A rescaled method for RBF approximation

Stefano De Marchi, Andrea Idda and Gabriele Santin

Abstract In the recent paper [7], a new method to compute stable kernel-based interpolants has been presented. This *rescaled interpolation* method combines the standard kernel interpolation with a properly defined rescaling operation, which smooths the oscillations of the interpolant. Although promising, this procedure lacks a systematic theoretical investigation. Through our analysis, this novel method can be understood as standard kernel interpolation by means of a properly rescaled kernel. This point of view allow us to consider its error and stability properties.

1 Introduction

In the last decades radial basis functions have shown to be a flexible mathematical tool to solve scattered data interpolation problems, to model neural networks, a meshfree method for solving differential equations, parametrizing shape and surfaces and so on. Interested readers can refer, for example, to the comprehensive monographs [3, 12, 9], that give the necessary theoretical background and discuss many of the applications here enumerated.

In the paper [7] has been presented an interpolation method for the construction of stable kernel-based interpolants, called *rescaled interpolation*. It is a consistent local method that combines the standard kernel interpolation with a properly defined

Stefano De Marchi
Department of Mathematics - University of Padova (Italy), e-mail: demarchi@math.unipd.it

Andrea Idda
Banco Popolare - Verona (Italy), e-mail: a.idda1989@gmail.com

Gabriele Santin
IANS - University of Stuttgart (Germany), e-mail: gabriele.santin@mathematik.uni-stuttgart.de

rescaling operation, which essentially smooths the oscillations of the interpolant. Although promising, the method lacks a systematic theoretical understanding.

After recalling some necessary notation, we present the method and prove that it is an instance of the well-known *Shepard's method*, when certain weight functions are used. In particular, as for the Shepard's one, it reproduces constant functions.

Second, it is possible to define a modified set of cardinal functions strictly related to the ones of the not-rescaled kernel. Through these functions, we define a Lebesgue function for the rescaled interpolation process, and study its maximum - the Lebesgue constant - in different settings.

Also, a preliminary theoretical result on the estimation of the interpolation error is presented.

As an application, we couple our method with a partition of unity algorithm. This setting seems to be the most promising, and we illustrate its behavior with some experiments. The method has been also compared with the variably scaled kernel interpolation studied in [**?**].

We summarize briefly the paper structure. In the next section we introduce some basic definitions useful to understand the results presented in the paper. Then, in successive section 3 we present the *rescaled localized* RBF interpolant and discuss some of its properties. In particular, in the successive subsection 3.1 we present the kernel-based approach to the rescaled interpolant, formalizing some results already presented in [7]. In the case of kernels depending on the shape parameter, an interesting property of the method is that the shape parameter can be chosen neither too small or too big. In section 4 we show that this interpolant is indeed a *Shepard's approximant* and so it reproduces constants functions "at glance". Stability results of this construction are detailed in the successive subsection. The most promising application is the Partion of Unity Method (PUM). In Section 5 we apply the rescaled interpolant to the PUM, showing then in the numerical experiments its effectiveness. Finally in Section 5, we compare the behavior of the standand interpolant with respect to that of the rescaled and the *variably scaled* ones, which was firstly studied in [**?**]. We will show that the combination of PUM with the rescaled interpolant provides a stable method for interpolation.

2 Uselful notations

We start by recalling some notations useful for the sequel and necessary to understand the results that we are going to present.

Given a real Hilbert space $\mathscr{H}$ of functions from $\mathbb{R}^d$ to $\mathbb{R}$ with inner product $\langle , \rangle_{\mathscr{H}}$, a function $K : \Omega \times \Omega \longrightarrow \mathbb{R}$ with $\Omega \subset \mathbb{R}^d$, is called *reproducing kernel* for $\mathscr{H}$ if the two properties hold:

(i) $K(\cdot, x) \in \mathscr{H}$ for all $x \in \Omega$;
(ii) $\langle f, K(\cdot, x) \rangle_{\mathscr{H}} = f(x)$ for all $f \in \mathscr{H}$ and $x \in \Omega$.

The kernel is symmetric (by property (ii)) and positive definite by construction (cf. e.g. [9, §2.3] or [8, §13.1]).

Let $\Phi : \mathbb{R}^d \to \mathbb{R}$ be a continuous function. Φ is called *radial* if there exists a continuous function $\varphi : \mathbb{R}_0^+ \to \mathbb{R}$ such that $\Phi(x) = \varphi(\|x\|)$.

We are interested on *radial kernels*, i.e. kernels of the form $K(x,y) = \Phi(x,y) = \varphi(\|x-y\|)$, which means *invariant under translation and rotation*, with $\|\cdot\|$ the Euclidean distance. Radial kernels are often referred as *Radial Basis Functions* of shortly RBF. Moreover, since φ is a univariate function, we can call it the *basic radial function* and its values are function of a positive variable $r = \|x\|$.

Examples of $\mathscr{C}^\infty$ kernels are in Table 1 and kernels with finite smoothness in Table 2. An interesting review of nonstandand kernels is the paper [6] which presents many examples of kernels besides the classical ones presented in the Tables 1 and 2. We have written the kernels by introducing the parameter $\varepsilon > 0$ which is simply a *shape parameter*: for $\varepsilon \to \infty$, the functions become more and more spiky while as $\varepsilon \to 0$ they become flatter (see Figure 1). In the case of Wendland's kernels the parameter d denotes the maximal space dimension for which the functions are positive definite. With the symbol $\doteq$ we denote "equal up to some constants factors". The parameter k indicates that they are $\mathscr{C}^{2k}(\mathbb{R}^d)$. For example for $d = 1$, $l = 3$ we have $\varphi_{1,1}(r) \doteq (1-r)_+^4(4r+1)$, which is the well-known compactly supported Wendland $\mathscr{C}^2$ function. Introducing also for Wendland functions a shape parameter $\varepsilon > 0$, we simply stretch the support to $[0, 1/\varepsilon]$ having $\delta = 1/\varepsilon$ as its diameter.

kernel	parameter	support	name
$\varphi(\varepsilon r) = e^{-\varepsilon^2 r^2}$	$\varepsilon > 0$	$\mathbb{R}_0^+$	gaussian
$\varphi(\varepsilon r) = (1+\varepsilon^2 r^2)^{-1}$	$\varepsilon > 0$	$\mathbb{R}_0^+$	inverse quadrics
$\varphi(\varepsilon r) = (1+\varepsilon^2 r^2)^{-1/2}$	$\varepsilon > 0$	$\mathbb{R}_0^+$	inverse multiquadrics

Table 1 Examples of infinitely smooth kernels

kernel	parameter	support	name
$\varphi(\varepsilon r) = e^{-\varepsilon r}$	$\varepsilon > 0$	$\mathbb{R}_0^+$	M0
$\varphi(\varepsilon r) = (1+\varepsilon r)e^{-\varepsilon r}$	$\varepsilon > 0$	$\mathbb{R}_0^+$	M2
$\varphi_{d,0}(r) \doteq (1-r)_+^l$	$l = \lfloor d/2+k+1 \rfloor$	$[0,1]$	W0
$\varphi_{d,1}(r) \doteq (1-r)_+^{l+1}((l+1)r+1)$	$l = \lfloor d/2+k+1 \rfloor$	$[0,1]$	W2

Table 2 Examples of kernels with finite smoothness. M0 and M2 are Matérn kernels, W0 and W2 are Wendland kernels with smoothness 0 and 2 respectively.

Now, given a function $f : \Omega \to \mathbb{R}$, a set $X = \{x_1, \ldots, x_N\} \subset \Omega$ of N distinct points and the values $f_X = (f(x_1), \ldots, f(x_N))^T$ of the function f at the set X, we seek for interpolants of f_X of the form

$$P_f(x) = \sum_{i=1}^{N} c_i K(x, x_i),\ x \in \Omega. \tag{1}$$

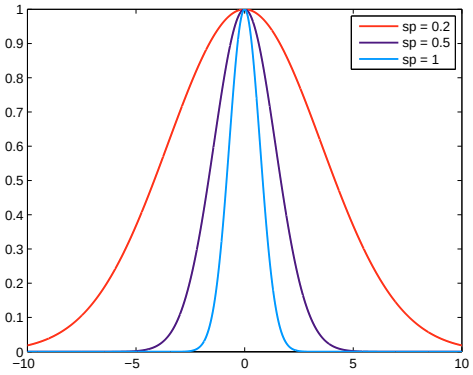

Fig. 1 The effect of changing the shape parameter on the gaussian kernel plotted in $[-10,10]$

for kernels K which are strictly positive definite and radial. This means that P_f is a function in $H_K(X) = \text{span}\{K(\cdot,x_i),\ i = 1,\dots,N\}$, formed by translates of K at the point set X. The coefficients in (1) are determined by imposing the interpolation conditions, which is equivalent to find the unique solution of the linear system $Ac = f_X$ with $A_{i,j} = K(x_i,x_j)$. Since the kernel K is assumed to be strictly positive definite then the solution exists and is unique.

The Hilbert space in which the kernel K is reproducing, is known as the associate *native space*. We will denote it by $\mathcal{N}_K$, instead of $\mathcal{H}$, to underline the dependence on the kernel. It is equipped by the scalar product $(\cdot,\cdot)_{\mathcal{N}_K}$, from which we get the *native space norm* $\|\ \|_{\mathcal{N}_K}$ (cf. e.g. [8]).

The interpolation process by kernels of functions $f \in \mathcal{N}_K$, gives pointwise errors of the form (see e.g. [9, p. 174] or [6, p. 19])

$$|f(x) - P_f(x)| \leq C h_{X,\Omega}^{\beta} \|f\|_{\mathcal{N}_K} \tag{2}$$

for some appropriate exponent β depending on the smoothness of the kernel K, and $h_{X,\Omega}$ that denotes the *mesh-size* (or fill-distance) $h_{X,\Omega} = \max_{x\in\Omega} \min_{x_i\in X} \|x - x_i\|_2$. For functions belonging to bigger spaces than the native space, suitable estimates are based on *sampling inequalities* as discussed e.g. in [9, §9.4].

3 The rescaled interpolant

In [7] the authors have proposed a new compactly supported RBF interpolant with the aim of a more accurate interpolantion even by using a small diameter for the support. More precisely, on the set of points X, we consider the constant function $g(x) = 1\ \forall x \in \Omega$, and we denote by $P_g(x)$ the corresponding kernel-based inter-

polant, that is

$$P_g(x) = \sum_{i=1}^{N} d_i K(x, x_i),$$

whose coefficients $d = (d_1, \ldots, d_N)^T$ can be determined as the solution of the linear system $Ad = 1$, with the vector 1 of ones.

Then, the *rescaled* interpolant is

$$\hat{P}_f(x) = \frac{P_f(x)}{P_g(x)} = \frac{\sum_{i=1}^{N} c_i K(x, x_i)}{\sum_{i=1}^{N} d_i K(x, x_i)}. \tag{3}$$

As a simple illustrative example we want to interpolate $f(x) = x$ on the interval $[0,1]$ by using the W2 function at the points set $X = \{1/6, 1/2, 5/6\}$ with shape parameter $\varepsilon = 5$. The function, the interpolant and the errors $|f(x) - P_f(x)|$, $|f(x) - \hat{P}_f(x)|$ are displayed in Figure 2. The shape parameter has been chosen as the reciprocal

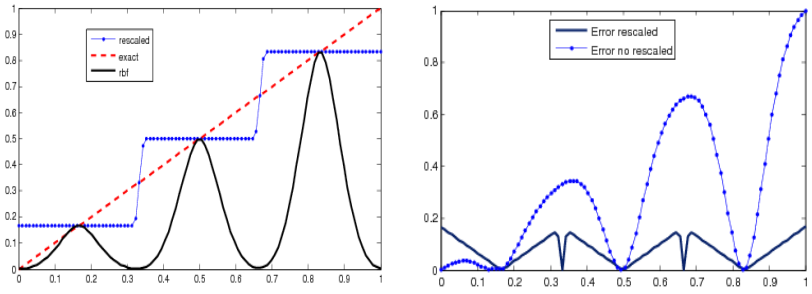

Fig. 2 Left: the function $f(x) = x$ interpolated with the compactly supported W2 function and the rescaled interpolant at the data set $X = \{1/3, 2/3, 5/6\}$ and $\varepsilon = 5$. Right: the corresponding absolute errors.

of the support radius of the corresponding basis function. Indeed, in this example the interpolant is the combination of *three* W2 radial functions all having radius of the support $r_j = 1/5,\ j = 1,2,3$. Refining the point set, that is considering the points set $X = \{0, 1/6, 1/3, 1/2, 2/3, 5/6, 1\}$, we get the results shown in Figure 3 showing the most interesting behavior of the rescaled interpolant, the property to reduce oscillations and so the interpolation error.

We show in Figure 4 the RMSE (Root Mean Square Error) of the stationary interpolation with the rescaled interpolant (1) w.r.t. the classical one (1) on a grid of 25 data points of the square $[0,1]^2$ at different values of the shape parameter, by using the W2 radial function for the 2d *Franke function*

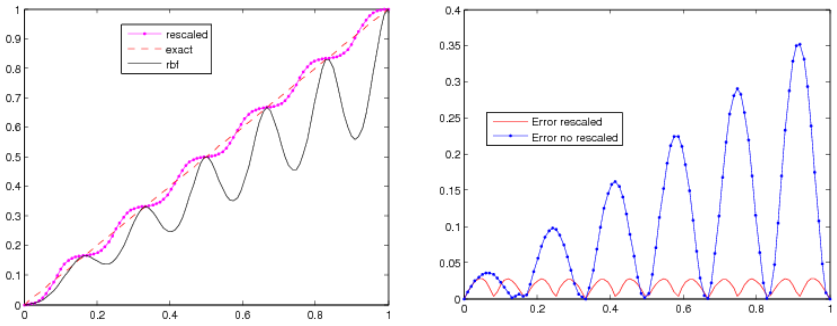

Fig. 3 As in Figure 2 for the set $X = \{0, 1/6, 1/3, 1/2, 2/3, 5/6, 1\}$ and $\varepsilon = 5$.

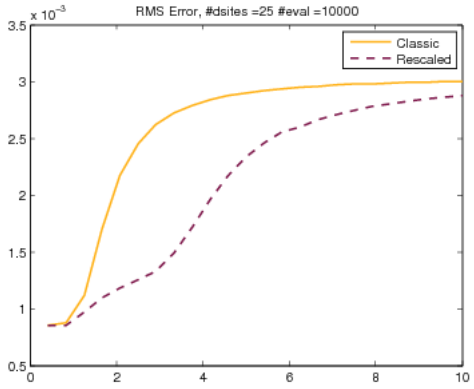

Fig. 4 RMSE behavior at different values of the shape parameter for interpolation of the Franke function with W2 radial function.

$$f(x,y) = \frac{3}{4}\mathrm{e}^{-\frac{1}{4}((9x-2)^2+(9y-2)^2)} + \frac{3}{4}\mathrm{e}^{-\frac{1}{49}(9x+1)^2-\frac{1}{10}(9y+1)} + \frac{1}{2}\mathrm{e}^{-\frac{1}{4}((9x-7)^2+(9y-3)^2)} - \frac{1}{5}\mathrm{e}^{-(9x-4)^2-(9y-7)^2}. \tag{4}$$

Similar results can be obtained by using different radial basis functions as studied in [7, 10].

Remarks. On looking to the way in which the interpolant is constructed and the previous figures we can observe

- The interpolant is smooth even for small radii of the support.
- Thanks to the normalization introduced in (3), the method can choose for each x_m a strategy to locally select the shape parameter ε in order to take into account the data points distribution. In the paper [7] the choice is made so that the local

radius of the compactly supported kernel gives a constant number of neighbors. This strategy fails when the points are uniformly distributed while gives much better results when the points are not equidistributed. In this second case we can in fact consider different radii and neighbor points.

3.1 The rescaled kernel

Since our interest is the study of the rescaled interpolant as a new kernel which has an associated native space we start by the following observation. An interpolation process consists in approximating the function f by its interpolant, say P_f, that is, for a constant $k_0 \in \mathbb{R}$,

$$P_f(x) \approx f(x), \ \forall\, x \in \Omega\,.$$

where equality holds for $x \in X$. Equivalently we can say that

$$P_f(x) - f(x) \approx k_0, \ \forall\, x \in \Omega\,.$$

Assuming that $f \neq 0$, then

$$\frac{P_f(x) - f(x)}{f(x)} \approx \frac{k_0}{f(x)} = k_0, \ \forall\, x \in \Omega\,, \tag{5}$$

the last equality holds when $f \equiv 1$. Hence, assuming that $f = k_1$, then from (5) we get

$$P_{k_1}(x) - k_1 \approx k_0, \ \forall\, x \in \Omega\,, \tag{6}$$

where we used k_0 to describe the new constant of the right hand side.

Proposition 1 *The relation $\approx$ induced by (6) is an equivalence relation.*

Proof. Let $f, g, h : \Omega \to \mathbb{R}$ and X be a set of distinct points of Ω.

- Reflexivity. Since $f = f$, then $f(\Omega) = f(\Omega)$ implies $f(X) = f(X)$ and so $f \approx f$.
- Symmetry. Let $f \approx g$, then $f(X) = g(X)$ that can be read from right to left and so $g \approx f$.
- Transitivity. Let $f \approx g$ and $g \approx h$. This means $f(X) = g(X)$ and $g(X) = h(X)$. Hence $f(X) = h(X)$, that is $f \approx h$.

This concludes the proof. □

We can use the transitive property of $\approx$ to combine (5) and (6) to get

$$\begin{aligned} \frac{P_f(x)}{f(x)} - k_1 &\approx P_{k_1}(x) - k_1 \\ \frac{P_f(x)}{f(x)} &\approx P_{k_1}(x) \\ \frac{P_f(x)}{P_{k_1}(x)} &\approx f(x)\,. \end{aligned}$$

Therefore, functions of the form $\dfrac{P_f(x)}{P_{k_1}(x)}$ are rescaled interpolants.

In our setting, we can notice that both $P_f(x)$ and $P_g(x)$ are constructed by using the kernel K with associate native space $\mathcal{N}_K$. In order to identify the native space associated to the rescaled interpolant, we may proceed as follows. Letting

$$\hat{P}_f(x) = \frac{P_f(x)}{P_g(x)}, \tag{7}$$

we may introduce a new kernel, say K_r, associated to the rescaled interpolant, from which we will characterize the associated native space $\mathcal{N}_{K_r}$.

Observing that

$$\hat{P}_f(x) = \frac{P_f(x)}{P_g(x)} = \sum_{j=1}^{N} c_j \frac{K(x,x_j)}{\sum_{i=1}^{N} d_i K(x,x_i)} \tag{8}$$

and recalling that the denominator is the interpolant of the constant function $g(x) = 1$, we have

$$\hat{P}_f(x) = \sum_{j=1}^{N} c_j \left[\frac{K(x,x_j)}{\sum_{i=1}^{N} d_i K(x,x_i) \sum_{i=1}^{N} d_i K(x_j,x_i)} \right]$$

Denoting $q(x) = \sum_{i=1}^{N} d_i K(x,x_i)$, then the square brakets can be re-written as the set of functions

$$\frac{K(x,x_j)}{q(x)\cdot q(x_j)}, \quad j = 1,\ldots,N$$

which can be interpreted as a *new basis* for the rescaled interpolant.

This theorem finds application in our setting.

Theorem 1 (cf. [1]). *Let $K : \Omega \times \Omega \to \mathbb{R}$ be a (strictly) positive definite kernel. Let $s : \Omega \to \mathbb{R}$ be a continuous and nonvanishing function in Ω. Then*

$$K_s(x,y) = s(x)s(y)K(x,y) \tag{9}$$

is (strictly) positive definite.

In fact, letting P_g the interpolant of the constant $g \equiv 1$ which is by construction continuous in Ω, if it is also non-vanishing, then $s = 1/P_g$. But this property follows from the general error estimation (2), since we can find a set of points X so that $\|P_g(x) - g\|_\infty < g$ that implies $P_g(x) \neq 0$, $\forall x \in \Omega$.

It follows that we can consider $s = 1/P_g$, which is continuous, non-vanishing on Ω and consider the *rescaled kernel*

$$K_r(x,y) = \frac{1}{P_g(x)} \frac{1}{P_g(y)} K(x,y) \tag{10}$$

which turns out to be (strictly) positive definite and we will denote its associate native space by $\mathcal{N}_{K_r}$.

This discussion shows that the rescaled kernel K_r is a kernel approximation process which is well-posed and preserves the properties of the kernel K.

Moreover, we can rewrite the rescaled interpolant as

$$\begin{aligned}\hat{P}_f(x) &= \sum_{j=1}^{N} c_j K_r(x,x_j) \\ &= \sum_{j=1}^{N} c_j \left\{ \frac{1}{\sum_{i=1}^{N} d_i K(x,x_i)} \frac{1}{\sum_{i=1}^{N} d_i K(x_i,x)} \right\} K(x,x_j) \\ &= \sum_{j=1}^{N} c_j \frac{1}{\sum_{i=1}^{N} d_i K(x,x_i)} K(x,x_j)\,,\end{aligned}$$

since $\frac{1}{\sum_{i=1}^{N} d_i K(x,x_i)} = 1\,,\ \forall x \in \Omega$ is the interpolant of the function $g = 1$. This formulation allows to prove formally that the rescaled interpolant reproduces the constants.

Theorem 2. *Let K be a strictly positive definite kernel, $f : \Omega \to \mathbb{R}\backslash\{0\}$ such that $f(x) = a$, $a \in \mathbb{R}\backslash\{0\}$ and $X = \{x_1,\dots,x_N\} \subset \Omega$. Then the associated rescaled interpolant (3) (or equivalently (8)) is such that $\hat{P}_{r,a}(x) = a,\ \forall x$.*

Proof. The interpolation conditions give the linear system

$$A_r \mathbf{c} = \mathbf{a}$$

where $\mathbf{a} = (a,\dots,a)^T$ and A_r denotes the collocation matrix w.r.t. the rescaled basis of the function $f(x) = a$. The previous system can be written as

$$\begin{aligned} a \cdot \left(\frac{1}{a} A_r \mathbf{c} \right) &= a \cdot \mathbf{1}\,, \\ \frac{1}{a} A_r \mathbf{c} &= \mathbf{1}\,. \end{aligned}$$

where $\mathbf{1} = (1,\dots,1)^T$. Hence, denoting as $\hat{P}_{r,a}$ the rescaled interpolant of the constant a

$$\hat{P}_{r,a}(\cdot) = a \cdot \hat{P}_g(\cdot) = a \cdot \frac{P_g(\cdot)}{P_g(\cdot)} = a\,, \tag{11}$$

as required. □

Obviously the previous results holds for $a = 0$. This comes immediately from (11).

4 Rescaling is equivalent to the Shepard's method

We take into account here a different point of view. We firstly compute the cardinal function form of the interpolatant, then we show the connection with the Shepard's method (see e.g. [8, §23.1]) and provide a stability bound based on the Lebesgue constant. We need to recall the following result (cf e.g. [8, §14.2] or [13]).

Proposition 2 *For any set $X_N = \{x_1, \dots, x_N\} \subset \Omega$ of pairwise distinct points, there exists a unique* `cardinal basis` $U = \{u_j\}_{j=1}^N$ *of the span$\{K(\cdot,x), x \in X_N\}$, i.e. a set of functions such that $u_j(x_i) = \delta_{ij}$, $1 \le i, j \le N$.*

Using the basis U, the standard interpolant of a function $f \in \mathscr{H}$ can be written in the form $P_f = \sum_{j=1}^N f(x_j) u_j$, and the interpolant of the function $g \equiv 1$ reads as $P_g = \sum_{j=1}^N u_j$. The rescaled interpolant of f then takes the form

$$\hat{P}_f = \frac{\sum_{j=1}^N f(x_j) u_j}{\sum_{k=1}^N u_k} = \sum_{j=1}^N f(x_j) \frac{u_j}{\sum_{k=1}^N u_k} =: \sum_{j=1}^N f(x_j) \hat{u}_j,$$

where we introduced the N functions $\hat{u}_j := u_j / \left(\sum_{k=1}^N u_k\right)$. These functions are still cardinal functions, since $\hat{u}_j(x_i) = \delta_{ij}$, but they do not belong to the subspace span$\{K(\cdot,x), x \in X_N\}$, in general. But they form a partition of unity, in the sense that, for all $x \in \Omega$, we have

$$\sum_{j=1}^N \hat{u}_j(x) = \sum_{j=1}^N \frac{u_j(x)}{\sum_{k=1}^N u_k(x)} = \frac{\sum_{j=1}^N u_j(x)}{\sum_{k=1}^N u_k(x)} = 1.$$

This construction proves the following result.

Proposition 3 *The rescaled interpolation method is a Shepard's method, where the weight functions are defined as $\hat{u}_j = u_j / \left(\sum_{k=1}^N u_k\right)$, $\{u_j\}_j$ being the cardinal basis of span$\{K(\cdot,x), x \in X\}$.*

Remarks

- Looking at Figures 2 and 3, we notice the typical "flat-spot" behaviour of the Shepard's approximation.
- Although this connection allows to relate our work with other existing methods, we remark that it also highlights some possibly severe limitations in the present approach. Indeed, the main reason to consider Shepard's methods relies on the easy computation of the weight functions $\{\hat{u}_j\}_j$, which are usually constructed by solving a small linear system instead of a full interpolation problem. In our case, instead, the computation of the weights requires the computation of the cardinal basis, that is the solution of the full interpolation problem. In this view, we can't expect good numerical results when the number of points increases. On the other hand, when the method is coupled with a *partition of unity* procedure, as will be discussed in the next sections, the systems to be solved are of small size, hence a stable computation of the interpolation of $g = 1$ can be expected.

From this construction we can easily derive stability bounds for the rescaled interpolation process. In fact, as happens in polynomial interpolation by using the cardinal functions, we can define the Lebesgue function $\Lambda_N(x) := \sum_{j=1}^N |u_j(x)|$, and its maximum over Ω, $\lambda_N := \|\Lambda_N\|_\infty$. that is the *Lebesgue constant* which controls the stability of the interpolation process. In fact, for any $x \in \Omega$

$$|P_f(x)| = \left| \sum_{j=1}^N f(x_j) u_j(x) \right| \leq \left(\sum_{j=1}^N |u_j(x)| \right) \|f\|_{\infty,X} \leq \lambda_N \|f\|_{\infty,X}.$$

Extending the setting to our case, and by using the rescaled cardinal functions $\{\hat{u}_j\}_j$ instead of the classical cardinals, we can write

$$\hat{\Lambda}_N(x) := \sum_{j=1}^N |\hat{u}_j(x)|, \;\; \hat{\lambda}_N := \|\hat{\Lambda}_N\|_\infty,$$

which gives the stability bound

$$\|\hat{P}_f\|_\infty \leq \hat{\lambda}_N \|f\|_{\infty,X}.$$

Hence, to quantify the stability gain of the rescaled interpolation process over the standard one, we can simply compare the behavior of $\hat{\lambda}_N$ and λ_N. Numerical experiments showing this comparison are presented in Section 6.

5 Application to PUM

Given the domain $\Omega \subset \mathbb{R}^d$, we consider its partition $\{\Omega_k \subset \Omega,\, k = 1, \ldots, n\}$ with Ω_k that possibly overlap, such that $\Omega \subseteq \cup_{k=1}^n \Omega_k$. We then consider compactly supported functions w_k with $\texttt{supp}(w_k) \subseteq \Omega_k$, forming a partition of Ω, that is

$$\sum_{k=1}^n w_k(x) = 1\,, \;\; \forall x \in \Omega\,. \tag{12}$$

Then we construct of a *local interpolant*, p_k, in RBF form

$$p_k(x; X_k) = \sum_{j=1}^{n_k} c_j^{(k)} \Phi_j^{(k)}(x)\,, \tag{13}$$

where X_k is a set of distinct points of Ω_k having $n_k = |X_k|$ as its cardinality and $\Phi^{(k)}$ the RBF kernel at Ω_k. The global interpolant on Ω can be written as

$$P_f(x) = \sum_{k=1}^n p_k(x; X_k) w_k(x), \;\; x \in \Omega\,. \tag{14}$$

If the local fit interpolates at a given data points, that is $p_k(x_l) = f(x_l)$, then thanks to the partition of unity property (12) we can conclude that the global fit is also interpolating at the same point

$$P_f(\mathbf{x}_l) = \sum_{k=1}^{n} p_k(x_l; X_k) w_k(x_l) = \sum_{k=1}^{n} f(x_l) w_k(x_l) = f(x_l)\,.$$

We can apply the rescaled interpolant to this framework as follows

- by applying the rescaling to the global interpolant (14);
- or by applying the rescaling to every local interpolant (13).

The first approach is equivalent to apply the PUM for interpolating the constant function 1. Hence, it makes sense to rescale *every* local interpolant. The application of the rescaling to every local interpolant of the form (13) gives a global rescaled interpolant of the form

$$P_f(x) = \sum_{k=1}^{n} \hat{R}_k(x; X_k) w_k(x), \;\; x \in \Omega\,. \tag{15}$$

with

$$\hat{R}_k(x; X_k) = \sum_{j=1}^{n_k} c_j^{(k)} \frac{\Phi_j^{(k)}(x)}{P_1^{(k)}(x)} = \sum_{j=1}^{n_k} c_j^{(k)} \frac{\Phi_j^{(k)}(x)}{\sum_{l=1}^{n_k} d_l^{(k)} \Phi_l^{(k)}(x)}\,,$$

where the coefficients $d_l^{(k)}$ are chosen so that $\sum_{l=1}^{n_k} d_l^{(k)} \Phi_l^{(k)}(x) = 1, \;\; \forall x \in X_k\,.$

6 Numerical examples

In this Section the rescaled method is tested in different instances with the aim of analyzing its performance and validate our understandings. The first set of experiments, presented in the next subsection, compare the standard and rescaled Lebesgue functions, as defined in Section 4. In this case the tests are limited to a small number N of samples in order to be able to stably compute the cardinal functions, and the results have to be intended as theoretical investigation of the relation between the two Lebesgue functions. In Section 6.2, instead, we focus on the comparison between the the PUM and the Rescaled-PUM, and we consider a more realistic number of points. This is possible by limiting the number of points in each subdomain in the partition of unity, so that a stable computation of the rescaled interpolant is numerically feasible.

6.1 Comparison of the standard and rescaled Lebesgue functions

In these experiments we compare the standard Lebesgue function with that of the rescaled one. Since we need to directly compute the cardinal functions, which is a seriously unstable operation, we keep the example as simple as possible to avoid comparing the effect of the ill-conditioning. To this end, we take $\Omega = [-1,1]$, a small number of fixed node points (equally spaced) and two classical kernels: the Gaussian kernel (globally supported and $\mathscr{C}^\infty$) and the Wendland W2 kernel, which is compactly supported and $\mathscr{C}^2$. The computation of the standard and rescaled Lebesgue functions has been repeated for $\varepsilon = 0.5, 1, 4, 8$ (Gaussian kernel) and $\varepsilon = 0.5, 1, 2, 4$ (Wendland kernel), as shown in Figures 5 and 6. In the latter case the behavior of the Lebesgue function does not change for bigger values of the shape parameter, that is why we stopped at $\varepsilon = 4$. Taking other set of nodes (like random or Chebyshev points), due to the above observation on the computation of the cardinal functions, we indeed observed essentially a similar behaviour for the Gaussian kernel and opposite for the Wendland kernel. See Figure 7 for a simple example with Chebsyshev points.

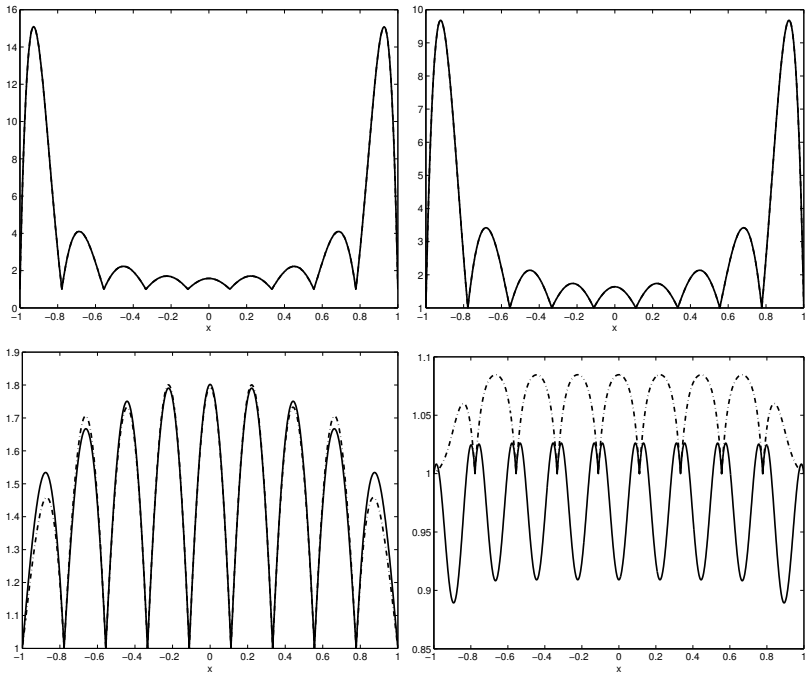

Fig. 5 Standard Lebesgue function (solid line) and the rescaled Lebesgue function (dotted line) for the Gaussian kernel on 10 equally spaced points of $[-1,1]$ with different ε. From top left to bottom right, $\varepsilon = 0.5, 1, 4, 8$.

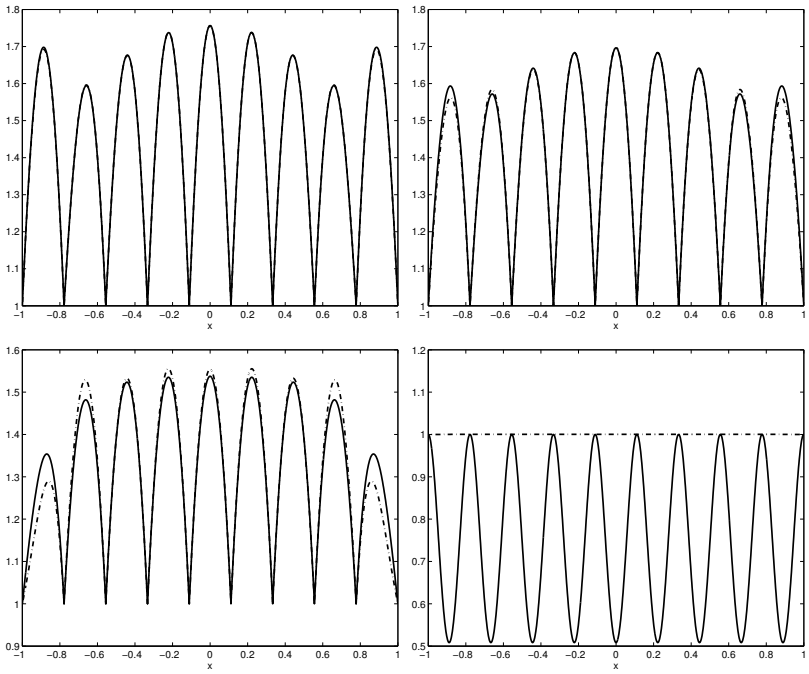

Fig. 6 Comparison between the standard Lebesgue function (solid line) and the rescaled Lebesgue function (dotted line) for the $\mathscr{C}^2$ Wendland kernel on 10 equally spaced points of $[-1,1]$ with different ε. From top left to bottom right, $\varepsilon = 0.5, 1, 2, 4$.

Similar behaviour can be obseverd in the two dimensional setting. In Figure 8 we show the comparison between the Lebesgue functions for the Wendand function with standard cardinal functions and the rescaled ones on the cardiod contained in $[-1,1]^2$.

In Figure 9 we did a similar test on the square. We have chosen $\varepsilon = 3.85$ because we wanted to show that bigger values of ε are meaningless. The reason of this relies in the support of the Wendland function which is $1/\varepsilon$. In the example, we have taken 25 equally spaced points on the square, so that for values of ε bigger than $\varepsilon_M = 2$ the cardinal functions have disjoint support. Therefore for values of $\varepsilon \geq 2\varepsilon_M$ we can not assume that the interpolant of the function 1 is always not vanishing since some points of the domain fall outside the support of the cardinal functions (giving a value 0 of the interpolant). This explains also why in the one dimensional case the cardinal functions with $\varepsilon = 4$ give a Lebesgue function identically equal to 1.

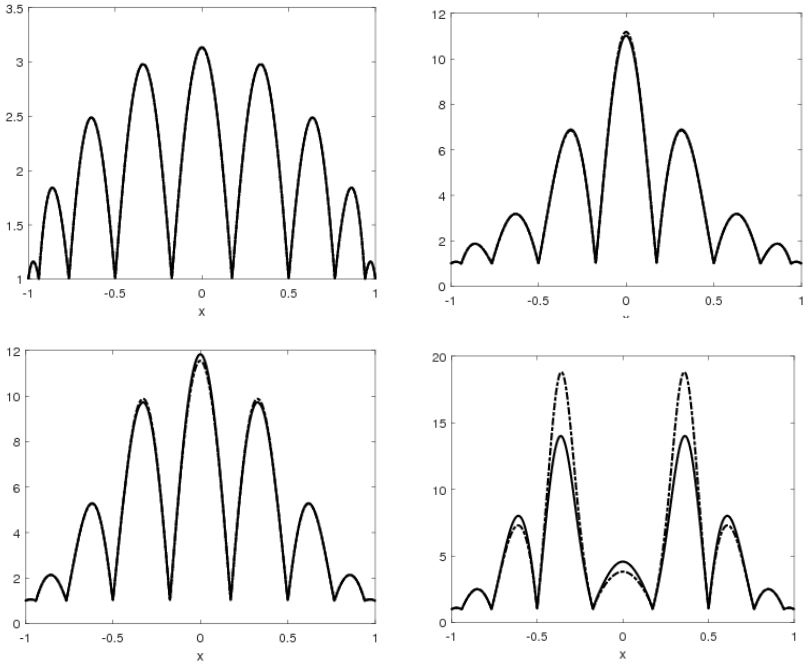

Fig. 7 Standard Lebesgue function (solid line) and the rescaled Lebesgue function (dotted line) for the Gaussian kernel (first row) and Wendland kernel (second row) on 10 Chebyshev points of $[-1,1]$ with $\varepsilon = 0.5$ (left) and $\varepsilon = 2$ (right).

6.2 Comparison between PUM and Rescaled-PUM

We investigate here the relation between the PUM and the Rescaled-PUM (R-PUM). Two different experiments are analyzed. First, we compare the two methods for a fixed and relatively small number N of samples, in order to highlight the behavior of the two methods with respect to changes in the shape parameter of the kernel. Then we let N increase and we analyze the convergence profile of the two methods, both for a fixed and an optimized shape parameters.

To test the two methods we use a C^2 Wendland kernel K on bivariate domains $\Omega \subset \mathbb{R}^2$, and a partition of unity defined on a covering of Ω made by balls Ω_j of constant radius and equally spaced centers, such that the resulting number of subdomains is

$$n^2 = \lfloor \frac{N}{4} \rfloor, \tag{16}$$

where $\lfloor \cdot \rfloor$ denotes the integer part. This covering is then equipped with a partition of unity defined by weight functions

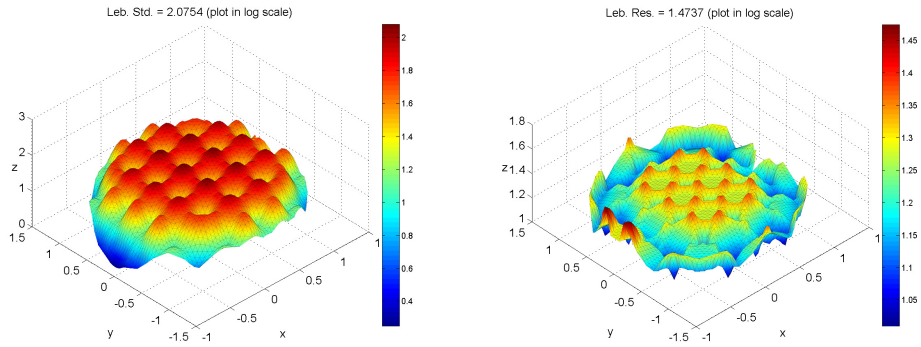

Fig. 8 Comparison between the Lebesgue function with standard basis (Left) and the rescaled one (right) for the $\mathscr{C}^2$ Wendland kernel on the cardiod with $\varepsilon = 3$.

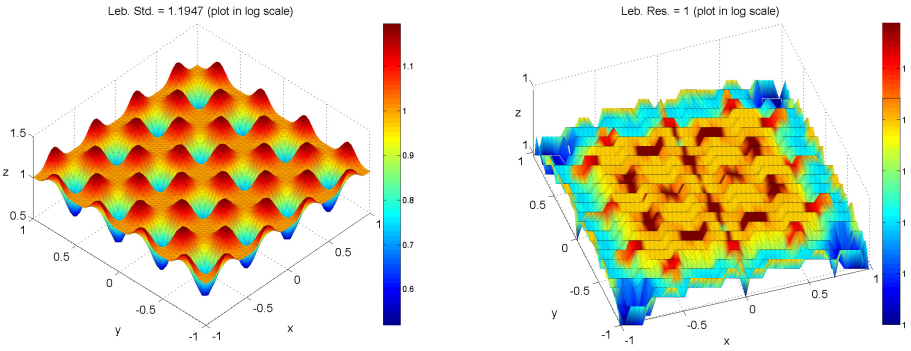

Fig. 9 Comparison between the Lebesgue function with standard basis (Left) and the rescaled one (right) for the $\mathscr{C}^2$ Wendland kernel on the square with $\varepsilon = 3.85$.

$$w_j(x) = \frac{K(x,x_j)}{\sum_{k=1}^{n} K(x,x_k)},$$

K being the Wendland kernel itself.

The error is measured in both cases on a grid of 100×100 uniformly distributed points in the convex hull of the sample set X. We consider both the maximal absolute error and the root mean square error (RMSE).

The algorithm is implemented using a *block-based* partition of unity algorithm presented in [4], which leads to a faster evaluation of the interpolant.

In both experiments, we use as a target function the 2d *Askley's test function* (well-known for testing optimization algorithms [11]), which is defined as

$$f(x,y) = -20\, e^{-0.2\sqrt{0.5(x^2+y^2)}} - e^{0.5(\cos(2\pi x)+\cos(2\pi y))} + 20 + e. \qquad (17)$$

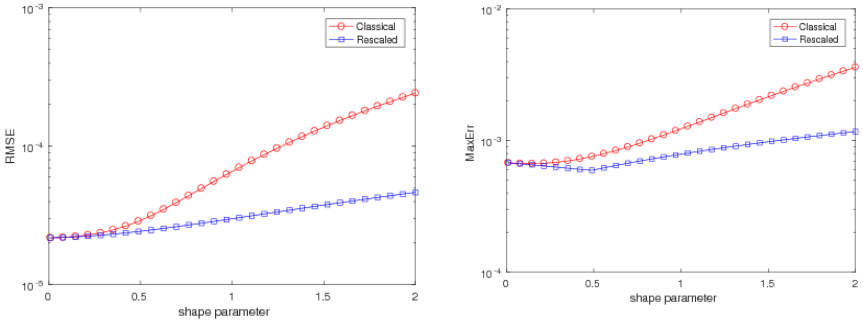

Fig. 10 RMSE (Left) and MAXERR (Right) for the classical and the rescaled PU for $\varepsilon \in [0.01, 2]$ chosen with Trial&Error.

For the first test, we consider $N = 10^3$ Halton points on the disk Ω of radius 0.5 and centered in $(0.5, 0.5)$. The two methods are tested on 30 values of the shape parameter $\varepsilon \in [0.01, 2]$. The results are shown in the Figure 10. For both error measurements, the PUM and R-PUM attain the same error for a flat kernel ($\varepsilon \approx 0$), while the R-PUM gives better results for a localized one ($\varepsilon = 2$). These results are easily explained, since a localized interpolant is smoothed by the rescaling procedure, and a decreasing of the error can be obtained also in the areas of Ω not well covered by X. The differences between the two methods are more evident on looking at the RMSE, which is a more global measurement than the pointwise maximal error.

We extend now the above comparisons by investigating the convergence behavior as the number N of function samples increases. We let the number n of subdomains to increase along with N (by using the formula (16)), such that each subdomain contains a nearly constant number of points. We remark, as pointed out earlier, that this is the only viable approach to ensure a stable computation of the interpolant of $g = 1$ with the direct method, while an increased size of the local linear system could possibly lead to severe ill-conditioning.

The kernel and partition of unity are as above, and we again address the approximation of the Askley's test function. For simplicity, we use again Halton points on the unit square $[0,1]^2$. We test the method for $N = 10^3, 2 \cdot 10^3, \ldots, 50 \cdot 10^3$. The resulting minimum, maximum, and mean number of points per subdomain are depicted in Figure 11.

At first, we compare the two methods by using a fixed shape parameter $\varepsilon = 1$: the interpolation errors are shown in Figure 12. In this experiment the R-PUM clearly exhibits the same convergence behavior of the PUM, with a slight improvement of about one order of magnitude in the best case. This effect is particularly evident when the RMSE error is computed, while the maximal error of the two methods behave nearly the same. This is again probably due to the smoothing effect of the rescaling procedure, which promotes a global accuracy (here, global for each subdomain) while it may over-regularize pointwise details, which leads to a reduced gain in the maximal error.

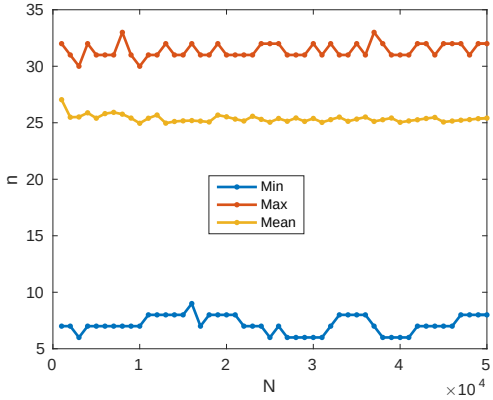

Fig. 11 Minimum, maximum, and mean number of points n per subdomain as the global number N of points increases for the experiments of Section 6.2

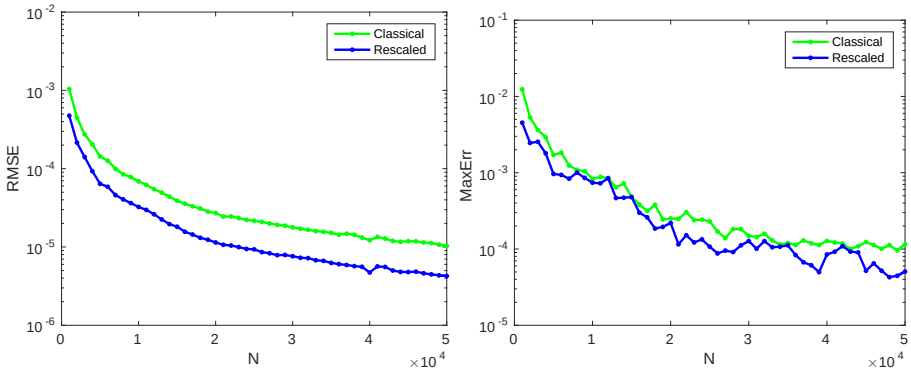

Fig. 12 RMSE (left) and maximal error (right) for the PUM (green) and the R-PUM (blue), as functions of the global number N of points, as described in Section 6.2

We then repeat the same experiment with shape parameter validation. Namely, the shape parameter is optimized by testing 30 equally spaced values in $[0.01, 2]$, and by picking, for each N, the value which minimize the error. Both root mean square error and maximal error are used, producing two different sequences of parameters for each method. The experiment is restricted in this case to a maximal value $N = 20 \cdot 10^3$.

In Figure 13 we report the maximal errors and the corresponding selected shape parameters, while in Figure 14 we show the same results with respect to the RMS error. It is worth noticing that when the RMSE is used as an error indicator, the PUM and R-PUM select exactly the same shape parameter (the green and blue lines overlap), while when the maximal error is employed the two methods select notably different parameters. The results highlight once more that rescaled method behaves better than the standard one with respect to the RMSE, while the two are essentially

equivalent with respect to the maximal error. In both cases, the convergence order for this setting appear to be essentially the same.

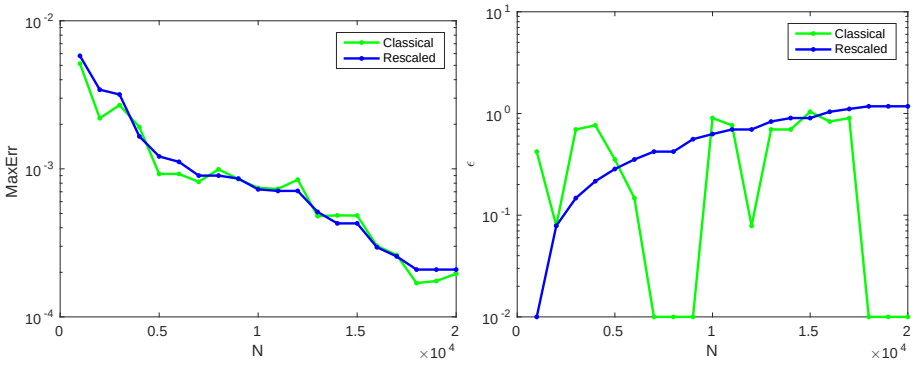

Fig. 13 Maximal error (left) and corresponding value of the shape parameter (right) for the PUM (green) and the R-PUM (blue), as functions of the global number N of points, as described in Section 6.2

11. As a final remark, we observe that letting T_{RPU} the evaluation time of the

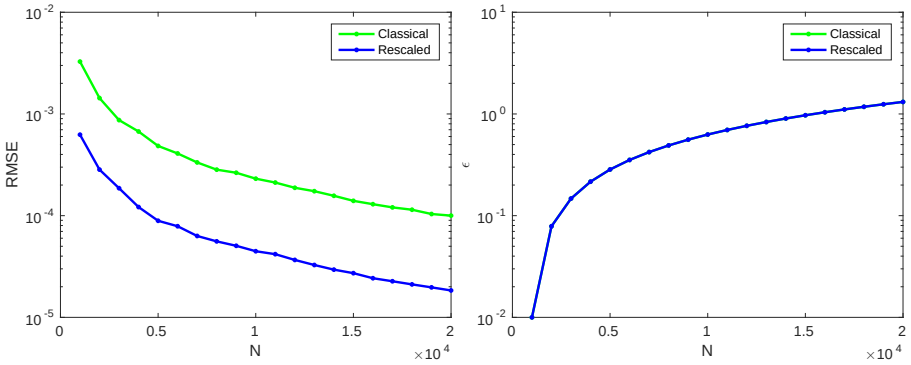

Fig. 14 RMS error (left) and corresponding value of the shape parameter (right) for the PUM (green) and the R-PUM (blue), as functions of the global number N of points, as described in Section 6.2

rescaled interpolant and T_{PU} that of the classical one, we generally saw that $T_{RPU} < c\,T_{PU}$ with $c \approx 1.05$.

7 Conclusions

In this paper we have studied more deeply the rescaled localized radial basis function interpolant introduced in [7]. We have proved that this interpolant gives a partition of unity method that reproduces constant functions, exactely as does the Shepard's method. One feature is that the shape parameter of the kernel can be chosen in a safe range, not too small and "relatively large", avoiding the classical numerical instability that occours when the shape parameter is too small, or a severe ill-conditioning in the opposite case.

The numerical experiments on the R-PUM show that the method allows to improve the maximal error over a classical PUM.

Acknowledgements . This work has been supported by the INdAM-GNCS funds 2016 and by the ex 60% funds, year 2015, of the University of Padova. We are greatful to the anonymous referees for their comments and observations, which allowed us to improve the paper.

References

1. Nachman Aronszajn: Theory of Reproducing Kernels. *Trans. Amer. Math. Soc.*, **68** (1950), 337–404.
2. L. Bos and S. De Marchi: Univariate Radial Basis Functions with Compact Support Cardinal Functions *East J. Approx.*, **14**(1) (2008), 69–80.
3. M. Buhmann: *Radial Basis Functions: Theory and Implementations*, Cambrige University Press, 2003.
4. R. Cavoretto, A. De Rossi and E. Perracchione: Efficient computation of partition of unity interpolants through a block-based searching technique. *arXiv:1604.04585* (2016).
5. A. De Rossi, E. Perracchione and E. Venturino: Fast strategy for PU interpolation: an application for the reconstruction of separatrix manifolds. *Dolomites Res. Notes Approx.*, Special Issue of Vol **9** - Kernel-based Methods and Function Approximation (2016), 3–12.
6. Stefano De Marchi and Robert Schaback: Nonstandard Kernelss and their Applications, *Dolomites Res. Notes Approx.*, **2** (2009), 16–43.
7. Simone Deparis, Davide Forti, Alfio Quarteroni: A rescaled localized radial basis function interpolation on non-cartesian and non-conforming grids. *SIAM J. Sci. Comp.*, **36**(6) (2014), A2745–A2762.
8. Gregory E. Fasshauer: *Meshfree Approximation Methods with Matlab*. World Scientific Publishing, Interdisciplinary Mathematical Sciences - Vol.6, (2007).
9. Gregory E. Fasshauer and Michael McCourt: *Kernel-based Approximation Methods using Matlab*. World Scientific Publishing, Interdisciplinary Mathematical Sciences - Vol.19, (2015).
10. Andrea Idda: *A comparison of some RBF interpolation methods: theory and numerics*, Master's degree thesis, Department of Mathematics, University of Padova (2015).
11. Oldenhuis Rody: Many test functions for global optimization algorithms. Matlab Central File Exchange (2015).
12. Holger Wendland: Fast evaluation of radial basis functions: methods based on partition of unity. in *Approximation theory X*, (St. Louis 2001), 473–483. Innov. Appl. Math, Vanderbilt Univ. Press, Nashville, Tennessee (2002).
13. Wu, Z. and Schaback, R.: Local error estimates for radial basis function interpolation of scattered data, *IMA J. Numer. Anal*, **13** (1993), 13–27.