Adaptive Refinement Techniques for RBF-PU Collocation

Adaptive Refinement Techniques for RBF-PU Collocation*

R. Cavoretto[0000−0001−6076−4115] and A. De Rossi[0000−0003−1285−3820]

Department of Mathematics "Giuseppe Peano", University of Torino
Via Carlo Alberto 10, 10123 Torino, Italy
{roberto.cavoretto,alessandra.derossi}@unito.it

Abstract. We propose new adaptive refinement techniques for solving Poisson problems via a collocation radial basis function partition of unity (RBF-PU) method. As the construction of an adaptive RBF-PU method is still an open problem, we present two algorithms based on different error indicators and refinement strategies that turn out to be particularly suited for a RBF-PU scheme. More precisely, the first algorithm is characterized by an error estimator based on the comparison of two collocation solutions evaluated on a coarser set and a finer one, while the second one depends on an error estimate that is obtained by a comparison between the global collocation solution and the associated local RBF interpolant. Numerical results support our study and show the effectiveness of our algorithms.

Keywords: Adaptive algorithms · Refinement strategies · RBF methods · Meshless methods · Elliptic PDEs.

1 Introduction

In this work we face the problem of designing new adaptive refinement algorithms for solving 2D Poisson problems via radial basis functions (RBFs) methods. Here, we focus on the construction of adaptive techniques based on the use of the partition of unity (RBF-PU) method, which is known in literature in the context of meshless methods for solution of interpolation and collocation problems (see [5,6]). More precisely, we present two algorithms within a RBF-PU framework, which in this paper we will call Algorithm 1 and Algorithm 2. The former is characterized by an error estimator based on the comparison of two collocation solutions evaluated on a coarser set and a finer one, the latter instead depends on an error estimate that is obtained by a comparison between the global collocation

* The authors acknowledge support from the Department of Mathematics "Giuseppe Peano" of the University of Torino via Project 2019 "Mathematics for applications". Moreover, this work was partially supported by INdAM – GNCS Project 2019 "Kernel-based approximation, multiresolution and subdivision methods and related applications". This research has been accomplished within RITA (Research ITalian network on Approximation).

solution and the associated local RBF interpolant. Numerical experiments show performance of both adaptive procedures.

The paper is organized as follows. Section 2 contains a description of the RBF-PU method for solving Poisson PDE problems. In Section 3 we present two adaptive algorithms that involve different refinement strategies. In Section 4 we show some numerical results obtained to illustrate the performance of the refinement techniques.

2 RBF-PU Method for 2D Poisson PDEs

2.1 The RBF-PU Method

Given an open and bounded domain $\Omega \subseteq \mathbb{R}^2$ and a function $u : \Omega \to \mathbb{R}$, we consider a set $X_N = \{\boldsymbol{x}_i\}_{i=1}^N \subseteq \Omega$ of collocation points. The domain Ω is then covered by d subdomains Ω_j such that $\bigcup_{j=1}^d \Omega_j \supseteq \Omega$ with some mild overlaps among them [6]. In addition, we assume that the subdomains are circles of fixed radius so that the covering of Ω is given by $\{\Omega_j\}_{j=1}^d$. Associated with each subdomain Ω_j we introduce a family of compactly supported, nonnegative and continuous weights w_j, with $\text{supp}(w_j) \subseteq \Omega_j$. The weight functions w_j form a partition of unity, i.e.

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

As functions w_j we choose to use Shepard's weights, which involve compactly supported functions such as Wendland C^2 functions defined on Ω_j (see [5]). The RBF-PU method can thus be expressed as follows

$$\tilde{u}(\boldsymbol{x}) = \sum_{j=1}^{d} w_j(\boldsymbol{x}) \tilde{u}_j(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega, \tag{1}$$

where $\tilde{u}_j$ defines the local RBF approximant

$$\tilde{u}_j(\boldsymbol{x}) = \sum_{i=1}^{N_j} c_i^j \phi_\varepsilon(||\boldsymbol{x} - \boldsymbol{x}_i^j||_2). \tag{2}$$

In (2) N_j indicates the number of points $\boldsymbol{x}_i^j \in X_{N_j} = X_N \cap \Omega_j$, c_i^j represents an unknown real coefficient, $||\cdot||_2$ is the Euclidean norm, and $\phi : \mathbb{R}_{\geq 0} \to \mathbb{R}$ denotes a RBF depending on a positive *shape parameter* ε such that

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

Some examples of popular RBFs [3,6] are given by

$$\begin{aligned} &\phi_\varepsilon(r) = \sqrt{1 + \varepsilon^2 r^2}, && \text{MultiQuadric (MQ)}, \\ &\phi_\varepsilon(r) = \exp(-\varepsilon r)(\varepsilon^3 r^3 + 6\varepsilon^2 r^2 + 15\varepsilon r + 15), && \text{Matérn } C^6 \text{ (M6)}. \end{aligned}$$

2.2 Solution of Poisson Problems

In order to show some applications of the RBF-PU collocation method to elliptic PDEs, we define a Poisson problem with Dirichlet boundary conditions

$$\begin{aligned} -\Delta u(\boldsymbol{x}) &= f(\boldsymbol{x}), \quad && \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) &= g(\boldsymbol{x}), \quad && \boldsymbol{x} \in \partial\Omega, \end{aligned} \tag{3}$$

where Δ is the Laplace operator [4]. Moreover, for convenience, we split the set X_N of discretization points into a set X_{N_I} of interior points and a set X_{N_B} of boundary points such that $X_N = X_{N_I} \cup X_{N_B}$, where N_I and N_B represent the number of interior and boundary collocation points, respectively.

Now, to find an approximate solution of the form (1), the problem (3) is discretized as follows

$$\begin{aligned} -\Delta\tilde{u}(\boldsymbol{x}_i) = -\sum_{j=1}^{d} \Delta\left(w_j(\boldsymbol{x}_i)\tilde{u}_j(\boldsymbol{x}_i)\right) &= f(\boldsymbol{x}_i), \quad && \boldsymbol{x}_i \in X_{N_I}, \\ \tilde{u}(\boldsymbol{x}_i) = \sum_{j=1}^{d} w_j(\boldsymbol{x}_i)\tilde{u}_j(\boldsymbol{x}_i) &= g(\boldsymbol{x}_i), \quad && \boldsymbol{x}_i \in X_{N_B}. \end{aligned} \tag{4}$$

The differential operator can be expanded to get

$$\begin{aligned} -\Delta\left(w_j(\boldsymbol{x}_i)\tilde{u}_j(\boldsymbol{x}_i)\right) = & -\Delta w_j(\boldsymbol{x}_i)\tilde{u}_j(\boldsymbol{x}_i) - 2\nabla w_j(\boldsymbol{x}_i)\cdot\nabla\tilde{u}_j(\boldsymbol{x}_i) \\ & -w_j(\boldsymbol{x}_i)\Delta\tilde{u}_j(\boldsymbol{x}_i), \qquad \boldsymbol{x}_i \in X_{N_I}. \end{aligned} \tag{5}$$

We then define the vector $\tilde{\boldsymbol{u}}_j = (\tilde{u}_j(\boldsymbol{x}_1^j), \dots, \tilde{u}_j(\boldsymbol{x}_{N_j}^j))^T$ of local nodal values and the local coefficient vector $\boldsymbol{c}_j = (c_1^j, \dots, c_{N_j}^j)^T$. Further, if we denote by $A_j \in \mathbb{R}^{N_j \times N_j}$ the matrix of entries $A_{ki} = \phi_\varepsilon(||\boldsymbol{x}_k^j - \boldsymbol{x}_i^j||_2)$, $k, i = 1, \dots, N_j$, from (2) we know that $\boldsymbol{c}_j = A_j^{-1}\tilde{\boldsymbol{u}}_j$, and so we obtain

$$\Delta\tilde{\boldsymbol{u}}_j = A_j^{\Delta} A_j^{-1}\tilde{\boldsymbol{u}}_j, \qquad \nabla\tilde{\boldsymbol{u}}_j = A_j^{\nabla} A_j^{-1}\tilde{\boldsymbol{u}}_j, \tag{6}$$

where A_j^{Δ} and A_j^{∇}, $j = 1, \dots, N_j$, are the matrices whose entries are

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

Associated with each subdomain Ω_j, we consider the following diagonal matrix

$$W_j^{\Delta} = \mathrm{diag}\left(\Delta w_j(\boldsymbol{x}_1^j), \dots, \Delta w_j(\boldsymbol{x}_{N_j}^j)\right),$$

with W_j^{∇} and W_j defined in similar way. To derive the discrete operator P_j, we differentiate (4) by using a product derivative rule and then apply the relations given in (6). By means of (5) and incorporating the Dirichlet boundary conditions, the discrete local Laplacian is given by

$$(P_j)_{ki} = \begin{cases} (\bar{P}_j)_{ki}, & \boldsymbol{x}_i^j \in X_{N_I}, \\ \delta_{ki}, & \boldsymbol{x}_i^j \in X_{N_B}, \end{cases}$$

where δ_{ki} is the Kronecker delta and

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

By assembling the local matrices P_j into the global matrix P, i.e.

$$(P_j)_{ki} = \sum_{j=1}^{d} (P_j)_{\eta_{kj}, \eta_{ij}}, \qquad k, i = 1, \dots, N,$$

we can obtain the global discrete operator and solve the sparse linear system

$$P \boldsymbol{y} = \boldsymbol{u}, \tag{7}$$

where $\boldsymbol{u} = (u_1, \dots, u_N)^T$ is defined by

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

and the numerical solution $\boldsymbol{y} = (\tilde{u}(\boldsymbol{x}_1), \dots, \tilde{u}(\boldsymbol{x}_N))^T$ is obtained by inverting the collocation matrix P in (7) (see [1]).

3 Adaptive Refinement Techniques

In this section we present two refinement algorithms that can be used to solve a Poisson problem via the RBF-PU method. They are based on different error indicators and refinement strategies, which are applied in the adaptive process.

3.1 Algorithm 1

At first, we define two sets of grid collocation points of size $N_1^{(0)}$ and $N_2^{(0)}$, such that $N_1^{(0)} < N_2^{(0)}$, where the symbol $^{(0)}$ identifies the iteration number. For the sake of clarity, we denote these sets as $X_{N_1^{(0)}}$ and $X_{N_2^{(0)}}$, respectively. Then, the iterative procedure starts and, for $k = 0, 1, \dots$, the collocation solutions $\tilde{u}_{N_1^{(k)}}$ and $\tilde{u}_{N_2^{(k)}}$ of the form (1) are computed on $N_1^{(k)}$ and $N_2^{(k)}$ collocation points. In order to know where we need to refine, we compare the two approximate solutions evaluated at the (coarser) set $X_{N_1^{(k)}}$, supposing that the solution computed on the set $X_{N_2^{(k)}}$ is more accurate than the previous one. So the error indicator is given by

$$E_i^{(k)} = |\tilde{u}_{N_2^{(k)}}(\boldsymbol{x}_i^{(k)}) - \tilde{u}_{N_1^{(k)}}(\boldsymbol{x}_i^{(k)})|, \qquad \boldsymbol{x}_i^{(k)} \in X_{N_1^{(k)}}.$$

After fixing a tolerance τ, we detect all points $\boldsymbol{x}_i^{(k)} \in X_{N_1^{(k)}}$ such that

$$E_i^{(k)} > \tau. \tag{8}$$

To refine the distribution of discretization points, we compute the *separation distance*

$$q_{X_{N_1^{(k)}}} = \frac{1}{2} \min_{i \neq j} ||\boldsymbol{x}_i^{(k)} - \boldsymbol{x}_j^{(k)}||_2, \qquad \boldsymbol{x}_i^{(k)}, \boldsymbol{x}_j^{(k)} \in X_{N_1^{(k)}}, \tag{9}$$

and, afterward, for $k = 0, 1, \ldots,$ we update the sets $X_{N_1^{(k+1)}}$ and $X_{N_2^{(k+1)}}$ of collocation points. In particular, if the condition (8) is satisfied, we add to $\boldsymbol{x}_i^{(k)}$ four and eight points to generate the sets $X_{N_1^{(k+1)}}$ and $X_{N_2^{(k+1)}}$, respectively (see Figure 1, left to right). In both cases the new sets are obtained by either adding or subtracting the value of (9) to the components of $\boldsymbol{x}_i^{(k)}$. Notice that these new sets are such that $X_{N_1^{(k)}} \subset X_{N_2^{(k)}}$, for $k = 1, 2, \ldots$. Finally, the adaptive algorithm stops when there are no points anymore that satisfy the condition (8), returning the set $X_{N_2^{(k^*)}}$, with k^* denoting the last iteration.

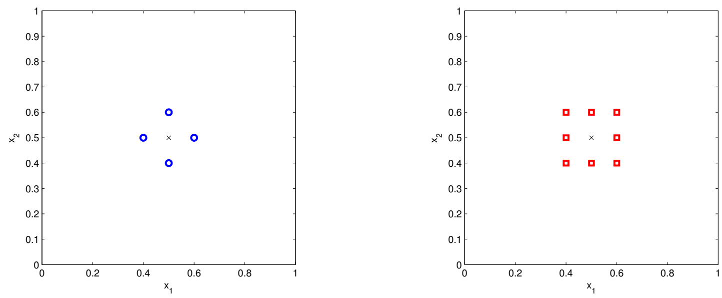

Fig. 1. Example of refinement around the point $\boldsymbol{x}_i^{(k)}$, marked by a black cross, to generate the new sets $X_{N_1^{(k+1)}}$ (left) and $X_{N_2^{(k+1)}}$ (right).

3.2 Algorithm 2

At the beginning, we define a set $X_{N^{(0)}}$ of grid collocation points, i.e. $X_{N^{(0)}} = X_N$, with the symbol $^{(0)}$ identifying – as in Algorithm 1 – the iteration of our adaptive procedure. So for $k = 1, 2, \ldots$ we compute the collocation solution $\tilde{u}_{N^{(k)}}$ of the form (1), and then for each subdomain Ω_j, $j = 1, \ldots, d$, we construct a local RBF interpolant $I\tilde{u}_{\Omega_j}^{(k)}$. It is obtained by using the (local) approximate values, which are obtained by solving the global collocation system (7). Acting in this way, we can give an error estimate evaluating $\tilde{u}_{N^{(k)}}$ and $I\tilde{u}_{\Omega_j}^{(k)}$ on a set $\Xi^{(k)} = \{\boldsymbol{\xi}_i\}_{i=1}^{n_k}$ of check points, being n_k the number of check points at the k-th iteration. Thus, the error indicator is given by

$$E_i^{(k)} = |\tilde{u}_{N^{(k)}}(\boldsymbol{\xi}_i^{(k)}) - I\tilde{u}_{\Omega_j}^{(k)}(\boldsymbol{\xi}_i^{(k)})|, \qquad \boldsymbol{\xi}_i^{(k)} \in \Xi^{(k)}. \tag{10}$$

Now, fixed two positive tolerances $\tau_{\min} < \tau_{\max}$ and the k-th iteration, if $E_i^{(k)}$ in (10) is larger than $\tau_{\max}$, we add the point $\boldsymbol{\xi}_i^{(k)}$ among the collocation points. Instead, if $E_i^{(k)}$ is smaller than $\tau_{\min}$, the check point is removed along with its nearest point. We can thus define the sets

$$Z_{T_{\max}^{(k)}} = \{\boldsymbol{\xi}_i^{(k)} \in \Xi^{(k)} \, : \, E_i^{(k)} > \tau_{\max}, \, i = 1, \ldots, T_{\max}^{(k)}\}$$

and

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

where $\bar{\boldsymbol{x}}_i^{(k)}$ is the point closest to $\boldsymbol{\xi}_i^{(k)}$. Iteratively, for $k = 2, 3, \ldots$, we can therefore obtain a new set of discretization points

$$X_{N^{(k)}} = X_{N_I^{(k)}} \cup X_{N_B^{(k)}}, \quad \text{where} \quad X_{N_I^{(k)}} = (X_{N_I^{(k-1)}} \cup Z_{T_{\max}^{(k-1)}}) \backslash Z_{T_{\min}^{(k-1)}}.$$

The process stops when the set $Z_{T_{\min}^{(k)}}$ is empty. Note that this adaptive refinement technique, which obeys the common paradigm of solve-estimate-refine/coarsen till a stopping criterion is satisfied, is based on the adaptive scheme given in [2].

4 Numerical Experiments

In this section we summarize the results obtained by the use of the refinement algorithms, described in Section 3 and implemented in MATLAB. All tests are carried out on a laptop with an Intel(R) Core(TM) i7-6500U CPU 2.50 GHz processor and 8GB RAM.

In this study we focus on two Poisson problems of the form (3) defined on the unit square, i.e. the domain $\Omega = [0, 1]^2$. The exact solutions of our elliptic problems are

$$\begin{aligned} \text{T1}: \;\; & u_1(x_1, x_2) = \sin(x_1 + 2x_2^2) - \sin(2x_1^2 + (x_2 - 0.5)^2), \\ \text{T2}: \;\; & u_2(x_1, x_2) = \exp(-8((x_1 - 0.5)^2 + (x_2 - 0.05)^2)). \end{aligned}$$

In Figure 2 we give a graphical representation of these analytic solutions.

In the tests we illustrate the performance of the adaptive RBF-PU scheme obtained by using the M6-RBF with $\varepsilon = 3$ and applying each of the two refinement algorithms. On the one hand, for Algorithm 1 the two starting sets defined in Subsection 3.1 consist of ${N_1}^{(0)} = 289$ and ${N_2}^{(0)} = 1089$ grid collocation points, while the tolerance is $\tau = 10^{-5}$. On the other hand, for Algorithm 2 described in Subsection 3.2 we start with $N^{(0)} = 121$ grid collocation points, whose stepsize is h_0 and tolerances are given by $(\tau_{\min}, \tau_{\max}) = (10^{-8}, 10^{-5})$; then, we refine such points by generating iteratively grid points, which have a stepsize $h_k = h_{k-1}/2$, $k = 1, 2, \ldots$, and are used as check points in (10). To measure the quality of our results, we compute the Root Mean Square Error (RMSE), that is,

$$\text{RMSE} = \left(\frac{1}{N_{eval}} \sum_{i=1}^{N_{eval}} |u(\boldsymbol{z}_i) - \tilde{u}(\boldsymbol{z}_i)|^2 \right)^{1/2}.$$

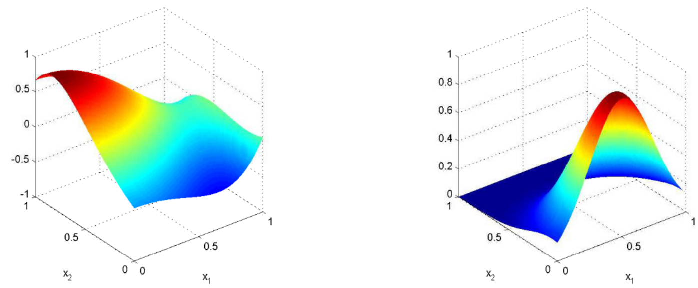

Fig. 2. Graphs of exact solutions of Poisson problems T1 (left) and T2 (right).

which is evaluated on a grid of $N_{eval} = 40 \times 40$ evaluation points. Then, to analyze the stability of the method, we evaluate the Condition Number (CN) of the sparse collocation matrix P in (7) by using the MATLAB command `condest`. As to efficiency we report the CPU times computed in seconds.

Therefore, in Tables 1–2 we present the results obtained, also reporting the final number N_{fin} of discretization points. In addition, in Figure 3 we show the "final grids" obtained by applying the adaptive refinement techniques.

Test Problem	N_{fin}	RMSE	CN	time
T1	1058	5.54e−6	1.35e+07	4.1
T2	1445	3.88e−6	1.03e+07	4.7

Table 1. Results obtained by using Algorithm 1 with M6, $\varepsilon = 3$ and $\tau = 10^{-5}$.

Test Problem	N_{fin}	RMSE	CN	time
T1	920	1.64e−5	4.76e+06	3.2
T2	1079	1.19e−5	9.16e+06	4.4

Table 2. Results obtained by using Algorithm 2 with M6, $\varepsilon = 3$ and $(\tau_{\min}, \tau_{\max}) = (10^{-8}, 10^{-5})$.

From these results, we observe as Algorithm 1 increases the number of points only in some more specific areas where the solution behavior varies, while Algorithm 2 distributes the points in a more uniform way. This results in greater accuracy of Algorithm 1, even if it is paid with a larger final number of points returned by the iterative procedure. These differences are also evident in terms of efficiency, because Algorithm 2 turns out to converge slightly more quickly than Algorithm 1. However, we note that both algorithms complete their work in few seconds. Similar conclusions can be done as regards the CN.

Finally, to assess the advantage of our adaptive schemes, as a comparison in case of the problem T1 we report the results obtained by applying the RBF-PU method on a uniform point set. In particular, in order to achieve a similar level of accuracy as found in Tables 1–2, we need 6400 collocation points to get a

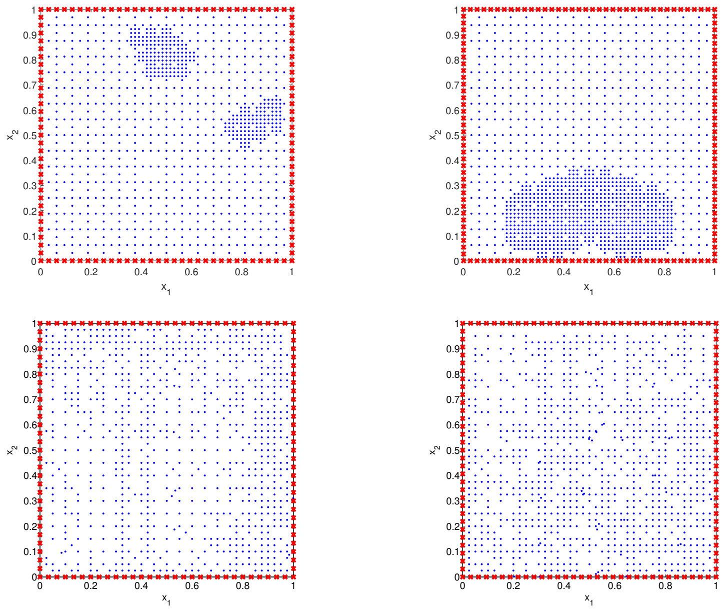

Fig. 3. Final distribution of discretization points obtained by applying the adaptive algorithms with M6, $\varepsilon = 3$, for problems T1 (left) and T2 (right). Top: Algorithm 1, bottom: Algorithm 2.

RMSE $= 5.57\text{e}{-6}$ in 6.5 seconds (cf. Table 1), and 2500 collocation points to reach a RMSE $= 1.74\text{e}{-5}$ in 3.3 seconds (cf. Table 2). As evident from these experiments, the adaptive algorithms are useful to reduce discretization error and CPU time, mainly when a higher level of accuracy is required.

References

1. Cavoretto, R., De Rossi, A.: Adaptive meshless refinement schemes for RBF-PUM collocation. Appl. Math. Lett. **90**, 131–138 (2019)
2. Driscoll, T.A., Heryudono, A.R.H.: Adaptive residual subsampling methods for radial basis function interpolation and collocation problems. Comput. Math. Appl. **53**, 927–939 (2007)
3. Fasshauer, G.E., McCourt, M.J.: Kernel-based Approximation Methods using MATLAB. Interdisciplinary Mathematical Sciences, vol. 19, World Scientific Publishing Co., Singapore (2015)
4. Heryudono, A., Larsson, E., Ramage, A., Sydow, L.V.: Preconditioning for radial basis function partition of unity methods. J. Sci. Comput. **67**, 1089–1109 (2016)
5. Safdari-Vaighani, A., Heryudono, A., Larsson, E.: A radial basis function partition of unity collocation method for convection-diffusion equations. J. Sci. Comput. **64**, 341–367 (2015)
6. Wendland, H.: Scattered Data Approximation. Cambridge Monogr. Appl. Comput. Math., vol. 17, Cambridge Univ. Press, Cambridge (2005)