaches instead, the system might be singular [24] and its uniqueness can be ensured only under several restrictions, which in particular lead to distinguish between collocation points and RBF centres [32]. The non-symmetry of the matrix L suggests that its non-singularity could be ensured only with a similar distinction between RBF centres and collocation data. This needs further investigations. Alternatively, one can also use the least squares approach proposed in [30].

3 RBF-PU collocation: stability issues

For several choices of the shape parameter the solution of an elliptic PDE via PU collocation might be inaccurate. Indeed, we can easily note that in (2.6) multiplying by the inverse of the local matrix A_j might lead to instability that is carried over to the global collocation matrix L. We already pointed out in the introduction that RBF-QR methods effectively solve such problem, but we are here interested in stably computing the solution for any kernel. We consider two approaches: Tikhonov regularization, which is also well-known in the context of neural networks [4], and VSKs [2]. The former, described in the next subsection, gives acceptable results but we show that it is outperformed by the use of our hybrid procedure based on VSKs, which will be presented in Subsection 3.2.

3.1 Tikhonov SVD regularization (TSVD)

As we noticed, the final collocation system (2.7) could be severely ill-conditioned. A stable solution of (2.7) can be found by the Tikhonov regularization method [45], which essentially gives a penalized least square solution. We denote by $\tilde{\boldsymbol{z}}$ the solution depending on the *Tikhonov matrix* Γ given by

$$\min_{\boldsymbol{z}} \left(||L\boldsymbol{z} - \boldsymbol{f}||_2^2 + ||\Gamma \boldsymbol{z}||_2^2 \right). \tag{3.1}$$

The minimum is

$$\tilde{\boldsymbol{z}} = (L^T L + \Gamma^T \Gamma)^{-1} L^T \boldsymbol{f}, \tag{3.2}$$

and the penalty term $||\Gamma \boldsymbol{z}||_2^2$ in (3.1) is designed to improve the stability, hence making the problem less sensitive to ill-conditioning. According to [16,40], in what follows, we consider $\Gamma = \sqrt{\gamma} I$, with $\gamma > 0 \in \mathbb{R}$ (see also Section 5 for further details).

In [22], it has been proved that (3.2) can be expressed as

$$\tilde{\boldsymbol{z}} = V\, D\, U^T \boldsymbol{f},$$

where V and U come from the SVD of L, i.e. $L = U \Sigma V^T$ and Σ is the diagonal matrix of the singular values σ_i of L. Then, the entries of the diagonal matrix D are given by

$$d_i = \frac{\sigma_i}{\sigma_i^2 + \gamma},$$

$i = 1, \dots, N$. Note that, also when the local matrices (especially A_j) are severely ill-conditioned, the Tikhonov regularization only acts on the final system.

3.2 Hybrid Variably Scaled Kernels (HVSK)

Unlike Tikhonov regularization, the HVSK approach enables us to intervene on the local discrete operators, producing truly more accurate and stable solutions, as numerically shown in Section 5.

VSKs were introduced in [2] and the main idea behind their definition is to consider the shape parameter as an *additional coordinate* the kernel depends on. That is, the scale parameter is considered as a continuous function. More precisely, we can define a VSK as follows (cf. [2, Def. 2.1]).

Definition 1 Let $\psi : \mathbb{R}^M \to (0, \infty)$ be a given scale function. A Variably Scaled Kernel (VSK) K_ψ on $\mathbb{R}^M$ is

$$K_\psi(\boldsymbol{x}, \boldsymbol{y}) := \mathcal{K}((\boldsymbol{x}, \psi(\boldsymbol{x})), (\boldsymbol{y}, \psi(\boldsymbol{y}))), \quad \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^M, \tag{3.3}$$

where $\mathcal{K}$ is a kernel on $\mathbb{R}^{M+1}$.

It is easy to show that if in (3.3) $\mathcal{K}$ is positive definite on $\mathbb{R}^{M+1}$, so the VSK K_ψ is on $\mathbb{R}^M$ and thus the VSK interpolant is uniquely defined.

In particular the local VSK interpolant can be defined as follows.

Definition 2 Given the set of points $\mathcal{X}_{N_j} = \{\boldsymbol{x}_i^j,\ i = 1, \dots N_j\}$ on the subdomain Ω_j and the (local) scale function $\psi_j : \mathbb{R}^M \to (0, \infty)$, then the local VSK interpolant is

$$R_{\psi_j}(\boldsymbol{x}) = \sum_{k=1}^{N_j} c_k^j \mathcal{K}((\boldsymbol{x}, \psi_j(\boldsymbol{x})), (\boldsymbol{x}_k^j, \psi_j(\boldsymbol{x}_k^j))), \quad \boldsymbol{x} \in \Omega_j.$$

Furthermore, ψ_j defines a function

$$\Psi_j : \boldsymbol{x} \mapsto (\boldsymbol{x}, \psi_j(\boldsymbol{x})),$$

which maps the space $\mathbb{R}^M$ into a M-dimensional submanifold $\Psi_j(\mathbb{R}^M)$ of $\mathbb{R}^{M+1}$ and the set of nodes $\mathcal{X}_{N_j} \subset \Omega_j \subset \mathbb{R}^M$ into $\Psi_j(\mathcal{X}_{N_j}) \subset \Psi_j(\Omega_j) \subset \Psi_j(\mathbb{R}^M) \subset \mathbb{R}^{M+1}$. As a consequence, the interpolation by the kernel $\mathcal{K}$ takes place on $\mathbb{R}^{M+1}$ at the transformed points set $\Psi_j(\mathcal{X}_{N_j})$.

For the interpolation setting, in [2], the authors prove that the error and stability analysis of the VSK on $\mathbb{R}^M$ coincides with that of a fixed-scale problem on a submanifold on $\mathbb{R}^{M+1}$. In other words, referring to Theorem 1, we know that for VSKs the interpolation error depends on the fill distance as well. Therefore, in order to have a better understanding of the error analysis we only need to discuss how the fill distance changes when VSKs are used. Indeed, both the fill distance and separation distance will be transformed by Ψ_j and will roughly be multiplied by a factor that scales with the norm of the gradient of ψ_j or the Lipschitz constant ℓ of ψ_j, depending on the regularity of ψ_j. Indeed

$$\|\Psi_j(\boldsymbol{x}) - \Psi_j(\boldsymbol{y})\|_2^2 = \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 + (\psi_j(\boldsymbol{x}) - \psi_j(\boldsymbol{y}))^2 \leq \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 (1 + \ell)^2,$$

so that

$$\|\Psi_j(\boldsymbol{x}) - \Psi_j(\boldsymbol{y})\|_2^2 \geq \|\boldsymbol{x} - \boldsymbol{y}\|_2^2,$$

which shows that distances will blow up with Ψ_j, letting separation distance never decrease and improving the stability of the process. In fact, the ill-conditioning grows with the decrease of the separation distance and not necessarily with the increase of the number of data points. Unfortunately, also the fill distance, which is a measure of the interpolation error (see Theorem 1), grows. For this reason the new idea of the HVSK approach.

With particular scale functions the conditioning of the kernel matrix can be sensibly reduced [2]. However, aside the case in which noise is introduced, this might lead to a decrease of the accuracy of the solution with respect to the standard basis. In practice, we will see that the error via VSKs is usually higher than the one that can be found with the optimal shape parameter ε_{opt}, that is the reason why we propose the use of HVSK technique. The idea is as follows:

- take a constant shape parameter ε,
- compute the local kernel matrix A_j on a subdomain Ω_j,
- use the scaling with ε and standard bases as long as A_j is not close to be singular, i.e. as long as the conditioning is acceptable.

Concerning the last step, we check if the conditioning is acceptable by fixing a threshold and applying the VSKs if and only if the smallest singular value of A_j is below the prescribed tolerance. To this end we compute the SVD of the local matrices. The computational cost is affordable, being such matrices of small size.

For the collocation via VSKs we can prove the following result.

Proposition 1 *Let us consider a radial function $\phi(\cdot)$ at least twice differentiable associated to the VSK, $\mathcal{K}$. Letting*

$$\omega_\alpha = (x_{i\alpha}^j - x_{k\alpha}^j) + (\psi_j(\boldsymbol{x}_i^j) - \psi_j(\boldsymbol{x}_k^j))\frac{\partial \psi_j(\boldsymbol{x}_i^j)}{\partial x_{i\alpha}^j}, \quad \alpha = 1, \ldots, M,$$

$$d_{\omega_\alpha} = 1 + \left(\frac{\partial \psi_j(\boldsymbol{x}_i^j)}{\partial x_{i\alpha}^j}\right)^2 + (\psi_j(\boldsymbol{x}_i^j) - \psi_j(\boldsymbol{x}_k^j))\frac{\partial^2 \psi_j}{\partial^2 x_{i\alpha}^j}(\boldsymbol{x}_i^j), \quad \alpha = 1, \ldots, M,$$

and

$$\rho = \left(||\boldsymbol{x}_i^j - \boldsymbol{x}_k^j||_2^2 + (\psi_j(\boldsymbol{x}_i^j) - \psi_j(\boldsymbol{x}_k^j))^2\right)^{1/2},$$

then the entries of the local VSK differentiation matrices for the BVP (2.3) with $\mathscr{L} = -\Delta$ are given by

$$(A_{\psi_j}^{\nabla})_{ik} = \left((A_{\psi_j}^1)_{ik}, \ldots, (A_{\psi_j}^M)_{ik}\right) = \left(\frac{\omega_1}{\rho}\frac{d\phi(\rho)}{d\rho}, \ldots, \frac{\omega_M}{\rho}\frac{d\phi(\rho)}{d\rho}\right), \tag{3.4}$$

$i, k = 1, \ldots, N_j$, and

$$(A_{\psi_j}^{\Delta})_{ik} = \sum_{\alpha=1}^{M}\left[\frac{d^2\phi(\rho)}{d\rho^2}\frac{\omega_\alpha^2}{\rho^2} + \frac{d\phi(\rho)}{d\rho}\left(\frac{d_{\omega_\alpha}}{\rho} - \frac{\omega_\alpha^2}{\rho^3}\right)\right], \quad i, k = 1, \ldots, N_j. \tag{3.5}$$

Proof From [2], we know that if the VSK $\mathcal{K}$ is radial then it can be seen as a univariate function $\phi = \phi(\rho)$. Thus, the entries of the associated kernel matrix A_{ψ_j} are given by

$$(A_{\psi_j})_{ik} = \phi\left(\left(||\boldsymbol{x}_i^j - \boldsymbol{x}_k^j||_2^2 + (\psi_j(\boldsymbol{x}_i^j) - \psi_j(\boldsymbol{x}_k^j))^2\right)^{1/2}\right), \quad i,k = 1,\dots,N_j.$$

Moreover, since

$$\frac{\partial \phi(\rho)}{\partial x_{i\alpha}^j} = \frac{d\phi(\rho)}{d\rho}\frac{\partial \rho}{\partial x_{i\alpha}^j}, \quad \alpha = 1,\dots,M\,,$$

and

$$\frac{\partial \rho}{\partial x_{i\alpha}^j} = \frac{1}{\rho}\left[(x_{i\alpha}^j - x_{k\alpha}^j) + (\psi_j(\boldsymbol{x}_i^j) - \psi_j(\boldsymbol{x}_k^j))\frac{\partial \psi_j(\boldsymbol{x}_i^j)}{\partial x_{i\alpha}^j}\right],$$

$\alpha = 1,\dots,M$, then (3.4) easily follows. The same argument shows that

$$\frac{\partial^2 \rho}{\partial^2 x_{i\alpha}^j} = \frac{1}{\rho}\left[1 + \left(\frac{\partial \psi_j(\boldsymbol{x}_i^j)}{\partial x_{i\alpha}^j}\right)^2 + (\psi_j(\boldsymbol{x}_i^j) - \psi_j(\boldsymbol{x}_k^j))\frac{\partial^2 \psi_j}{\partial x_{i\alpha}^j}\right] - \frac{\omega_\alpha^2}{\rho^3}, \quad \alpha = 1,\dots,M.$$

Finally from the fact that

$$\frac{\partial^2 \phi(\rho)}{\partial^2 x_{i\alpha}^j} = \frac{d^2\phi(\rho)}{d\rho^2}\frac{\omega_\alpha^2}{\rho^2} + \frac{d\phi(\rho)}{d\rho}\frac{\partial^2 \rho}{\partial^2 x_{i\alpha}^j}, \quad \alpha = 1,\dots,M,$$

(3.5) follows. □

From this result we obtain that the discrete local operator based on the VSKs (2.6) takes the form

$$\bar{L}_{\psi_j} = \left(\bar{W}_j^{\Delta} A_{\psi_j} + 2\bar{W}_j^{\nabla} \cdot A_{\psi_j}^{\nabla} + \bar{W}_j A_{\psi_j}^{\Delta}\right) A_{\psi_j}^{-1}.$$

We will numerically show that, by collocating via HVSKs, we are able to provide stable approximations for the solution of elliptic PDEs. However, a theoretical analysis of the error via HVSKs for collocation schemes needs further investigations. This is a consequence of the more general fact that there are no trivial extensions of Theorem 1 for the collocation setting. Indeed, its proof is based on bounding the global error in function of the local ones. This can be done since for each patch we have a well-posed interpolation problem. On the opposite, in the collocation setting, we do not have well-posed local collocation problems, in fact interior patches have no boundary conditions.

4 RBF-PU collocation: the Adaptive Residual Subsampling scheme

Dealing with adaptivity, the two main computational issues concern the stability, related to the oversampling of certain regions of high variation of the solution, and the choice of the data sets. In view of the considerations made in the previous section, we will use the HVSK technique to enhance the stability. This will be more evident in Section 5, in which numerical tests will show that the HVSK approach

performs better than TSVD, improving the stability of the RBF-standard method (RBF-S).

Concerning the data sets, differently from what is usually done in literature (see e.g. [15]), we do not consider grid data. Our aim in fact is to obtain a method that, at the same time, works with well distributed nodes and is easy to implement on different geometries of the hypercube $[0,1]^M$. Grid data are not extremely suitable, thus we take and compare two kinds of data sets: the classical low-discrepancy Halton nodes and the greedy points. Both are generated as sequences of points. The latter have the advantage of being *similar* to grid data, in the sense that they provide a set of quasi-uniform points in the Euclidean distance (see [10,39]). Moreover, such points are independent of the basis function ϕ. This observation suggests an algorithm which is based only on geometric considerations and that allows to generate a similar set of points as a sequence. More precisely, the *geometric greedy points* are generated as follows:

- Choose $\boldsymbol{z}_0$ on $\partial\Omega$ and let $\mathcal{Z}_0 := \{\boldsymbol{z}_0\}$.
- Let $\mathcal{Z}_i \subset \Omega$, $\boldsymbol{z}_{i+1} := \max_{\boldsymbol{z}\in\Omega\setminus\mathcal{Z}_i} \mathtt{dist}(\boldsymbol{z}, \mathcal{Z}_i)$.

where `dist` in our setting is the Euclidean distance. As a remark, in [3] the authors point out that this algorithm can generate equidistant points on compact sets of $\mathbb{R}^M$ with respect to a generic metric.

On Ω let us consider the set $\mathcal{X}_{N^{(1)}} \equiv \mathcal{X}_N = \{\boldsymbol{x}_i^{(1)},\ i = 1,\dots,N^{(1)}\}$. Along the boundary we take the set of points $\mathcal{X}_{N_b^{(1)}}$ with

$$N_b^{(1)} = (\hat{N}+2)^M - N_c^{(1)}, \quad \text{where} \quad \hat{N} = \lceil N_c^{(1)} \rceil^{1/M} \tag{4.1}$$

This choice follows from the fact that if $N^{(1)}$ grid data are taken on $[0,1]^M$ then exactly $(\hat{N}+2)^M - N_c^{(1)}$ lie on the boundary.

Our ARS strategy is based on the residual subsampling technique proposed in [15]. At the first step, the ARS schemes defines a set of interior points

$$\mathcal{Y}_{\tilde{N}^{(1)}} \equiv \mathcal{Y}_{\tilde{N}_c^{(1)}} = \{\boldsymbol{y}_i^{(1)},\ i = 1,\dots,\tilde{N}^{(1)}\},$$

and takes

$$\tilde{N}^{(1)} = \left\lceil pN_c^{(1)} \right\rceil,\ p \in \mathbb{R}^+.$$

ARS algorithm.

Let τ_1 and τ_2 be two tolerances, $\tau_2 < \tau_1$.

(a) Consider the initial set of nodes and compute the solution on $\mathcal{X}_{N^{(1)}}$. Then, evaluate the residuals

$$r_i^{(1)} = \left| f\left(\boldsymbol{y}_i^{(1)}\right) - \mathcal{I}_{N^{(1)}}\left(\boldsymbol{y}_i^{(1)}\right) \right|, \quad i = 1,\dots,\tilde{N}^{(1)}.$$

where $\mathcal{I}_{N^{(1)}}$ is the approximate solution. We then define

$$\mathcal{S}_{T_1^{(1)}} = \{\boldsymbol{y}_i^{(1)} : r_i^{(1)} > \tau_1,\ i = 1,\dots,T_1^{(1)}\},$$

and

$$\mathcal{S}_{T_2^{(1)}} = \{\bar{\boldsymbol{x}}_i^{(1)} : r_i^{(1)} < \tau_2,\ i = 1,\dots,T_2^{(1)}\},$$

where $\bar{\boldsymbol{x}}_i^{(1)}$ is the nearest point to $\boldsymbol{y}_i^{(1)}$ and $T_1^{(1)}$ and $T_2^{(1)}$ simply identify the cardinality of the two sets.

(b) At the $(k+1)$-th step, we take the following new discretization nodes

$$\mathcal{X}_{N^{(k+1)}} = \mathcal{X}_{N_b^{(k+1)}} \cup \mathcal{X}_{N_c^{(k+1)}},$$

where

$$\mathcal{X}_{N_c^{(k+1)}} = \left(\mathcal{X}_{N_c^{(k)}} \cup \mathcal{S}_{T_1{}^{(k)}}\right) \setminus \mathcal{S}_{T_2{}^{(k)}},$$

and again $\mathcal{X}_{N_b^{(k+1)}}$ is constructed so that (4.1) is fulfilled.

We define the $(k+1)$-th training set of interior nodes by taking firstly $\bar{N}_c^{(k+1)}$ points on Ω with $\bar{N}^{(k+1)} = \left\lceil pN_c^{(k+1)} \right\rceil$. Hence

$$\mathcal{Y}_{\tilde{N}^{(k+1)}} = \mathcal{Y}_{\tilde{N}_c^{(k+1)}} \cup \mathcal{S}_{T_2{}^{(k)}} = \{\boldsymbol{y}_i^{(k+1)},\ i = 1, \dots, \tilde{N}^{(k+1)}\}.$$

In this way we are also able to eventually remove nodes of the previously computed data set.

(c) Stop when $\mathcal{S}_{T_1^{(k)}} = \emptyset$.

Notes.

- If all residuals are greater than the chosen threshold, the number of points is doubled at each iteration. Therefore after several steps, we expect that only few residuals are greater than the chosen threshold and moreover that they are located only where the solution varies faster. Nevertheless, the algorithm computes a large initial set of test points and consequently large evaluation matrices, even if in the end only few test nodes become new centres for the basis functions. To avoid this drawback, if at the k-th step $\texttt{card}(\mathcal{S}_{T_1{}^{(k)}}) < aN_c^{(k+1)}$, for a certain $a < 1$, we define a reduced number of check points. Specifically, for each data belonging to $\mathcal{S}_{T_1{}^{(k)}}$ we compute its k nearest neighbours with respect to the set $\mathcal{X}_{N^{(k+1)}}$ and we define $\boldsymbol{y}_i^{(k+1)}$ as a greedy point on this neighbourhood. This is an advantage with respect to the method proposed in [15]. Indeed, in the last mentioned paper, even if only few points are added, the training sets always consist of large grids. Obviously, this is computationally expensive because it leads to evaluate many residuals (by constructing useless large evaluation matrices). Hence, the use of greedy points produces a benefit for the computational cost of the algorithm.
- If the analytical form of the PDE is not known, by defining several subsets of the original one, it comes easy to identify a training set at each step. The criterion we use here to select new centres is based on residual subsampling for which the solution is supposed to be known. Alternatively, following the suggestions provided by [9], a point $\boldsymbol{y}_i^{(k)}$ becomes a new centre if

$$|\mathcal{I}_{N^{(k)}}(\boldsymbol{y}_i^{(k)}) - \mathcal{I}(\bar{\boldsymbol{x}}_i^{(k)})|, \tag{4.2}$$

is greater than a prescribed tolerance, where $\bar{\boldsymbol{x}}_i^{(k)} \in \mathcal{X}_{N_c^{(k)}}$ is its nearest point. We can think of

$$\frac{|\mathcal{I}_{N^{(k)}}(\boldsymbol{y}_i^{(k)}) - \mathcal{I}_{N^{(k)}}(\bar{\boldsymbol{x}}_i^{(k)})|}{||\boldsymbol{y}_i^{(k)} - \bar{\boldsymbol{x}}_i^{(k)}||_2},$$

as an estimate of the directional derivative of the solution in $\boldsymbol{y}_i^{(k)}$ (cf. [34]). Also in [34], the reader can find other useful criteria.

5 Numerical experiments

Experiments are carried out on an Intel(R) Core(TM) i7 CPU 4712MQ 2.13 GHz processor. The software is available for the scientific community and can be freely downloaded at `http://www.math.unipd.it/~demarchi/RBF/HVSK_PU.zip`.

In this section we firstly show the benefits of the HVSK approach, comparing it with TSVD, RBF-S and RBF-QR methods. Then, we will present numerical experiments to test the ARS technique which is based on the HVSK method. About the RBF-QR, we use the MATLAB code downloadable at `http://www.it.uu.se/research/scientific_computing/project/rbf/software`.

In what follows, the space dimension is $M = 2$, the Root Mean Square Error (RMSE) is computed on a grid of $s = 40 \times 40$ points $\tilde{\boldsymbol{x}}_i$, $i = 1, \ldots, s$

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

and we also evaluate the 2-norm Condition Number (CN) of the collocation matrix L.

Following [2], for the HVSK technique on Ω_j we consider the scale function

$$\psi_j(\boldsymbol{x}) = \sum_{i=1}^{N_j} |p_i^j(\boldsymbol{x}, \boldsymbol{x}_i^j)|,$$

with

$$p_i^j(\boldsymbol{x}, \boldsymbol{x}_i^j) = \frac{1}{\pi}\arctan(h_i^j(x_1 - x_{i1}^j))e^{-5(x_2 - x_{i2}^j)},$$

where $\boldsymbol{x} = (x_1, x_2)$, $\boldsymbol{x}_i^j = (x_{i1}^j, x_{i2}^j) \in \Omega_j$ and $h_i^j \in \mathbb{R}^+$, $i = 1, \ldots, N_j$. From extensive numerical experiments, we found reliable results when h_i^j assumes *small* values. The function p_i^j increases more rapidly if h_i^j is large. Therefore we look for larger values of h_i^j when the points are *clustered*. Thus, a possible choice is

$$h_i^j = \frac{q_{N_j}}{s_i^j} 10^{-5},$$

where q_{N_j} is the separation distance of the j-th subdomain and s_i^j is the distance between $\boldsymbol{x}_i^j \in \Omega_j$ and its nearest point on the j-th patch. Nevertheless, this choice is computationally expensive and a nearest neighbour procedure must be applied. That is why here we fix $h_i^j = 7 \cdot 10^{-6}$ for all $i = 1, \ldots, N_j$, and $j = 1, \ldots, d$.

5.1 Stability issues

In order to test the HVSK method, we consider an elliptic problem on $\Omega = [0, 1]^2$ with a manufactured solution from which we can easily compute the functions g_1 and g_2 of the Poisson problem (2.4). In particular we take

$$f_1(x_1, x_2) = \sin(x_1 + 2x_2^2) - \sin(2x_1^2 + (x_2 - 0.5)^2).$$

Experiments are performed considering several sets of Halton nodes on the unit square Ω. For the PU weights we take the Wendland's C^2 function

$$\tilde{W}(r) = (1-\varepsilon r)_+^4(4\varepsilon r + 1),$$

where $(\cdot)_+$ denotes the truncated power function and $\varepsilon > 0$.

As radial function we consider the Gaussian kernel

$$\phi(r) = \mathrm{e}^{-\varepsilon^2 r^2}.$$

We show the results obtained by means of both TSVD and HVSK, by computing the RMSEs and CNs for 20 values of the shape parameter ε, uniformly spaced in logarithmic scale in the range $[10^{-3}, 10^2]$. For a suitable selection of the Tikhonov parameter γ we refer to [16,40]. We have found good results for $\gamma \in [10^{-15}, 10^{-10}]$. In the numerical experiments that follow, we have actually selected the optimal value, say γ^*, via *trials and errors*.

We test our method on $N_c = 81, 289, 1089, 4225$, Halton data and N_b boundary points as in (4.1). Finally, we also need to fix the number of patches and related radius. The former should be chosen proportionally to the number of points N, while the latter must be chosen so that subdomains form a covering of Ω. To fulfill such properties, we select the number of patches d such that (see e.g. [6])

$$d = \left\lfloor \frac{\sqrt[M]{N_c}}{2^{M-1}} \right\rfloor^M,$$

and the radius δ as follows

$$\delta = \left(\frac{2}{d}\right)^{1/M}. \tag{5.1}$$

In Table 1 we provide fill and separation distances for the VSK compared with the ones of the original data set. For the VSK approach, these quantities correspond to the fill and separation distances of the original data set mapped via the scale function. As expected, both distances grow, leading to a more stable scheme in the VSK case (due to the increase of the separation distance) that however might cause a decrease of the accuracy (due to the increase of the fill distance that thanks to the choice of ψ_j is moderate). In this framework, the use of the mixed technique results particularly meaningful.

In Figures 1 and 2 we respectively compare the RMSEs and CNs obtained by means of TSVD and HVSK with those of the RBF-S and RBF-QR methods. In Table 2 we also report the corresponding CPU times for $\varepsilon = 10^{-3}$. Note that, both the HVSK and RBF-QR methods are comparable with the computational cost of the standard bases. Indeed, the only difference for HVSK consists in defining and evaluating the scale function. Furthermore, even if the measured CPU times are slightly different, when the methods execute the same routine, in Table 2 we report the same CPU times to avoid confusion.

We now need to discuss when, instead of the standard bases depending on ε, VSKs should be applied. Here, VSKs are used on a subdomain Ω_j if and only if σ_m, i.e. the minimum of the singular values associated to A_j, is such that $\sigma_m < 10^{-16}/\varepsilon^4$. This tolerance has been validated only numerically on different test cases. Being dependent on the shape parameter, it means that for small

shape parameters, i.e. when usually the instability becomes more evident, VSKs are almost always applied. This allows to overcome the instability issues and at the same time to recover the optimal solution given by the standard bases. Furthermore, one can use the VSKs also for *large* shape parameters. In those cases the conditioning with standard bases is always acceptable but usually the accuracy of the methods gets worse. Thus, in this example VSKs are also applied when $\sigma_m > 10^{-11}$.

From Figure 1, we note that both RBF-QR and HVSK outperform the other approaches. Moreover, as expected, the RBF-QR method gives more accurate and stable results than any other technique considered here. Nevertheless, we remark that VSKs are independent of the choice of the kernel, while the RBF-QR approaches are based on the Gaussian kernel. In applications or in the adaptivity framework, when points are clustered, this is an advantage for HVSK. In those cases, the use of smooth functions, as the Gaussian, is not recommended. On the opposite, kernels with limited regularities are strongly advised.

The condition numbers plotted in Figure 2 are coherent with the fact that HVSK and RBF-QR are more stable. However, the difference of the conditioning of the HVSK and of the RBF-S is not always appreciable. Nevertheless, the errors show that, differently from the standard bases, the HVSK approach is able to moderately reduce the conditioning and, as a consequence, provides stable solutions. On the contrary, even if TSVD sensibly diminishes the condition number, the results are not completely satisfactory in terms of accuracy. This is due to the fact that we do not intervene on the computation of (2.6). Thus, the instability due to the evaluation of the inverse of the kernel matrix is carried over to the final system.

We also point out that, from other numerical experiments here omitted, we note that HVSK performs better than other methods based on Tikhonov approaches, such as computations based on Riley's algorithm [16], or the one proposed in [40]. However, we have to mention that differently from [40], we do not employ multiple precision.

Remark 4 One may argue that, to achieve both accuracy and efficiency, there is no need to use stable methods, but selecting the optimal shape parameter would be sufficient. Unfortunately, there are no a priori optimal choices for its value and one

N_c	data set	$h_{\mathcal{X}_N}$	$q_{\mathcal{X}_N}$
81	original	1.03E − 01	1.07E − 02
	mapped	2.93E − 01	3.68E − 02
289	original	5.72E − 02	2.07E − 03
	mapped	6.34E − 02	6.73E − 03
1089	original	3.27E − 02	1.12E − 03
	mapped	5.79E − 02	4.34E − 03
4225	original	1.68E − 02	1.74E − 04
	mapped	4.85E − 02	6.79E − 04

Table 1: Separation and fill distances of the original data set compared with the ones mapped via VSKs.

always needs to use very costly techniques, such as cross-validation or maximum likelihood method (see e.g. [17] for a general overview).

Concerning the method used to solve the final collocation system (2.7), we take into account both direct and iterative methods. Numerically, we observed that the direct one (computed with the standard `mldivide.m` MATLAB function) is more effective in terms of efficiency. In Table 2 we compare these results with the `gmres.m` routine that takes an incomplete LU factorization of L as preconditioner. Moreover, differently from [23], we also note that the use of preconditioners does not produce a significant regularization of the solution. Finally, we quote the fact that no iterative methods can be used for the case of TSVD which requires high complexity costs due to the computation of SVDs of the potentially final large matrices.

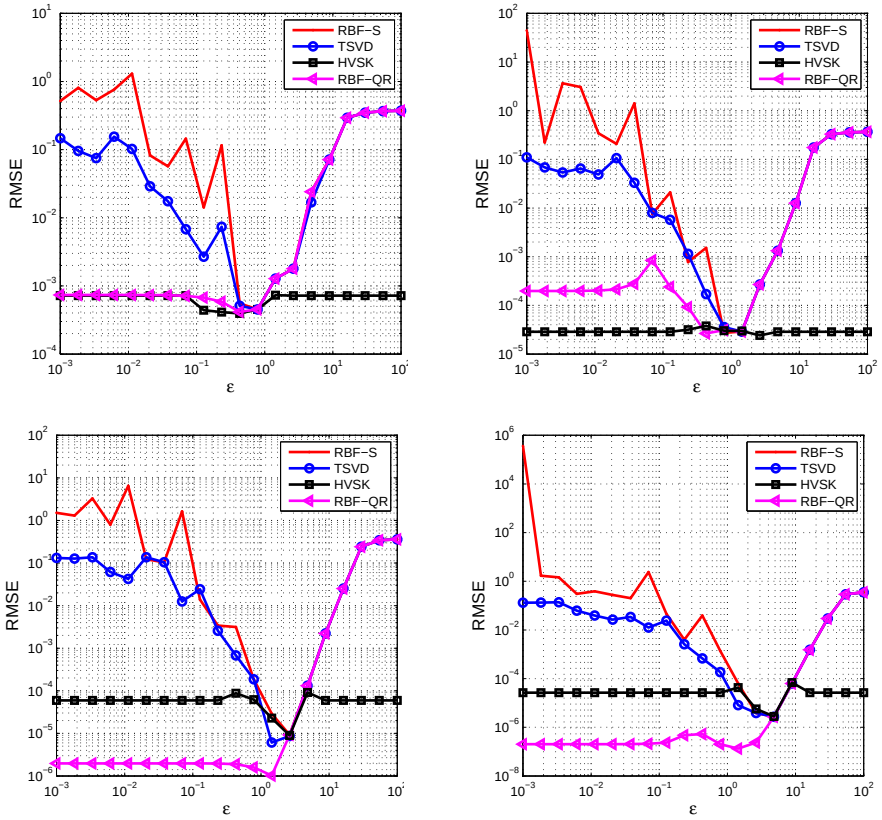

Fig. 1: RMSEs obtained by varying ε for the Gaussian C^∞ kernel. From left to right, top to bottom, we consider $N_c = 81$, 289, 1089 and 4225 Halton data.

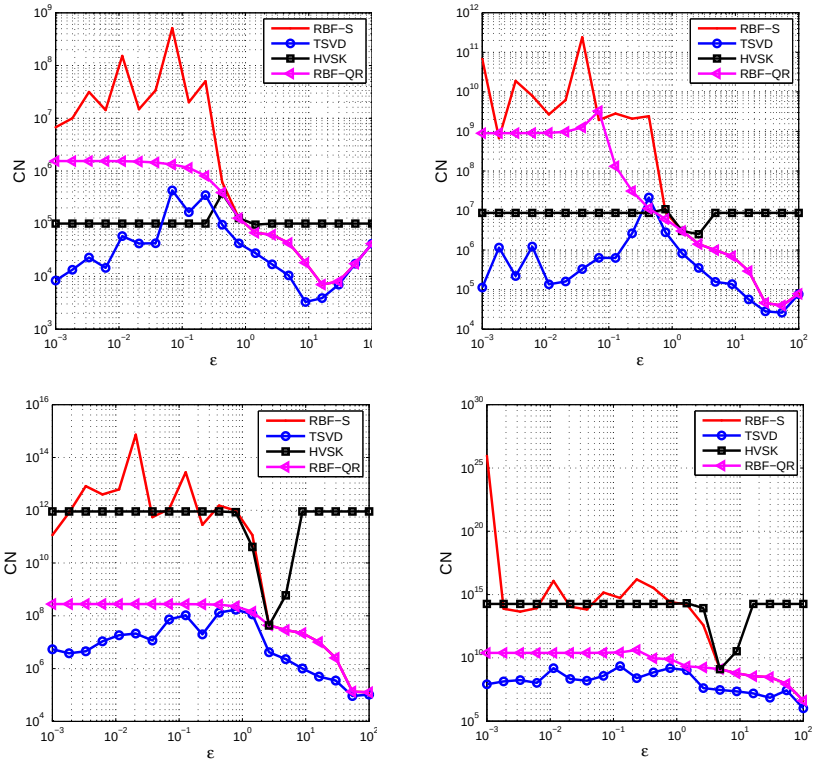

Fig. 2: Condition numbers of the matrix L obtained by varying ε for the Gaussian C^∞ kernel. From left to right, top to bottom, we consider $N_c = 81$, 289, 1089 and 4225 Halton data.

5.2 Adaptive residual subsampling scheme

We test the adaptive method based on the HVSK technique on three Poisson problems on $\Omega \subseteq [0,1]^2$ with known solutions:

$$f_2(x_1, x_2) = \frac{1}{20} \mathrm{e}^{4x_1} \cos(2x_1 + x_2),$$

$$f_3(x_1, x_2) = \frac{1}{2} x_2 [\cos(4x_1^2 + x_2^2 - 1)]^4 + \frac{1}{4} x_1,$$

and

$$f_4(x_1, x_2) = \mathrm{e}^{-8((x_1-0.5)^2+(x_2-0.05)^2)}.$$

Note that f_2 is quite easy to approximate while the main difficulties are in solving the elliptic problem with f_3, due to its oscillations (see e.g. [1]). The function f_4 is the gaussian peak function.

Nevertheless, we will point out that, also for the simplest test function f_2, the use of the HVSK approach is essential to ensure a numerical convergence of the ARS scheme.

In these cases we consider the Matérn C^6 radial function

$$\phi(r) = \mathrm{e}^{-\varepsilon r}(\varepsilon^3 r^3 + 6\varepsilon^2 r^2 + 15\varepsilon r + 15).$$

Moreover, we take $p = 1$, $a = 1/10$, $K = 1 + 2k$, $\tau_1 = 10^{-5}$ and $\tau_2 = 10^{-9}$.

We start with a data set consisting of $N_c = 100$ points on Ω and $d = N_c$. The radius of patches is set as in (5.1). Usually, the number of subdomains d is chosen so that $N/d \approx 2^M$ and since here d is kept fixed along the iterations, the subdomains are more and more filled out by points, i.e.

$$N^{(k)}/d \geq N^{(k-1)}/d, \quad k \geq 2.$$

For the test function f_2 we take Halton points and $\varepsilon = 0.3$. The first steps of the algorithm are plotted in Figure 3. The scheme successfully stops with a data set consisting of 1044 points, as displayed in Figure 4. Note that in the end, the subdomain having the largest number of points contains 100 data.

As a feedback on the accuracy, at each iteration we compute the Maximum of the Residuals (MR)

$$\mathrm{MR} = \max_{i=1,\dots,\tilde{N}^{(k)}} r_i^{(k)}.$$

Furthermore, we calculate the RMSE on an independent set of evaluation points consisting of a grid of 40×40 points. In Figure 5 (left), we report the iterations versus the MR and RMSE.

N_c	method	ε_{opt}	RMSE	t_L	t_D	t_I
81	RBF-S	0.78	4.52E − 04	4.02E − 01	3.10E − 03	1.34E − 01
	TSVD	0.78	4.52E − 04	4.02E − 01	2.18E − 02	–
	HVSK	0.42	3.94E − 04	9.23E − 01	3.10E − 03	1.34E − 01
	RBF-QR	0.42	4.16E − 04	9.19E − 01	3.10E − 03	1.34E − 01
289	RBF-S	0.78	2.75E − 05	8.25E − 01	1.20E − 02	2.13E − 01
	TSVD	1.43	2.88E − 05	8.25E − 01	3.77E − 01	–
	HVSK	2.63	2.41E − 05	1.77E + 00	1.20E − 02	2.13E − 01
	RBF-QR	1.43	2.61E − 05	1.90E + 00	1.20E − 02	2.13E − 01
1089	RBF-S	2.63	9.19E − 06	6.02E + 00	7.19E − 02	1.14E + 00
	TSVD	1.43	6.06E − 06	6.02E + 00	1.50E + 01	–
	HVSK	2.63	8.96E − 06	6.46E + 00	7.19E − 02	1.14E + 00
	RBF-QR	1.43	1.02E − 06	6.62E + 00	7.19E − 02	1.14E + 00
4225	RBF-S	4.83	2.70E − 06	4.11E + 01	6.69E − 01	5.27E + 00
	TSVD	4.83	2.54E − 06	4.11E + 01	1.12E + 03	–
	HVSK	4.83	2.78E − 06	4.25E + 01	6.69E − 01	5.27E + 00
	RBF-QR	1.43	1.32E − 07	4.49E + 01	6.69E − 01	5.27E + 00

Table 2: RMSEs for the optimal shape parameter obtained for the test function f_1 and several sets of Halton nodes. The CPU time (in seconds) t_L is the time needed to construct the matrix L. The times t_D and t_I are those required to solve the final system by direct and iterative approaches, respectively. The quantity t_I corresponds to the time needed for both constructing the preconditioner and solving the system.

We also plot in Figure 5 (right) the residuals obtained by taking only the standard basis, showing that the procedure does not stop successfully. The ARS method combined with the HVSK scheme indeed avoid this situation (see Figure 5 left) and enanches the stability of the collocation matrices. A comparison of the condition numbers of the two methods is plotted in Figure 6.

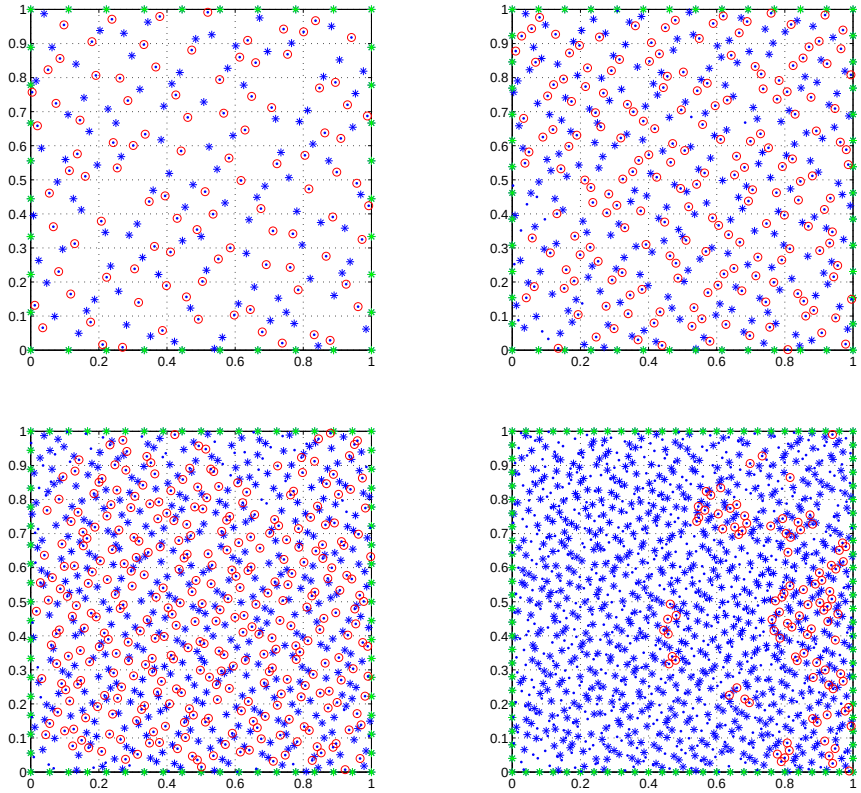

Fig. 3: An illustrative example. From top to bottom, left to right, we plot the first steps of the algorithm. The stars represent the data set at the k-th step, the dots the check nodes and the circles those check nodes that become new RBF centres at the $k+1$-th step.

Concerning the second test function f_3, in Figure 7 we plot the solutions and the final data sets obtained by considering both Halton (left) and greedy (right) points with shape parameter $\varepsilon = 3$. In both cases the ARS scheme stops after 10 iterations (see Figure 8). Nevertheless, with Halton data, it requires 2987 nodes and a maximum number of points per patch equal to 252. With greedy points it performs slightly better: indeed it stops with 2783 data and the maximum number of points per patch is equal to 338. In fact, greedy data are added only where the

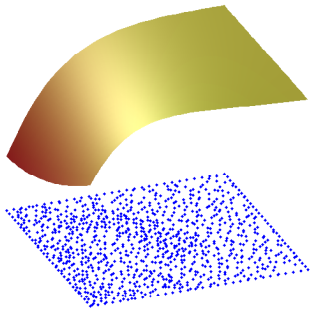

Fig. 4: The final data set and the so-reconstructed solution with Halton data for the test function f_2.

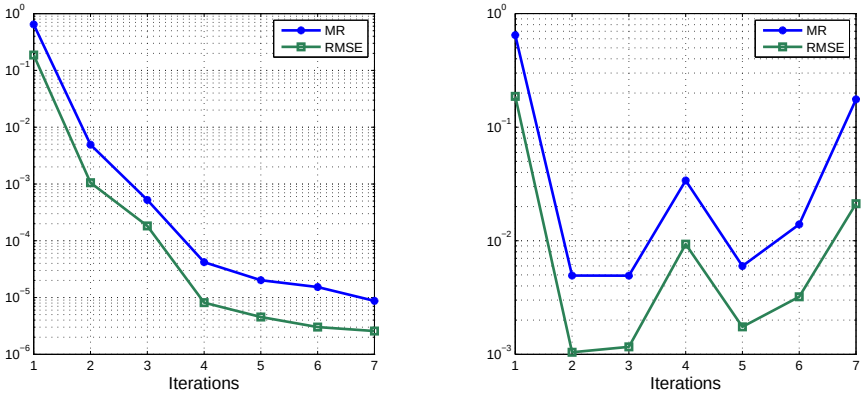

Fig. 5: The iterations versus the MR and RMSE with Halton data for the test function f_2. In the left frame we use the HVSK approach, while in the right one the standard bases.

solution grows more steeply and, differently from Halton points, they are coarse where the function is flatter.

As last example, we take an initial set of points (Halton and greedy data) on a circle inscribed in $[0,1]^2$, the test function f_4 and the shape parameter $\varepsilon = 3$. In this case, we use the criterion based on the directional derivatives (4.2). Again, we observe the pattern already provided by the greedy points with respect to Halton data. Indeed, from Figure 9, where we plot the two data sets and the reconstructed solutions, we note that Halton data oversample relatively flat regions and undersample the peak. The algorithm with Halton points stops after $k = 10$ iterations with 1740 points and a maximum number of points per patch equal to

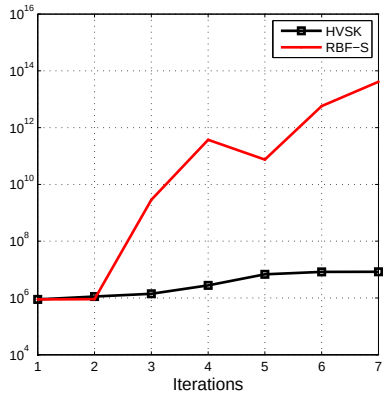

Fig. 6: The iterations versus the condition numbers of the final collocation matrix L.

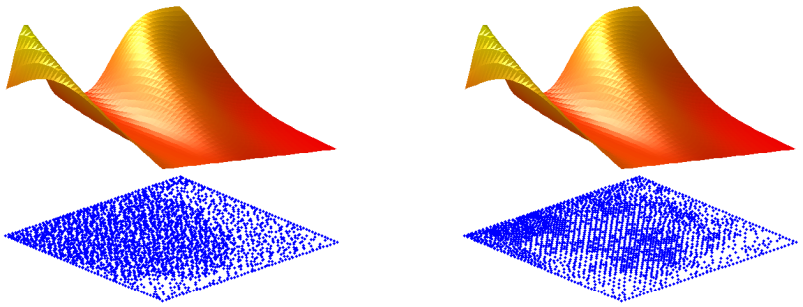

Fig. 7: The final data set and the so-reconstructed solutions for Halton (left) and greedy (right) data with the test function f_3.

224 (see Figure 10, left). The same approach with greedy points only requires 8 iterations and 1538 data, while the maximum number of points per patch is equal to 333 (see Figure 10, right).

Finally, to point out the efficiency, we report in Table 3 a comparison between the CPU times for the adaptive and non-adaptive methods, both computed via the HVSK scheme. The number of points for the non-adaptive HVSK scheme has been selected so that for the initial data sets all the residual are less than $\tau_1 = 10^{-5}$. As expected, we note that there is a remarkable difference for what concerns the number of points involved in the computation, truly larger for the non-adaptive case. The CPU times are instead comparable. For the non-adaptive method, the CPU time includes the time needed to test if for the taken data sets

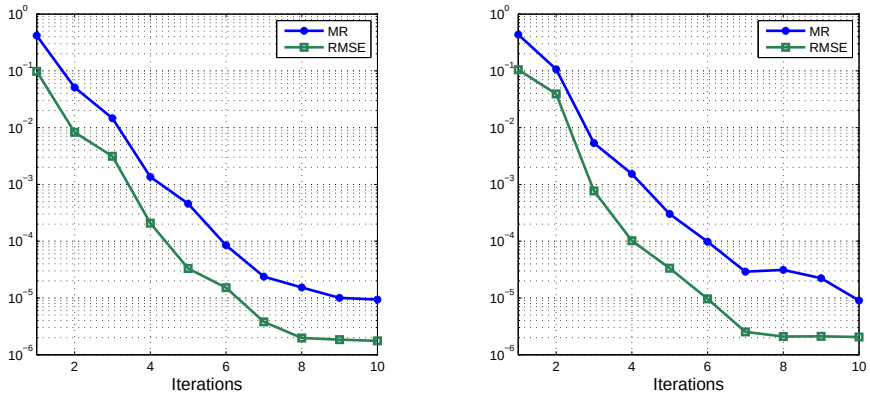

Fig. 8: The iterations versus the MR and RMSE for Halton (left) and greedy (right) data with the test function f_3.

all the residuals are less than τ_1. Of course, testing which check points become new nodes is the most time-consuming part of the algorithm, while the use of VSKs is very cheap. Indeed, in addition to the computation via standard bases, we only need evaluations of the scale function.

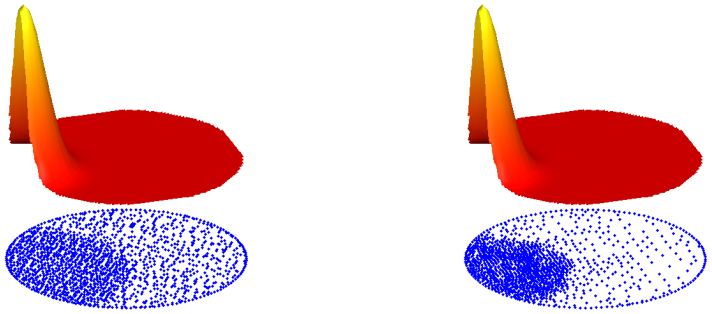

Fig. 9: The final data set and the so-reconstructed solutions for Halton (left) and greedy (right) data with the test function f_4.

6 Conclusions and work in progress

We presented a scheme to adaptively select RBF centres when a Poisson problem is solved by means of RBF-PU collocation. Moreover, thanks to the proposed new

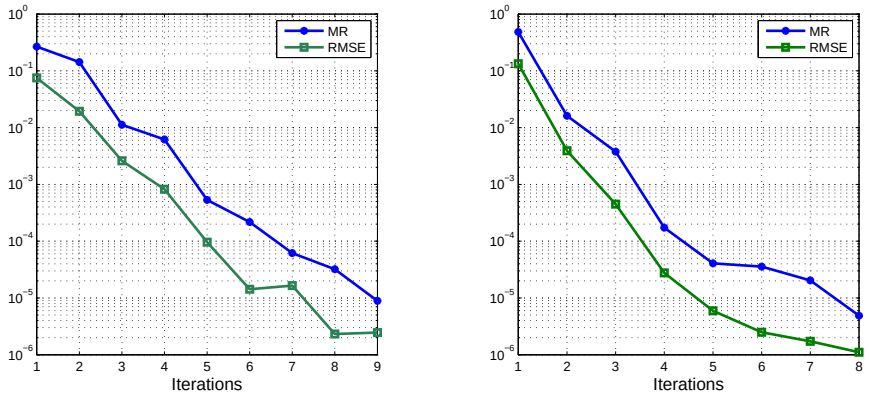

Fig. 10: The iterations versus the MR and RMSE for Halton (left) and greedy (right) data with the test function f_4.

HVSK technique, we enhance the stability of the algorithm. Future work consists in extending this investigation to parabolic PDEs, such as the heat equation, and in studying the potential use of a hybrid technique based on both VSKs and rescaled approximants, as well as rational RBFs [2,11,12,41]. Our aim is also the one of developing a parallel implementation of the described PU collocation scheme.

7 Acknowledgments

We sincerely thank the reviewers for their insightful comments. This research has been accomplished within Rete ITaliana di Approssimazione (RITA) and supported by GNCS-INδAM. The first author was partially supported by the research project *Approximation by radial basis functions and polynomials: applications to CT, MPI and PDEs on manifolds*, No. DOR1695473. The third author was partially supported by the research project *Radial basis functions approximations: stability issues and applications*, No. BIRD167404.

Test function	N	adaptivity	MR	t
f_1	2684	N	9.83E − 06	6.44E + 00
	1044	Y	9.82E − 06	1.00E + 01
f_2	7213	N	8.88E − 06	3.12E + 01
	2987	Y	9.37E − 06	3.26E + 01
f_3	4160	N	9.16E − 06	1.63E + 01
	1740	Y	9.87E − 06	1.51E + 01

Table 3: CPU times for adaptive (Y) and non-adaptive (N) methods with Halton data.

References

1. M. Bozzini, L. Lenarduzzi, M. Rossini, *Polyharmonic splines: An approximation method for noisy scattered data of extra-large size*, Appl. Math. Comput. **216** (2010), pp. 317–331.
2. M. Bozzini, L. Lenarduzzi, M. Rossini, R. Schaback, *Interpolation with variably scaled kernels*, IMA J. Numer. Anal. **35** (2015), pp. 199–219.
3. M. Caliari, S. De Marchi, M. Vianello, *Bivariate polynomial interpolation on the square at new nodal sets*, Appl. Math. Comput. **165** (2005), pp. 261–274.
4. R. Cancelliere, M. Gai, P. Gallinari, L. Rubini, *OCReP: An Optimally Conditioned Regularization for pseudoinversion based neural training*, Neural Netw. **71** (2015), pp. 76–87.
5. R. Cavoretto, A. De Rossi, F. Dell'Accio, F. Di Tommaso, *Fast computation of triangular Shepard interpolants*, to appear on J. Comput. Appl. Math. (2018).
6. 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), pp. 2568–2584.
7. R. Cavoretto, A. De Rossi, E. Perracchione, E. Venturino, *Graphical representation of separatrices of attraction basins in two and three-dimensional dynamical systems*, Int. J. Comput. Meth. **14** (2017), no. 1750008.
8. R. Cavoretto, G.E. Fasshauer, M. McCourt, *An introduction to the Hilbert-Schmidt SVD using iterated Brownian bridge kernels*, Numer. Algorithms **68** (2015), pp. 393–422.
9. O. Davydov, D.T. Oanh, *Adaptive meshless centres and RBF stencils for Poisson equation*, J. Comput. Phys. **304** (2011), pp. 230–287.
10. S. De Marchi, *On optimal center locations for radial basis interpolation: Computational aspects*, Rend. Sem. Mat. Torino, **61** (2003), pp. 343–358.
11. S. De Marchi, A. Idda, G. Santin, *A Rescaled Method for RBF Approximation*, G.E. Fasshauer et al. (eds), Approximation Theory XV: San Antonio 2016, vol. 201, 2017, pp. 39–59.
12. S. De Marchi, A. Martínez, E. Perracchione, *Fast and stable rational RBF-based partition of unity interpolation*, to appear on J. Comput. App. Math. 2018.
13. S. De Marchi, G. Santin, *Fast computation of orthonormal basis for RBF spaces through Krylov space methods*, BIT **55** (2015), pp. 949–966.
14. A. De Rossi, E. Perracchione, E. Venturino, *Fast strategy for PU interpolation: An application for the reconstruction of separatrix manifolds*, Dolom. Res. Notes Approx. **9** (2016), pp. 3–12.
15. T.A. Driscoll, A.R.H. Heryudono, *Adaptive residual subsampling methods for radial basis function interpolation and collocation problems*, Comput. Math. Appl. **53** (2007), pp. 927–939.
16. G.E. Fasshauer, *Dealing with Ill-Conditioned RBF Systems*, Dolomites Res. Notes Approx. **1** (2008).
17. G.E. Fasshauer, *Meshfree Approximations Methods with* Matlab, World Scientific, Singapore, 2007.
18. G.E. Fasshauer, J.G. Zhang, *On choosing "optimal" shape parameters for RBF approximation*, Numer. Algorithms **45** (2007), pp. 345–368.
19. P. Farrell, H. Wendland, *RBF multiscale collocation for second order elliptic boundary value problems*, J. Numer. Anal. **51** (2013), pp. 2403–2425.
20. B. Fornberg, E. Larsson, N. Flyer, *Stable computations with Gaussian radial basis functions*, SIAM J. Sci. Comput. **33** (2011), pp. 869–892.
21. E. Francomano, F.M. Hilker, M. Paliaga, E. Venturino, *An efficient method to reconstruct invariant manifolds of saddle points*, Dolom. Res. Notes Approx. **10** (2017), pp. 25–30.
22. M. Fuhry, L. Reichel, *A new Tikhonov regularization method*, Numer. Algorithms **59** (2012), pp. 433–445.
23. A. Heryudono, E. Larsson, A. Ramage, L. Von Sydow, *Preconditioning for radial basis function partition of unity methods*, J. Sci. Comput. **67** (2016), pp. 1089–1109.
24. Y.C. Hon, R. Schaback, *On unsymmetric collocation by radial basis functions*, Appl. Math. Comput. **119** (2001), pp. 177–186.
25. Y.C. Hon, R. Schaback, X. Zhou, *An adaptive greedy algorithm for solving large RBF collocation problems*, Numer. Algorithms **32** (2003), pp. 13–25.

26. E.J. Kansa, *Application of Hardy's multiquadric interpolation to hydrodynamics*, in: Proc. 1986 Simul. Conf. **4**, 1986, pp. 111–117.

27. M. Kowalewski, E. Larsson, A. Heryudono, *An adaptive interpolation scheme for molecular potential energy surfaces*, J. Chem. Phys. **145** (2016), pp. 84–104.
28. E. Larsson, B. Fornberg, *A numerical study of some radial basis function based solution methods for elliptic PDEs*, Comput. Math. Appl. **46** (2003), pp. 891–902.
29. 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), pp. A2096–A2119.

30. E. Larsson, V. Shcherbakov, A. Heryudono, *A least squares radial basis function partition of unity method for solving PDEs*, SIAM J. Sci. Comp. **39** (2017), pp. A2538–A2563.
31. L. Ling, E.J. Kansa, *A least-squares preconditioner for radial basis functions collocation methods*, Adv. Comput. Math. **23** (2005), pp. 31–54.
32. L. Ling, R. Opfer, R. Schaback, *Results on meshless collocation techniques*, Eng. Anal. Bound. Elem. **30** (2006), pp. 247–253.
33. J.M. Melenk, I. Babuška, *The partition of unity finite element method: Basic theory and applications*, Comput. Meth. Appl. Mech. Eng. **139** (1996), pp. 289–314.
34. D.T. Oanh, O. Davydov, H.X. Phu, *Adaptive RBF-FD method for elliptic problems with point singularities in 2D*, preprint, 2016.
35. M. Pazouki, R. Schaback, *Bases for kernel-based spaces*, J. Comput. Appl. Math. **236** (2011), pp. 575–588.

36. L. Romani, M. Rossini, D. Schenone, *Edge detection methods based on RBF interpolation*, to appear on J. Comput. Appl. Math. 2018.

37. M. Rossini, *Interpolating functions with gradient discontinuities via variably scaled kernels*, Dolom. Res. Notes Approx. **11** (2018), pp. 3–14.
38. A. Safdari-Vaighani, A. Heryudono, E. Larsson, *A radial basis function partition of unity collocation method for convection-diffusion equations arising in financial applications*, J. Sci. Comput. **64** (2015), pp. 341–367.
39. G. Santin, B. Haasdonk, *Convergence rate of the data-independent P-greedy algorithm in kernel-based approximation*, Dolomites Res. Notes Approx. 10 (2017), special issue pp. 68-78.
40. S.A. Sarra, *The Matlab radial basis function toolbox*, J. Open Research Software, **5** (2017), pp. 1–10.
41. S.A. Sarra, Y. Bay, *A rational radial basis function method for accurately resolving discontinuities and steep gradients*, preprint, (2017).
42. D. Shepard, *A two-dimensional interpolation function for irregularly spaced data*, in: Proceedings of 23-rd National Conference, Brandon/Systems Press, Princeton, 1968, pp. 517–524.
43. R. Schaback, *Convergence of unsymmetric kernel-based meshless collocation methods*, SIAM J. Numer. Anal. **45** (2007), pp. 333–351.
44. V. Shcherbakov, E. Larsson, *Radial basis function partition of unity methods for pricing vanilla basket options*, Comput. Math. Appl. **71** (2016), pp. 185–200.
45. A.N. Tikhonov, *Solution of incorrectly formulated problems and the regularization method*, Sov Math Dokl **4** (1963), pp. 1035–1038.
46. H. Wendland, *Fast evaluation of radial basis functions: Methods based on partition of unity*, in: C.K. Chui et al. (Eds.), Approximation Theory X: Wavelets, Splines, and Applications, Vanderbilt Univ. Press, Nashville, 2002, pp. 473–483.
47. H. Wendland, *Scattered data approximation*, Cambridge Monogr. Appl. Comput. Math., vol. 17, Cambridge Univ. Press, Cambridge, 2005.