POLITECNICO DI TORINO
Repository ISTITUZIONALE

Efficient Reduced Basis Algorithm (ERBA) for Kernel-Based Approximation

Original
Efficient Reduced Basis Algorithm (ERBA) for Kernel-Based Approximation / Marchetti, F.; Perracchione, E.. - In: JOURNAL OF SCIENTIFIC COMPUTING. - ISSN 0885-7474. - STAMPA. - 91:2(2022), pp. 1-17. [10.1007/s10915-022-01818-7]

Availability:
This version is available at: 11583/2966113 since: 2023-06-04T10:45:28Z

Publisher:
Springer

Published
DOI:10.1007/s10915-022-01818-7

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

28 April 2024

Journal of Scientific Computing (2022) 91:41
https://doi.org/10.1007/s10915-022-01818-7

Efficient Reduced Basis Algorithm (ERBA) for Kernel-Based Approximation

Francesco Marchetti[1] · **Emma Perracchione**[2]

Received: 7 July 2021 / Revised: 22 February 2022 / Accepted: 26 February 2022
© The Author(s) 2022

Abstract
The main purpose of this work is to provide an efficient scheme for constructing kernel-based reduced interpolation models. In the existing literature such problems are mainly addressed via the well-established *knot insertion* or *knot removal* schemes. Such iterative strategies are usually quite demanding from a computational point of view and our goal is to study an efficient implementation for data removal approaches, namely efficient reduced basis algorithm (ERBA). Focusing on kernel-based interpolation, the algorithm makes use of two iterative rules for removing data. The former, called ERBA-r, is based on classical residual evaluations. The latter, namely ERBA-p, is independent of the function values and relies on error bounds defined by the power function. In both cases, inspired by the so-called extended Rippa's algorithm, our ERBA takes advantage of a fast implementation.

Keywords Kernels · Reduced basis models · Efficient knot removal schemes · RBF approximation

Mathematics Subject Classification 65D05 · 41A05 · 65D10 · 65D15

1 Introduction

We introduce the kernel-based scattered data interpolation problem following [1, 2]. Let $\Omega \subseteq \mathbb{R}^d$ and $\mathcal{X} = \{\boldsymbol{x}_i,\ i = 1, \ldots, n\} \subset \Omega$ be a set of distinct nodes, $n \in \mathbb{N}$, where $\boldsymbol{x}_i = (x_{i,1}, \ldots, x_{i,d})^\intercal$, $i = 1, \ldots, n$. The scattered data interpolation problem consists in recovering an unknown function $f : \Omega \longrightarrow \mathbb{R}$ given its values at $\mathcal{X}$, i.e. $\boldsymbol{f} = \boldsymbol{f}_{|\mathcal{X}} = (f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n))^\intercal = (f_1, \ldots, f_n)^\intercal$. This can be achieved by imposing the interpolation conditions at $\mathcal{X}$. In particular, for kernel-based interpolation, the approximating function

✉ Francesco Marchetti
francesco.marchetti@math.unipd.it

Emma Perracchione
emma.perracchione@polito.it

1 Dipartimento di Matematica "Tullio Levi-Civita", Università di Padova, Padua, Italy

2 Dipartimento di Scienze Matematiche "Giuseppe Luigi Lagrange", Politecnico di Torino, Torino, Italy

Published online: 28 March 2022

assumes the form:

$$S_{f,\mathcal{X}}(\boldsymbol{x}) = \sum_{i=1}^{n} c_i \kappa_\varepsilon(\boldsymbol{x}, \boldsymbol{x}_i), \quad \boldsymbol{x} \in \Omega,$$

where $\boldsymbol{c} = (c_1, \ldots, c_n)^\mathsf{T} \in \mathbb{R}^n$ and $\kappa_\varepsilon : \Omega \times \Omega \longrightarrow \mathbb{R}$ is a strictly positive definite kernel depending on a *shape parameter* $\varepsilon > 0$. In addition, κ_ε is supposed to be *radial*, i.e. there exists a univariate function $\varphi_\varepsilon : \mathbb{R}_{\geq 0} \longrightarrow \mathbb{R}$ such that $\kappa_\varepsilon(\boldsymbol{x}, \boldsymbol{y}) = \varphi_\varepsilon(r)$, with $r := \|\boldsymbol{x} - \boldsymbol{y}\|_2$. As a consequence, such a setting is commonly referred to as *Radial Basis Function* (RBF) interpolation. With this clarification and with abuse of notation, we might write simply κ instead of κ_ε. The interpolation conditions are satisfied by finding the unique vector $\boldsymbol{c}$ so that

$$\mathsf{A}\boldsymbol{c} = \boldsymbol{f}, \tag{1}$$

where $\mathsf{A} = (\mathsf{A}_{i,j}) = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $i, j = 1, \ldots, n$, is the so-called interpolation (or collocation or simply kernel) matrix. The uniqueness of the solution of (1) is guaranteed as long as κ is strictly positive definite. For a more general formulation of the interpolant that involves *conditionally* positive definite kernels, and for a complete overview concerning kernel-based approximation, we refer the reader e.g. to [2].

The interpolant $S_{f,\mathcal{X}}$ belongs to the space

$$\mathcal{H}_\kappa = \operatorname{span}\{\kappa(\cdot, \boldsymbol{x}),\ \boldsymbol{x} \in \Omega\},$$

which, equipped with the bilinear form $(\cdot,\cdot)_{\mathcal{H}_\kappa}$, is a pre-Hilbert space with reproducing kernel κ. Moreover, the completion of $\mathcal{H}_\kappa$ with respect to the norm $\|\cdot\|_{\mathcal{H}_\kappa} = \sqrt{(\cdot,\cdot)_{\mathcal{H}_\kappa}}$ is the so-called *native space* $\mathcal{N}_\kappa$ associated to κ. Many upper bounds for the interpolation error, e.g. in terms of the fill-distance [1, Theorem 14.5, p. 121] and using sampling inequalities [3, 4], are available. Here we focus on the following pointwise error bound [1, Theorem 14.2, p. 117]:

$$|f(\boldsymbol{x}) - S_{f,\mathcal{X}}(\boldsymbol{x})| \leq P_{\kappa,\mathcal{X}}(\boldsymbol{x}) \|f\|_{\mathcal{N}_\kappa}, \quad f \in \mathcal{N}_\kappa, \quad \boldsymbol{x} \in \Omega, \tag{2}$$

where $P_{\kappa,\mathcal{X}}$ is the so-called *power function*. For the purposes of this paper, we directly define the power function as [5, p. 116, §14]:

$$P_{\kappa,\mathcal{X}}(\boldsymbol{x}) = \sqrt{\kappa(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{\kappa}^\mathsf{T}(\boldsymbol{x})\mathsf{A}^{-1}\boldsymbol{\kappa}(\boldsymbol{x})}. \tag{3}$$

where

$$\boldsymbol{\kappa}(\boldsymbol{x}) := (\kappa(\boldsymbol{x}, \boldsymbol{x}_1), \ldots, \kappa(\boldsymbol{x}, \boldsymbol{x}_n))^\mathsf{T}.$$

Note that (2) splits the error into two terms. The former, i.e. the power function, only depends on the nodes and on the kernel, while the latter takes into account the function f.

As a consequence, the power function gives information about how the interpolation error relates to the node distributions. Indeed, as for polynomial and spline interpolation, the approximation quality strongly depends on the distribution of the scattered data. In view of this, starting with an initial set of nodes, many *adaptive* strategies have been studied in order to construct *well-behaved* interpolation designs, i.e. interpolation sets which provide an accurate reconstruction and, preferably, affordable computational costs. In particular, the so-called *greedy* approaches match such purposes by iteratively adding new points to the interpolation set. The iterative rule is based on minimizing a pointwise upper bound for the interpolant. Precisely, those strategies rely on the residual of the interpolant (f-greedy) or on the value of the power function (p-greedy); refer to [6–9] for a general overview. Despite

the fact that we only focus on these two schemes, other strategies that combine the two above, known as $f \cdot p$ and f/p greedy, are available; we refer the reader to [10–12]. These methods fall under the context of *knot insertion* algorithms, which have been studied also in the context of adaptive least-squares approximation [1, §21].

Alternatively, as in our work, one could start by taking a *large* interpolation set, which provides an accurate approximation, and iteratively remove nodes until the resulting interpolation error does not exceed a fixed tolerance (see e.g. [13] for a general overview). These kinds of *knot removal* approaches aim to provide reduced models by neglecting as many points as possible from the initial set while preserving a target accuracy. However, they are usually computationally expensive, since many interpolants built upon large sets of nodes need to be constructed [14].

In this paper, in order to overcome the limitations related to the high computational complexity for reduced basis models, we take advantage of the Extension of Rippa's Algorithm (ERA) recently provided in [15]; see also [16]. More precisely, classical reduced basis iterative strategies are usually quite demanding (about $\mathcal{O}(n^4)$ operations) and our goal is to study an efficient implementation for data removal approaches, namely Efficient Reduced Basis Algorithm (ERBA), which only requires about $\mathcal{O}(n^3)$ operations. Since n might be large, we should call the algorithm *more* efficient RBA. However, with abuse of language, we simply refer to the algorithm as efficient RBA. For the ERBA scheme, besides the residual-based rule, that leads to the ERBA-r, and that can be derived from the ERA, we provide an efficient scheme for computing the power function as well. The resulting ERBA-p is a reduced model that takes advantage of being accurate, fast and easy to implement.

The paper is organised as follows. In Sect. 2, we fix some notations and we introduce the Reduced Basis Algorithm (RBA) based on knot removal strategies. Its efficient implementation, i.e. the ERBA scheme, and theoretical results are then provided in Sect. 3, where a detailed analysis of the computational complexity of the proposed schemes is also included. Numerical tests that confirm the theoretical findings are presented in Sect. 4. The conclusions are offered in Sect. 5.

2 The Reduced Basis Algorithm (RBA)

In what follows, we introduce our schemes for reduced basis models based on both the residual and on the power function minimization. The latter algorithm is quasi optimal in the sense that, referring to (2), we only take into account the power function, neglecting the term involving the native space norm of the sought function. This should not be seen as a drawback. Indeed, we are able to compute a quasi-optimal subset of points independently of the function values. As a consequence, it might be relevant if one has to deal with many measurements sampled at the same points. We further point out that the residual-based scheme has strong similarities with the knot removal algorithm shown in [1, §21]. We now present the RBA scheme and in the next section we focus on its fast implementation.

Let $\mathcal{X} = \{\boldsymbol{x}_i,\ i = 1, \dots, n\} \subset \Omega$ be the initial set of nodes, let $\mathcal{X}_{s-1} = \{\boldsymbol{x}_i,\ i = 1, \dots, n_{s-1}\} \subset \Omega$ be the reduced set at the $(s-1)$-th step of the algorithm ($\mathcal{X}_0 = \mathcal{X}$), and let $\tau \in \mathbb{R}_{>0}$ be a fixed tolerance. Choosing $\rho \in \mathbb{N}$, $\rho < n$, and letting $\ell = \lfloor n_{s-1}/\rho \rfloor$, the s-th step of the iterative scheme, $s \geq 1$, is as follows.

1. Partition $\mathcal{X}_{s-1}$ into ℓ test sets $\mathcal{X}_{s-1}^1, \dots, \mathcal{X}_{s-1}^\ell$, $|\mathcal{X}_{s-1}^j| = \rho^j$ with $\rho^j \in \{\rho, \dots, 2\rho - 1\}$, $\rho^j \leq \max(2\rho - 1, n_{s-1})$. Moreover, let $\overline{\mathcal{X}}_{s-1}^j := \mathcal{X}_{s-1} \setminus \mathcal{X}_{s-1}^j$ be the training sets, $j = 1, \dots, \ell$.

2. For each $\mathcal{X}_{s-1}^j$, $j = 1, \ldots, \ell$, compute:

   2a. for the residual-based scheme:

$$w^j = \frac{1}{\sqrt{\rho^j}} \| \boldsymbol{f}^j - \boldsymbol{S}^j \|_2, \tag{4}$$

   where $\boldsymbol{f}^j := \boldsymbol{f}_{|_{\mathcal{X}_{s-1}^j}}$ and $\boldsymbol{S}^j := S_{f,\overline{\mathcal{X}}_{s-1}^j}(\mathcal{X}_{s-1}^j)$ is the evaluation vector on $\mathcal{X}_{s-1}^j$ of the interpolant $S_{f,\overline{\mathcal{X}}_{s-1}^j}$. Here, we need to solve an $(n_{s-1} - \rho^j) \times (n_{s-1} - \rho^j)$ linear system for each $j = 1, \ldots, \ell$ (see (1));

   2b. for the power-based scheme:

$$w^j = \frac{1}{\sqrt{\rho^j}} \| \boldsymbol{P}^j \|_2,$$

   where $\boldsymbol{P}^j := P_{\kappa,\overline{\mathcal{X}}_{s-1}^j}(\mathcal{X}_{s-1}^j)$ is the evaluation vector on $\mathcal{X}_{s-1}^j$ of the power function $P_{\kappa,\overline{\mathcal{X}}_{s-1}^j}$. Here, we need to invert an $(n_{s-1} - \rho^j) \times (n_{s-1} - \rho^j)$ matrix for each $j = 1, \ldots, \ell$ (see (3)).

3. Choose

$$j^\star = \underset{j \in \{1,\ldots,\ell\}}{\operatorname{argmin}} \; w^j.$$

4. Let $r_{s-1} = w^{j^\star}$:

   4a. if $r_{s-1} \leq \tau$, define $\mathcal{X}_s = \mathcal{X}_{s-1} \setminus \mathcal{X}_{s-1}^{j^\star}$ and proceed iteratively with the s-th step;

   4b. if $r_{s-1} > \tau$, $\mathcal{X}_s = \mathcal{X}_{s-1}$ is the final interpolation set obtained by the procedure.

Remark 1 In the presented RBA algorithm, at each iteration, the data set is partitioned randomly. As also confirmed by the numerical experiments carried out in Sect. 4, since $\rho \ll n$, such randomness does not influence the algorithm in selecting meaningful designs of interpolation node sets according to the chosen approach. Nevertheless, if relevant, particular configurations of subsets can be taken into account (e.g. nearest neighbors).

The convergence of the residual-based rule has been studied in many context and for many basis functions, see e.g. [13] for a general overview. As far as the convergence of the proposed power function-based scheme is concerned, it is a consequence of [7, Lemma 2.3], precisely:

$$\| P_{\kappa,\mathcal{X}_{s-1}} \|_{L_\infty(\Omega)} \leq \| P_{\kappa,\mathcal{X}_s} \|_{L_\infty(\Omega)},$$

being $\mathcal{X}_s \subset \mathcal{X}_{s-1}$ nested sets.

From now on, in analyzing the computational complexity, we assume without loss of generality that ρ divides n_s at each step of the algorithm, and thus $\rho_j \equiv \rho$ holds true. Therefore, at each step, a classical implementation of this method requires to solving n_s different $(n_s - \rho) \times (n_s - \rho)$ linear systems, in the residual-based case, or the inversion of an $(n_s - \rho) \times (n_s - \rho)$ matrix in the power-based setting. Hence, in both situations, each step is computationally demanding. In [1, §21], a similar residual-based approach has been speeded up by using Rippa's algorithm in the case $\rho = 1$. In the next section, inspired by ERA [15], which speeds up the computational complexity for any $\rho \geq 1$, we provide a fast implementation of the proposed algorithm.

3 Efficient Reduced Basis Algorithm (ERBA)

To study the computational complexity of the proposed schemes, we have to focus on the calculation of the residuals (or of the power function) that is performed at each iteration. Then, fixed a certain step s, with $s = 1, 2, \ldots$, let us introduce the following notations: $\mathcal{X}_s := \mathcal{X}$, $n_s := n$, and let $\boldsymbol{p} := (p_1, \ldots, p_\rho)^\mathsf{T}$ be the vector of ρ indices related to the elements of the subset $\mathcal{X}_s^j := \mathcal{V}$, $|\mathcal{X}_s^j| = \rho$. We also adopt the following notations:

$$\begin{aligned}
\boldsymbol{f}_{\boldsymbol{p}} &:= (f_i)_{i\in\boldsymbol{p}}, \quad \boldsymbol{f}^{\boldsymbol{p}} := (f_i)_{i\notin\boldsymbol{p}},\\
\mathsf{A}^{\boldsymbol{p},\boldsymbol{p}} &:= (\mathsf{A}_{i,j})_{i,j\notin\boldsymbol{p}}, \quad \mathsf{A}_{\boldsymbol{p},\boldsymbol{p}} := (\mathsf{A}_{i,j})_{i,j\in\boldsymbol{p}}, \quad \mathsf{A}_{\boldsymbol{p},:} := (\mathsf{A}_{i,j})_{i\in\boldsymbol{p},j\in\{1,\ldots,n\}},\\
\boldsymbol{\kappa}(\boldsymbol{x}) &:= (\kappa(\boldsymbol{x},\boldsymbol{x}_1), \ldots, \kappa(\boldsymbol{x},\boldsymbol{x}_n))^\mathsf{T},\\
\boldsymbol{k}_{\boldsymbol{p}}(\boldsymbol{x}) &= (\boldsymbol{\kappa}(\boldsymbol{x})_i)_{i\in\boldsymbol{p}}, \quad \boldsymbol{k}^{\boldsymbol{p}}(\boldsymbol{x}) = (\boldsymbol{\kappa}(\boldsymbol{x})_i)_{i\notin\boldsymbol{p}},
\end{aligned}$$

In the following we explain our efficient strategy for the implementation of both the residual and power-based schemes, i.e. we speed up Step 2. of the RBA algorithm.

3.1 The ERBA-*r*

In step 2a. for the computation of $\boldsymbol{S}^j$ in Eq. (4), we need to compute the interpolant and hence solve a system that leads to a matrix inversion. Precisely, let us consider

$$S_{f,\overline{\mathcal{V}}}(\boldsymbol{x}) := S^{(\boldsymbol{p})}_{f,\mathcal{X}}(\boldsymbol{x}) = \sum_{i\notin\boldsymbol{p}} c_i^{(\boldsymbol{p})} \kappa_\varepsilon(\boldsymbol{x},\boldsymbol{x}_i), \quad \boldsymbol{x}\in\Omega,\ \boldsymbol{x}_i\in\mathcal{X},$$

where $\overline{\mathcal{V}} = \mathcal{X}\setminus\mathcal{V}$ and $\boldsymbol{c}^{(\boldsymbol{p})} := \left(c_i^{(\boldsymbol{p})}\right)_{i\notin\boldsymbol{p}}$ is determined by solving

$$\mathsf{A}^{\boldsymbol{p},\boldsymbol{p}}\boldsymbol{c}^{(\boldsymbol{p})} = \boldsymbol{f}^{\boldsymbol{p}}.$$

We are interested in computing the residual

$$\boldsymbol{e}_{\boldsymbol{p}} := \boldsymbol{f}_{\boldsymbol{p}} - ((\boldsymbol{c}^{(\boldsymbol{p})})^\mathsf{T}\boldsymbol{k}^{\boldsymbol{p}}(\mathcal{V}))^\mathsf{T}, \tag{5}$$

being $\boldsymbol{k}^{\boldsymbol{p}}(\mathcal{V}) = (\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j))_{i\in\boldsymbol{p},j\notin\boldsymbol{p}}$ an $(n-\rho)\times\rho$ matrix. Supposing $n/\rho = \ell\in\mathbb{N}$, a classic approach would lead to the resolution of ℓ different $(n-\rho)\times(n-\rho)$ linear systems, and thus, since usually $n \gg \rho$, to a computational cost of about $\mathcal{O}(n^4)$. In case $\rho = 1$ one could use the Rippa's algorithm reducing the complexity to $\mathcal{O}(n^3)$. In case of more folds, i.e. $\rho > 1$, we take advantage of the ERA scheme [15] for which (5) can be computed as

$$\boldsymbol{e}_{\boldsymbol{p}} = (\mathsf{A}^{-1}_{\boldsymbol{p},\boldsymbol{p}})^{-1}\boldsymbol{c}_{\boldsymbol{p}}. \tag{6}$$

Doing in this way, we need to invert an $n\times n$ matrix and then to solve ℓ different $\rho\times\rho$ linear systems. Therefore, the computational cost is about $\mathcal{O}(n^3)$, leading to a significant saving (cf. [15, §2.2]).

3.2 The ERBA-*p*

Here, we focus on the computation of the power function vector (see (3) and step 2b. of the algorithm presented in Sect. 2)

$$\boldsymbol{P}^{\mathcal{V}} = \sqrt{\mathrm{diag}(\boldsymbol{k}_{\boldsymbol{p}}(\mathcal{V}) - \boldsymbol{k}^{\boldsymbol{p}}(\mathcal{V})^\mathsf{T}(\mathsf{A}^{\boldsymbol{p},\boldsymbol{p}})^{-1}\boldsymbol{k}^{\boldsymbol{p}}(\mathcal{V}))}, \tag{7}$$

where diag$(\cdot)$ is the diagonal operator. The vector $\boldsymbol{P}^{\mathcal{V}}$ is usually computed by means of a for-loop over the elements of $\mathcal{V}$ (see [1, Program 17.1]). Hence, supposing again $n/\rho = \ell \in \mathbb{N}$, we need to invert ℓ matrices of dimension $(n-\rho) \times (n-\rho)$, leading to a computational cost of about $\mathcal{O}(\ell(n-\rho)^3) = \mathcal{O}(n(n-\rho)^3/\rho)$. Since $n \gg \rho$, the cost for each step of the algorithm is approximately $\mathcal{O}(n^4)$.

To provide a fast computation of the power function vector $\boldsymbol{P}^{\mathcal{V}}$, we first recall that, letting $\mathsf{R} \in \mathbb{R}^{n_1 \times n_2}$ and $\mathsf{S} \in \mathbb{R}^{n_2 \times n_1}$, we have that

$$\mathrm{diag}(\mathsf{RS}) = \mathrm{sum}(\mathsf{R} \odot \mathsf{S}^{\mathsf{T}}), \tag{8}$$

where $\odot$ denotes the Hadamard (or pointwise) product. Then, we obtain the following result.

Theorem 1 *The vector* (7) *can be computed as*

$$\boldsymbol{P}^{\mathcal{V}} = \sqrt{\mathrm{sum}(((A^{-1}_{\boldsymbol{p},\boldsymbol{p}})^{-1} A^{-1}_{\boldsymbol{p},:}) \odot \boldsymbol{k}(\mathcal{V})^{\mathsf{T}})},$$

where sum$(\cdot)$ *denotes the sum-by-rows operator.*

Proof Putting together (5) and (6), we have that

$$(\mathsf{A}^{-1}_{\boldsymbol{p},\boldsymbol{p}})^{-1} \boldsymbol{c}_{\boldsymbol{p}} = \boldsymbol{f}_{\boldsymbol{p}} - \boldsymbol{k}^{\boldsymbol{p}}(\mathcal{V})^{\mathsf{T}} (\mathsf{A}^{\boldsymbol{p},\boldsymbol{p}})^{-1} \boldsymbol{f}^{\boldsymbol{p}},$$
$$\boldsymbol{k}^{\boldsymbol{p}}(\mathcal{V})^{\mathsf{T}} (\mathsf{A}^{\boldsymbol{p},\boldsymbol{p}})^{-1} \boldsymbol{f}^{\boldsymbol{p}} = \boldsymbol{f}_{\boldsymbol{p}} - (\mathsf{A}^{-1}_{\boldsymbol{p},\boldsymbol{p}})^{-1} \boldsymbol{c}_{\boldsymbol{p}}.$$

Since $\boldsymbol{f}$ is arbitrary if the function f is not fixed, we can substitute it with $\boldsymbol{k}^{\boldsymbol{p}}(\mathcal{V})$ in the equation. Then,

$$\boldsymbol{k}^{\boldsymbol{p}}(\mathcal{V})^{\mathsf{T}} (\mathsf{A}^{\boldsymbol{p},\boldsymbol{p}})^{-1} \boldsymbol{k}^{\boldsymbol{p}}(\mathcal{V}) = \boldsymbol{k}_{\boldsymbol{p}}(\mathcal{V}) - (\mathsf{A}^{-1}_{\boldsymbol{p},\boldsymbol{p}})^{-1} \mathsf{A}^{-1}_{\boldsymbol{p},:} \boldsymbol{k}(\mathcal{V}).$$

Hence, recalling (7), we get

$$\boldsymbol{P}^{\mathcal{V}} = \sqrt{\mathrm{diag}((\mathsf{A}^{-1}_{\boldsymbol{p},\boldsymbol{p}})^{-1} \mathsf{A}^{-1}_{\boldsymbol{p},:} \boldsymbol{k}(\mathcal{V}))},$$

Finally, by applying (8) to $(\mathsf{A}^{-1}_{\boldsymbol{p},\boldsymbol{p}})^{-1} \mathsf{A}^{-1}_{\boldsymbol{p},:}$ and $\boldsymbol{k}(\mathcal{V})$, we conclude the proof. □

By adopting the scheme proposed in Theorem 1, the computational cost at each step is about $\mathcal{O}(n^3)$, indeed we need to invert a unique $n \times n$ matrix and to perform other *minor* calculations that are negligible as long as $n \gg \rho$. The proposed strategy is therefore faster than the classical framework, as confirmed by the experiments carried out in the next section.

4 Numerics

In what follows, we perform some numerical experiments to prove the efficiency of the proposed ERBA. The tests have been carried out in MATLAB on a Intel(R) Core(TM) i7-1165G7 CPU@2.80 GHz processor. The software is available for the scientific community at

`https://github.com/cesc14/ERBA.`

In Sect. 4.1 we compare our efficient implementation with the classical one. In doing so, we take both smooth and non-smooth target functions and different kernel bases. A focus on the CPU times is shown in Sect. 4.2, where we let both n and ρ vary. In Sect. 4.3, we

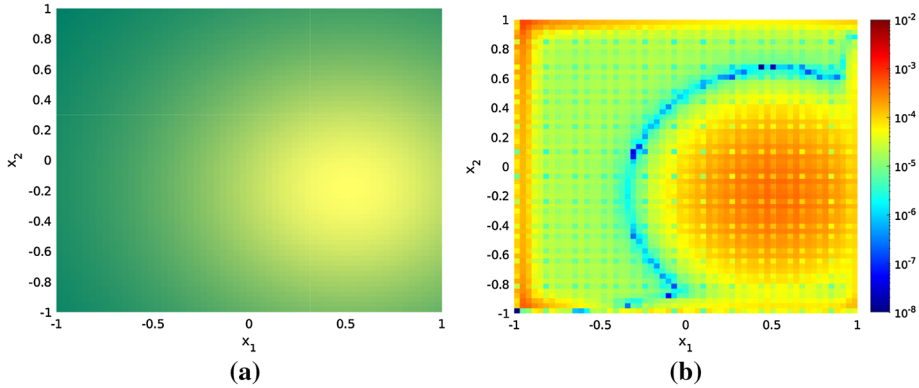


Fig. 1 (**a**) The function f. (**b**) The pointwise relative error between f and the interpolant constructed via the data set $\mathcal{X}$

analyze a stationary setting where the shape parameter varies according to the interpolation set. Finally, we present a test comparing the ERBA method and greedy schemes (which might be thought as forward RBA approaches).

4.1 ERBA versus RBA

Let $\Omega = [-1, 1]^2$. In this subsection, we take the strictly positive definite kernel

$$\varphi(r) = e^{-\varepsilon r}, \quad \text{Matérn } C^0,$$

and we set $\varepsilon = 1$. As interpolation data set $\mathcal{X} \subset \Omega$, we consider an $n \times n$ grid, with $n = 25$. Moreover, we take an $m \times m$ evaluation grid Ξ with $m = 60$.

4.1.1 Regular target function

Let $f : \Omega \longrightarrow \mathbb{R}$ be defined as

$$f(\boldsymbol{x}) = \frac{1}{1 + (x_1 - 0.5)^2 + (x_2 + 0.2)^2}, \quad \boldsymbol{x} = (x_1, x_2).$$

The associated RMSE computed via the whole data set $\mathcal{X}$ and evaluated on Ξ is $e_{\mathcal{X}} = 9.69\text{E}{-}05$. In Fig. 1, we plot the function f and the pointwise relative error between f and the interpolant $S_{f,\mathcal{X}}$ evaluated on Ξ.

For testing the proposed strategy, we set $\rho = 3$ and the tolerance $\tau_r = 2e_{\mathcal{X}}$ for ERBA-r, while for ERBA-p we take $\tau_p = 2\|\boldsymbol{P}^{\Xi}\|_2/m$, being $\boldsymbol{P}^{\Xi}$ the power function vector constructed via the nodes $\mathcal{X}$ and evaluated on Ξ. Denoting by $\mathcal{X}_s \subset \mathcal{X}$ the set of resulting reduced interpolation nodes, in Tables 1 and 2 we respectively compare the CPU times of ERBA-r and ERBA-p with their classical implementation. Moreover, the so-constructed reduced data sets, as well as the pointwise relative error between f and the interpolants constructed on $\mathcal{X}_s$, are depicted in Fig. 2.

The efficiency of our ERBA scheme is clearly deduced by Tables 1 and 2. Furthermore, as expected, we note that the ERBA-p tends to extract almost uniformly distributed nodes, while ERBA-r cluster the points where the target function has steep gradients. Such a behaviour

Table 1 Results for the ERBA-r scheme with f

$\lvert\mathcal{X}_s\rvert$	$e_{\mathcal{X}_s}$	ERBA CPU times	RBA CPU times
298	1.29E−04	3.14E+00	5.05E+02

CPU times are in seconds

Table 2 Results for the ERBA-p scheme with f

$\lvert\mathcal{X}_s\rvert$	$e_{\mathcal{X}_s}$	ERBA CPU times	RBA CPU times
103	2.41E−03	4.73E+00	5.28E+02

CPU times are in seconds

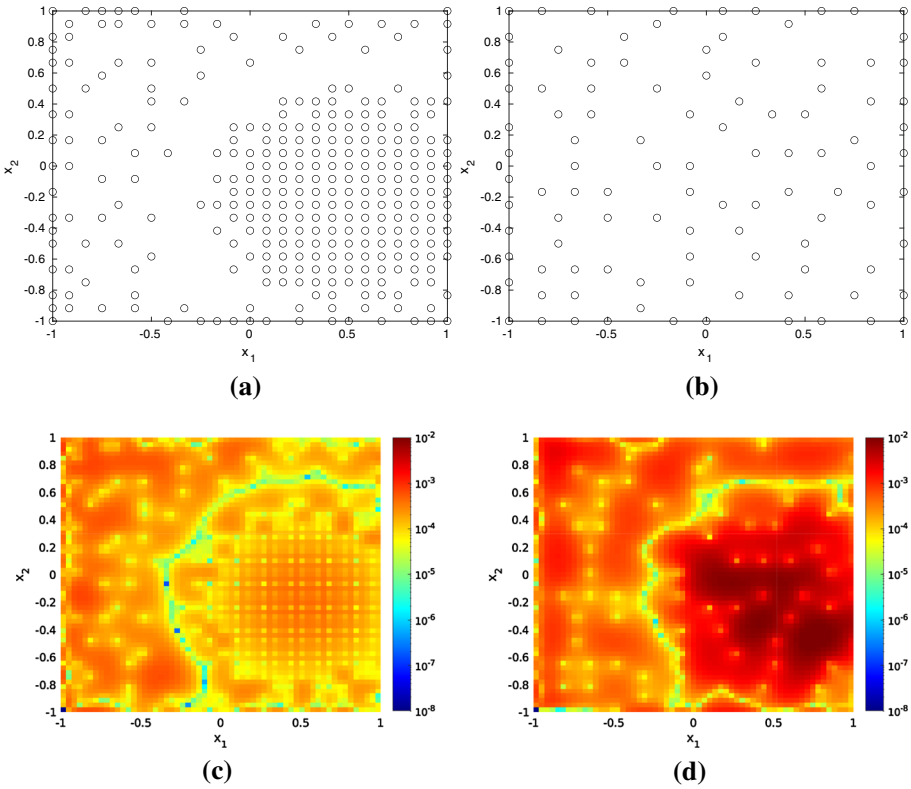


Fig. 2 Interpolation of f. (**a**) The reduced ERBA-r data set. (**b**) The reduced ERBA-p data set. The pointwise relative error between f and the interpolant constructed via the reduced ERBA-r (**c**) and ERBA-p (**d**) data set

will be amplified in the next experiment, which is carried out with a discontinuous target function.

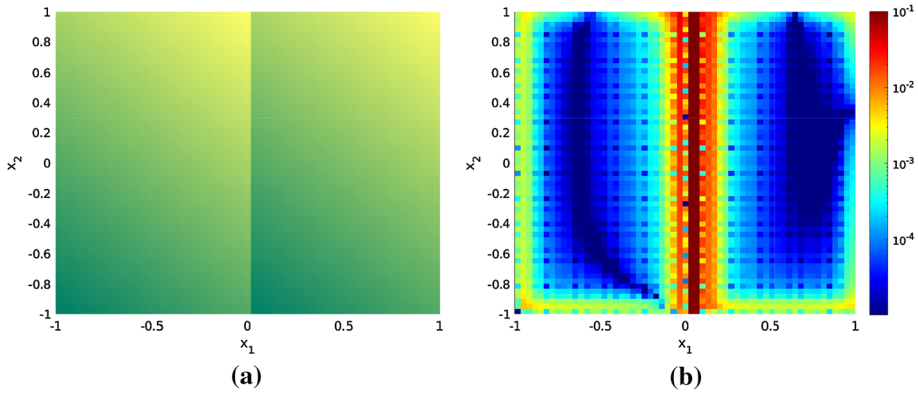


Fig. 3 (**a**) The function g. (**b**) The pointwise relative error between g and the interpolant constructed via the data set $\mathcal{X}$

Table 3 Results for the ERBA-r scheme with g

$\lvert\mathcal{X}_s\rvert$	$e_{\mathcal{X}_s}$	ERBA CPU times	RBA CPU times
82	1.62E−01	3.63E+00	5.82E+02

CPU times are in seconds

Table 4 Results for the ERBA-p scheme with g

$\lvert\mathcal{X}_s\rvert$	$e_{\mathcal{X}_s}$	ERBA CPU times	RBA CPU times
298	1.09E−01	4.36E+00	5.40E+02

CPU times are in seconds

4.1.2 Discontinuous target function

Let $g : \Omega \longrightarrow \mathbb{R}$ be defined as

$$g(\boldsymbol{x}) = \begin{cases} x_1 + x_2 - 3 & \text{if } x_1 > 0, \\ x_1 + x_2 - 2 & \text{if } x_1 \leq 0, \end{cases} \quad \boldsymbol{x} = (x_1, x_2).$$

The associated RMSE computed on Ξ is $e_{\mathcal{X}} = 1.14\text{E}{-}01$. In Fig. 3, we plot the function g and the pointwise relative error between g and the interpolant $S_{g,\mathcal{X}}$ evaluated on Ξ.

Here, we keep $\rho = 3$ and we set the tolerance $\tau_r = (3/2)e_{\mathcal{X}}$ for ERBA-r, while for ERBA-p we take $\tau_p = (3/2)\|\boldsymbol{P}^{\Xi}\|_2/m$, being $\boldsymbol{P}^{\Xi}$ the power function vector constructed via the nodes $\mathcal{X}$ and evaluated on Ξ. As in the previous test, we report the CPU times of ERBA-r and ERBA-p (in Tables 3, 4) and the reduced data sets together with the pointwise relative errors in Fig. 4. From these results with the discontinuous target function, we recover the same patterns of the previous experiments, i.e. the efficiency of the ERBA methods and the clustering effect of the ERBA-r.

4.2 Varying n and ρ

This subsection is dedicated to test how the efficacy of our schemes depends on n and ρ.

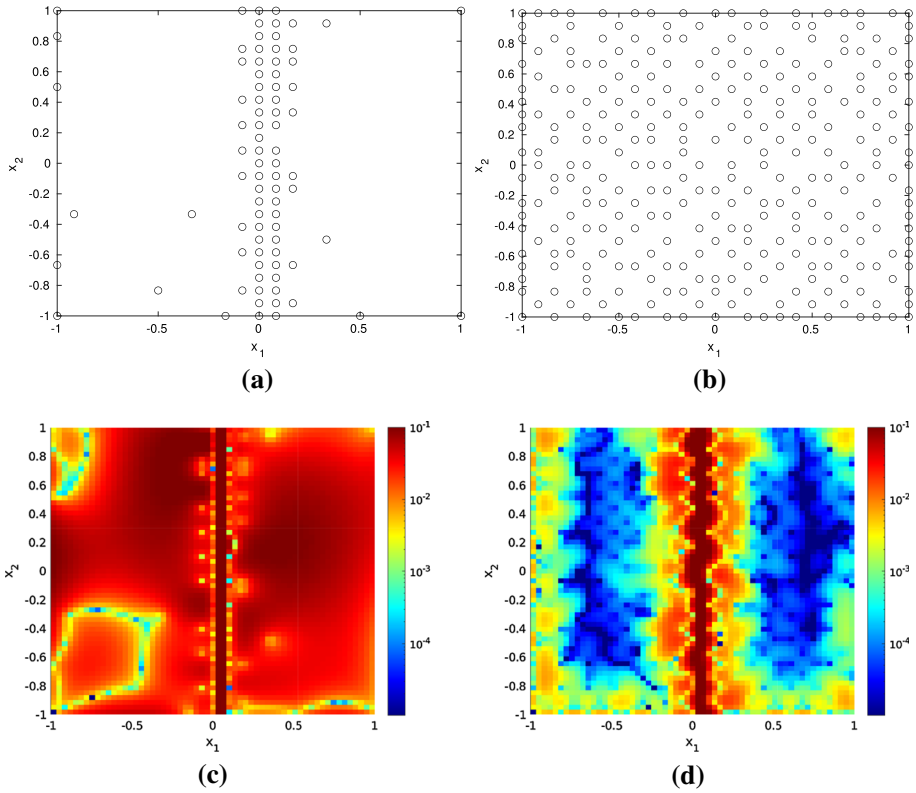


Fig. 4 Interpolation of g. (**a**) The reduced ERBA-r data set. (**b**) The reduced ERBA-p data set. The pointwise relative error between g and the interpolant constructed via the reduced ERBA-r (**c**) and ERBA-p (**d**) data set

4.2.1 A focus on n

As a further experiment, always with the aim of comparing the complexity of ERBA and RBA, we consider the setting of Sect. 4.1 (with target function f) and we compare the CPU times of the algorithms by varying the grid size of the initial data set $\mathcal{X}$. Precisely, we take $n = 15 + 3k$, $k = 0, \ldots, 7$. The results are reported in Fig. 5. We note that the saving in terms of computational complexity is relevant for each choice of n.

4.2.2 A focus on ρ

Here, always in the same setting of Sect. 4.1 (but with target function g), our purpose is to show how the CPU times and the number of reduced nodes provided by ERBA vary with the parameter ρ. Thus, we take $\rho = 2k$, $k = 1, \ldots, 7$, and we display the results in Fig. 6. We observe that the number of reduced nodes is almost constant, especially as long as ρ is small. As a consequence, the CPU times decrease with ρ, indeed less steps of the algorithm are performed.

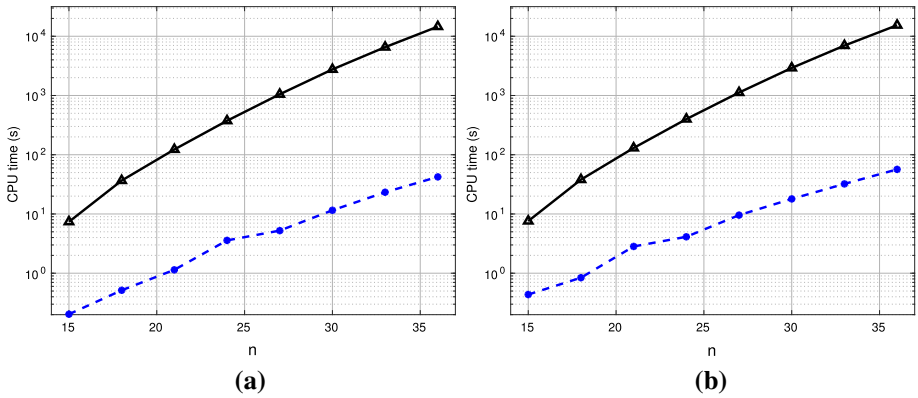


Fig. 5 The CPU times required by ERBA (dashed blue line) and RBA (solid black line) with f. (**a**) The residual based case. (**b**) The power based case

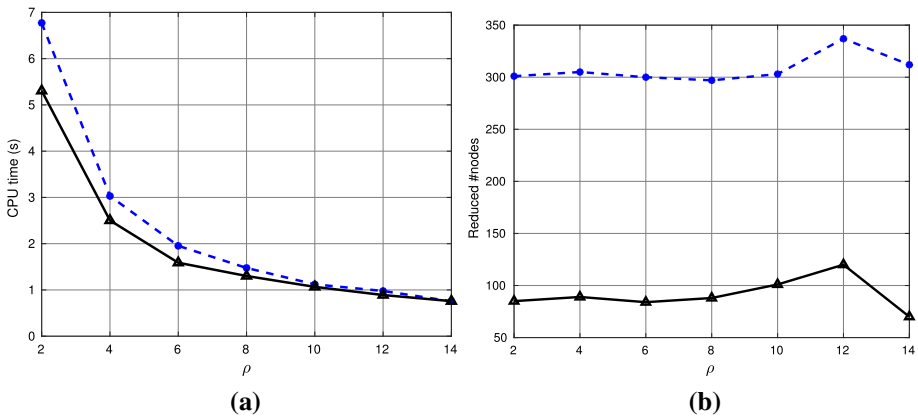


Fig. 6 ERBA-p (dashed blue line) and ERBA-r (solid black line) with g by varying ρ. (**a**) CPU time. (**b**) Final number of nodes

4.3 Stationary ERBA

In this subsection, we investigate the role of the shape parameter. We take the kernel

$$\varphi(r) = e^{-(\varepsilon r)^2}, \quad \text{Gaussian } C^\infty.$$

As interpolation set, we consider n^2 quasi-random *Halton points* [17] $\mathcal{H}$ in Ω, with $n = 37$, while we keep Ξ as evaluation grid. Here, we focus on a stationary interpolation approach: fixed an initial $\varepsilon_0 > 0$, at each iteration of our method, the shape parameter is set as $\varepsilon = \varepsilon_0 / h_{\mathcal{H}_{s-1},\Xi}$, where $h_{\mathcal{H}_{s-1},\Xi}$ is the fill-distance at the s-th step of the algorithm, i.e.,

$$h_{\mathcal{H}_{s-1},\Xi} = \max_{\boldsymbol{\xi}\in\Xi} \left(\min_{\boldsymbol{x}_i \in \mathcal{H}_{s-1}} \|\boldsymbol{\xi} - \boldsymbol{x}_i\|_2 \right).$$

In the following, we fix $\rho = 7$ and we consider the target function f introduced in Sect. 4.1. In Fig. 7, we plot the pointwise relative errors between such function and the interpolant constructed on $\mathcal{H}$. The associated RMSE computed on Ξ is $e_{\mathcal{H},f} = 1.23\text{E}{-}06$.

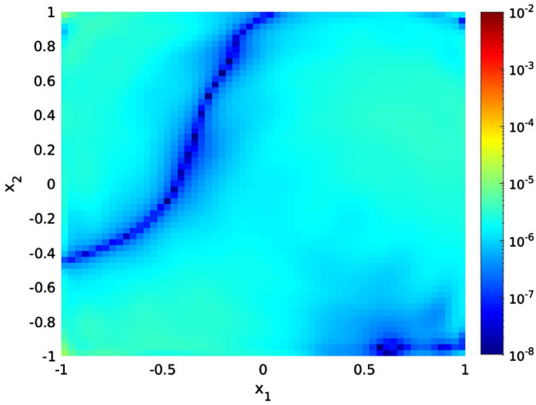
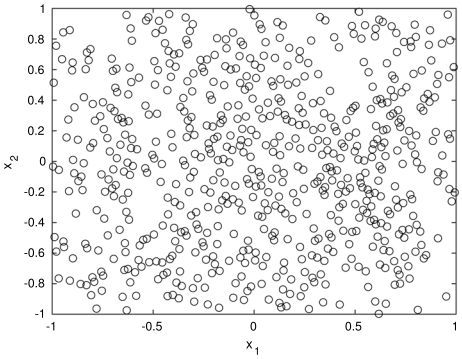
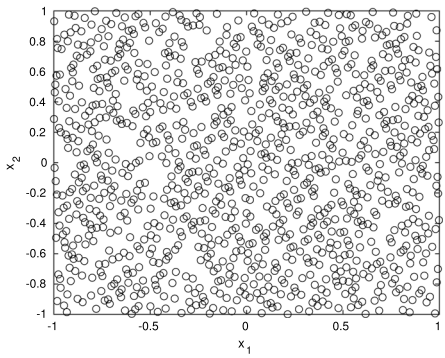


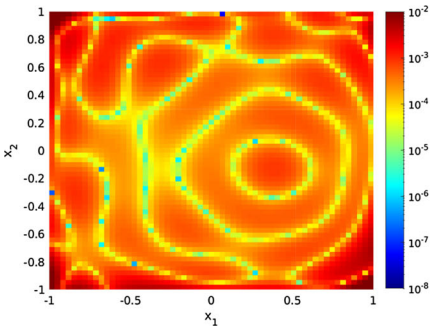
Fig. 7 The pointwise relative error between f and the interpolant constructed via the data set $\mathcal{H}$



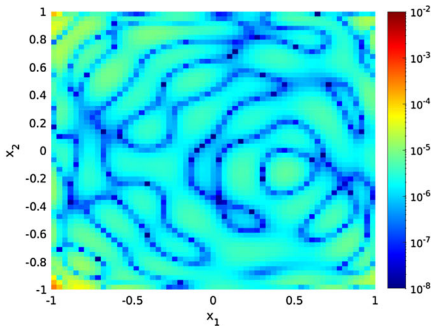
(a) $|\mathcal{H}_s| = 620,\ e_{\mathcal{H}_s} = 9.57\text{E} - 04.$



(b) $|\mathcal{H}_s| = 1096,\ e_{\mathcal{H}_s} = 3.29\text{E} - 06.$



(c)



(d)

Fig. 8 Interpolation of f. (**a**) The reduced ERBA-r data set. (**b**) The reduced ERBA-p data set. The pointwise relative error between f and the interpolant constructed via the reduced ERBA-r (**c**) and ERBA-p (**d**) data set

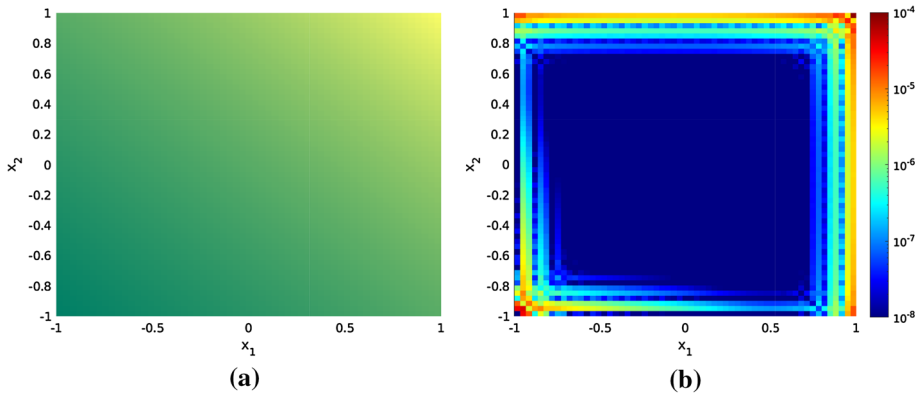


Fig. 9 (**a**) The function h. (**b**) The pointwise relative error between h and the interpolant constructed via the data set $\mathcal{Y}$

We now fix $\varepsilon_0 = 0.1$ and we set $\tau_r = 1\text{E}{+}04 r_0$ (for ERBA-r) and $\tau_p = (6/5) r_0$ (for ERBA-p), being r_0 the value produced at the first iteration of ERBA scheme as outlined in the algorithm described in Sect. 2. Then, the results of the stationary ERBA implementation are presented in Fig. 8. We point out that, for infinitely smooth kernels, such a stationary strategy becomes essential to partially avoid the typical ill-conditioning of the kernel matrices.

4.4 ERBA versus Greedy

As a final test, we carry out a comparison between our ERBA scheme and a greedy scheme that works in a specular manner with respect to RBA. For the sake of a clearer presentation, we detail it in the Appendix. We take the compactly-supported kernel

$$\varphi(r) = \max(0, 1 - \varepsilon r)^4 (4\varepsilon r + 1), \quad \text{Wendland } C^2,$$

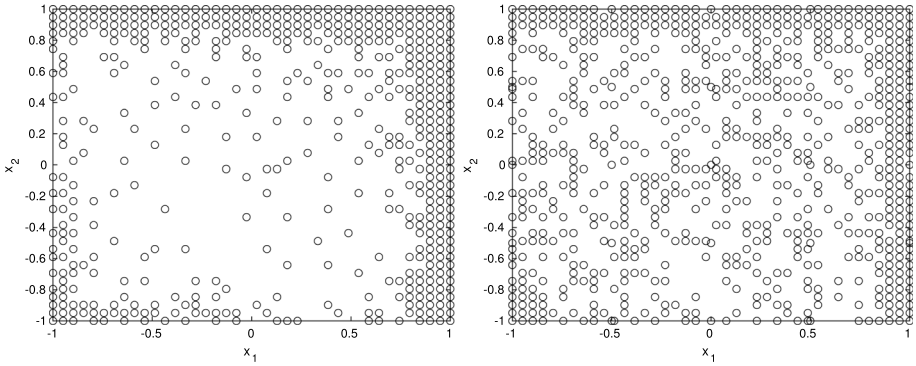
with $\varepsilon = 0.1$. As interpolation data set $\mathcal{Y} \subset \Omega$, we consider an $n \times n$ grid, with $n = 40$, while we keep Ξ as evaluation grid. Moreover, we consider the target function $h : \Omega \longrightarrow \mathbb{R}$ defined as

$$h(\boldsymbol{x}) = \tan\left(\frac{x_1 + x_2 + 3}{5}\right), \quad \boldsymbol{x} = (x_1, x_2).$$

The associated RMSE computed on Ξ is $e_{\mathcal{Y}} = 3.94\text{E} - 06$. In Fig. 9, we plot the function h and the pointwise relative error between h and the interpolant $S_{h,\mathcal{Y}}$ evaluated on Ξ.

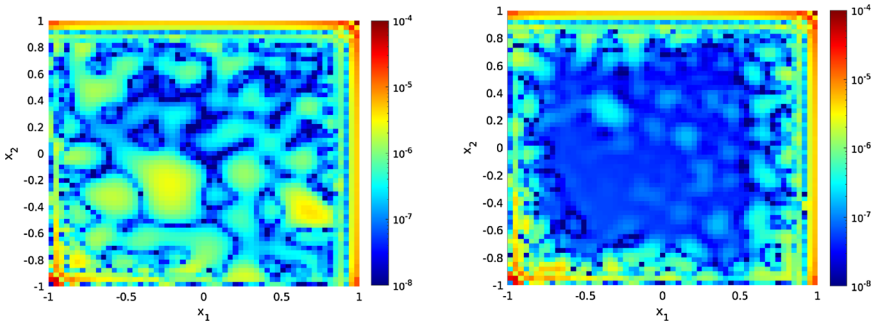
For both the ERBA and greedy algorithms we take $\rho = 5$. For ERBA-r and greedy-r, we set $\tau_r = 1\text{E}{-}06$, while we take $\tau_p = 5\|\boldsymbol{P}^{\Xi}\|_2/m$, being $\boldsymbol{P}^{\Xi}$ the power function vector constructed via the nodes $\mathcal{Y}$ and evaluated on Ξ, for ERBA-p and greedy-p. The initial set $\mathcal{X}_0$ for the greedy schemes is a 5×5 equispaced grid in Ω, which is nested in $\mathcal{X}$. In Figs. 10 and 11 we report the achieved results.

These experiments point out that in some cases, as when the number of nodes to remove is not *too large*, the ERBA is faster than the greedy approach. Indeed, the cost of greedy schemes increase when the number of the iterations grows.



(a) $|\mathcal{Y}_s| = 545,\ e_{\mathcal{Y}_s} = 3.97\mathrm{E}-06,$ CPU time$= 6.33\mathrm{E}+01.$

(b) $|\mathcal{Y}_s| = 760,\ e_{\mathcal{Y}_s} = 3.96\mathrm{E}-06,$ CPU time$= 1.00\mathrm{E}+03.$



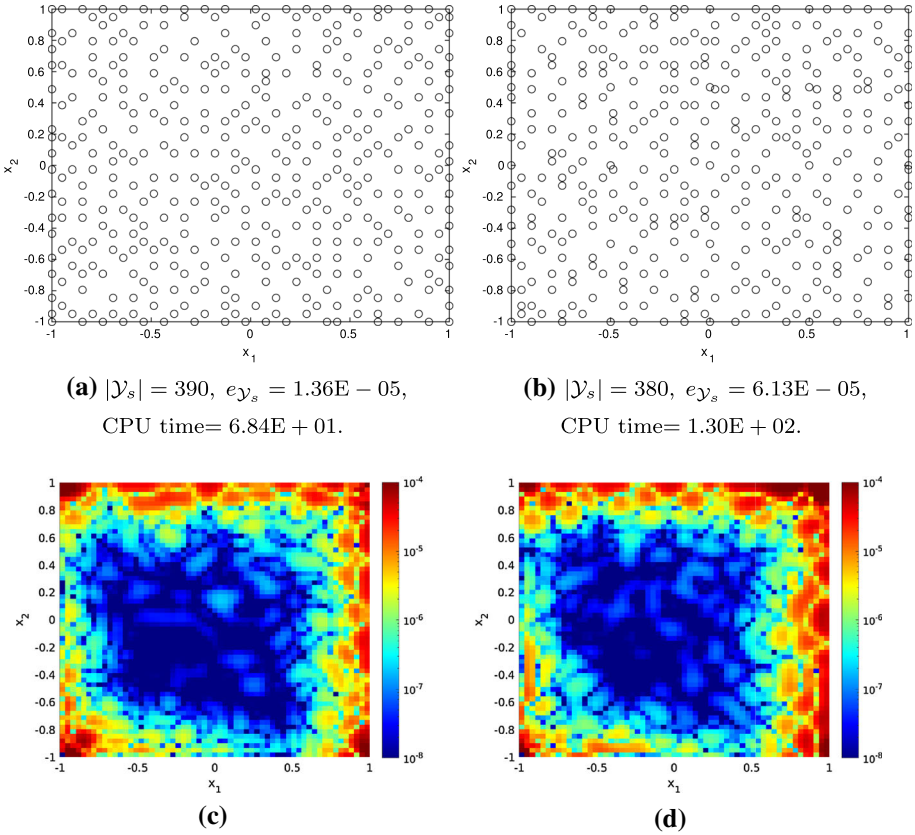
(c) **(d)**

Fig. 10 Interpolation of h. (**a**) The reduced ERBA-r data set. (**b**) The reduced greedy-r data set. The pointwise relative error between g and the interpolant constructed via the reduced ERBA-r (**c**) and greedy-r (**d**) data set

5 Discussion and Conclusions

We have investigated a fast computation of knot removal approaches. We have implemented two different strategies: the first one in based on classical residual-based schemes while the second one relies on power function error bounds. The latter is independent of the function values and tends to return *quasi-uniform* data (cf. [18] and [11, Theorem 15 & 19 & 20]), while the former, as expected, keeps more points where the corresponding function has steep gradients. In both cases we are able to significantly speed up the algorithm thanks to our implementation.

Work in progress consists in extending the proposed tool to other bases, e.g. splines [19] and to variably scaled kernels [20, 21]. Moreover, it can be helpful for validating the shape parameter in RBF interpolation; refer e.g. to [22].



(a) $|\mathcal{Y}_s| = 390,\ e_{\mathcal{Y}_s} = 1.36\text{E} - 05,$ CPU time$= 6.84\text{E} + 01.$

(b) $|\mathcal{Y}_s| = 380,\ e_{\mathcal{Y}_s} = 6.13\text{E} - 05,$ CPU time$= 1.30\text{E} + 02.$

(c) **(d)**

Fig. 11 Interpolation of h. (**a**) The reduced ERBA-p data set. (**b**) The reduced greedy-p data set. The pointwise relative error between h and the interpolant constructed via the reduced ERBA-p (**c**) and greedy-p (**d**) data set

Acknowledgements We sincerely thank the reviewers for helping us to significantly improve the manuscript. This research has been accomplished within Rete ITaliana di Approssimazione (RITA) and the UMI Group TAA Approximation Theory and Applications.

Funding This research has been partially funded by GNCS-INδAM and by the ASI-INAF grant "Artificial Intelligence for the analysis of solar FLARES data (AI-FLARES)".

Data availability Data sharing is not applicable to this article as no data sets were generated or analyzed.

Code Availability The software is available for the scientific community at https://github.com/cesc14/ERBA.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory

regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

Appendix: Greedy Algorithms

Let $\mathcal{X} = \{\boldsymbol{x}_i,\ i = 1, \ldots, n\} \subset \Omega$ be the initial set of nodes, let $\mathcal{X}_{s-1} = \{\boldsymbol{x}_i,\ i = 1, \ldots, n - n_{s-1}\} \subset \Omega$ be the extracted set at the $(s-1)$-th step of the algorithm. The set $\mathcal{X}_0 \subset \mathcal{X}$ is chosen taking $n_0 \ll n$. Let $\tau \in \mathbb{R}_{>0}$ be a fixed tolerance. Choosing $\rho \in \mathbb{N}$, $\rho < n$, and letting $\ell = \lfloor n - n_{s-1}/\rho \rfloor$, the s-th step of the iterative scheme, $s \geq 1$, is as follows.

1. Partition $\mathcal{X} \setminus \mathcal{X}_{s-1}$ into ℓ test sets $\mathcal{X}_{s-1}^1, \ldots, \mathcal{X}_{s-1}^\ell$, $|\mathcal{X}_{s-1}^j| = \rho^j$ with $\rho^j \in \{\rho, \ldots, 2\rho - 1\}$, $\rho^j \leq \max(2\rho - 1, n - n_{s-1})$.
2. For each $\mathcal{X}_{s-1}^j$, $j = 1, \ldots, \ell$, compute:

   2a. for the residual-based scheme (greedy-r):

   $$w^j = \frac{1}{\sqrt{\rho^j}} \| \boldsymbol{f}^j - \boldsymbol{S}^j \|_2,$$

   where $\boldsymbol{f}^j := \boldsymbol{f}_{|_{\mathcal{X}_{s-1}^j}}$ and $\boldsymbol{S}^j := S_{f,\mathcal{X}_{s-1}}(\mathcal{X}_{s-1}^j)$ is the evaluation vector on $\mathcal{X}_{s-1}^j$ of the interpolant $S_{f,\mathcal{X}_{s-1}}$;

   2b. for the power-based scheme (greedy-p):

   $$w^j = \frac{1}{\sqrt{\rho^j}} \| \boldsymbol{P}^j \|_2,$$

   where $\boldsymbol{P}^j := P_{\kappa,\mathcal{X}_{s-1}}(\mathcal{X}_{s-1}^j)$ is the evaluation vector on $\mathcal{X}_{s-1}^j$ of the power function $P_{\kappa,\mathcal{X}_{s-1}}$;
3. Choose

   $$j^\star = \underset{j \in \{1,\ldots,\ell\}}{\operatorname{argmax}}\ w^j.$$
4. Let $r_{s-1} = w^{j^\star}$:

   4a. if $r_{s-1} > \tau$, define $\mathcal{X}_s = \mathcal{X}_{s-1} \cup \mathcal{X}_{s-1}^{j^\star}$ and proceed iteratively with the s-th step;

   4b. if $r_{s-1} \leq \tau$, $\mathcal{X}_s = \mathcal{X}_{s-1}$ is the final interpolation set obtained by the procedure.

References

1. Fasshauer, G.E.: Meshfree Approximations Methods with Matlab. World Scientific, Singapore (2007)
2. Wendland, H.: Scattered Data Approximation, Cambridge Monogr. Appl. Comput. Math., vol. 17, Cambridge University Press, Cambridge (2005)
3. Fuselier, E., Wright, G.: Scattered data interpolation on embedded submanifolds with restricted positive definite kernels: Sobolev error estimates. SIAM J. Numer. Anal. **50**(3), 1753–1776 (2012)
4. Rieger, C.: Sampling Inequalities and Applications. Disseration, Göttingen (2008)
5. Fasshauer, G.E., McCourt, M.J.: Kernel-Based Approximation Methods Using Matlab. World Scientific, Singapore (2015)
6. 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)
7. Santin, G., Haasdonk, B.: Convergence rate of the data-independent P-greedy algorithm in kernel-based approximation. Dolomites Res. Notes Approx. **10**, 68–78 (2017)

8. Wirtz, D., Karajan, N., Haasdonk, B.: Surrogate modelling of multiscale models using kernel methods. Int. J. Numer. Methods Eng. **101**, 1–28 (2015)
9. Wirtz, D., Haasdonk, B.: A vectorial kernel orthogonal greedy algorithm. Dolomites Res. Notes Approx. **6**, 83–100 (2013)
10. Dutta, S., Farthing, M.W., Perracchione, E., Savant, G., Putti, M.: A greedy non-intrusive reduced order model for shallow water equations. J. Comput. Phys. **439**, 110378 (2021)
11. Wenzel, T., Santin, G., Haasdonk, B.: A novel class of stabilized greedy kernel approximation algorithms: convergence, stability and uniform point distribution. J. Approx. Theory **262**, 105508 (2021)
12. Wenzel, T., Santin, G., Haasdonk, B.: Analysis of target data-dependent greedy kernel algorithms: convergence rates for f-, $f \cdot P$- and f/P-greedy. arXiv: 2105.07411 (2021)
13. Lyche, T.: Knot removal for spline curves and surfaces. In: Cheney, E.W. et al (eds.), Approximation Theory, pp. 207–226 (1992)
14. Fasshauer,G.E.: Adaptive least squares fitting with radial basis functions on the sphere. In: Daehlen, M. et al. (eds.), Vanderbilt University Press (Nashville), pp. 141–150
15. Marchetti, F.: The extension of Rippa's algorithm beyond LOOCV. Appl. Math. Lett. **120**, 107262 (2021)
16. Rippa, S.: An algorithm for selecting a good value for the parameter in radial basis function interpolation. Adv. Comput. Math. **11**, 193–210 (1999)
17. Halton, J.H.: On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. Numer. Math. **2**, 84–90 (1960)
18. De Marchi, S., Schaback, R., Wendland, H.: Near-optimal data-independent point locations for radial basis function interpolation. Adv. Comput. Math. **23**, 317–330 (2005)
19. Campagna, R., Conti, C., Cuomo, S.: Smoothing exponential–polynomial splines for multiexponential decay data. Dolomites Res. Notes Approx. **12**, 86–100 (2018)
20. Campi, C., Marchetti, F., Perracchione, E.: Learning via variably scaled kernels. Adv. Comput. Math. **47**, 51 (2021)
21. Perracchione, E., Massone, A.M., Piana, M.: Feature augmentation for the inversion of the Fourier transform with limited data. Inverse Probl. **37**, 105001 (2021)
22. Cavoretto, R., De Rossi, A., Mukhametzhanov, M.S., Sergeyev, Y.D.: On the search of the shape parameter in radial basis functions using univariate global optimization methods. J. Glob. Optim. **79**, 305–327 (2021)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.