On the rescaled method for RBF approximation[1]

Stefano De Marchi
Department of Mathematics - University of Padova
MAIA 2016 - Luminy (France)
September 21, 2016

DIPARTIMENTO MATEMATICA

[1] joint work with Andrea Idda (Verona) and Gabriele Santin (Stuttgart)

Outline

Introduction

RBF Approximation

1. Data: $\Omega \subset \mathbb{R}^n$, $X \subset \Omega$, test function f
   - $X = \{x_1, \ldots, x_N\} \subset \Omega$
   - $f_1, \ldots, f_N$, where $f_i = f(x_i)$

Introduction

RBF Approximation

1. Data: $\Omega \subset \mathbb{R}^n$, $X \subset \Omega$, test function f
   - $X = \{x_1, \ldots, x_N\} \subset \Omega$
   - $f_1, \ldots, f_N$, where $f_i = f(x_i)$
2. Approximation setting: kernel K_ε, $\mathcal{N}_K(\Omega)$, $\mathcal{N}_K(X) \subset \mathcal{N}_K(\Omega)$
   - kernel $K = K_\varepsilon$, positive definite and radial
     examples:
     - globally supported: $K_\varepsilon(x, y) = e^{-(\varepsilon\|x-y\|)^2}$ (*gaussian*),
     - locally supported: $K_\varepsilon(x, y) = (1 - \varepsilon^2\|x - y\|^2)_+^4[4\varepsilon^2\|x - y\|^2 + 1]$ ($C^2(\mathbb{R}^2)$ *Wendland*)
   - native space $\mathcal{N}_K(\Omega)$ (where K is the reproducing kernel)
   - finite subspace $\mathcal{N}_K(X) = \operatorname{span}\{K(\cdot, x) : x \in X\} \subset \mathcal{N}_K(\Omega)$

Introduction

RBF Approximation

1. Data: $\Omega \subset \mathbb{R}^n$, $X \subset \Omega$, test function f
   - $X = \{x_1, \ldots, x_N\} \subset \Omega$
   - $f_1, \ldots, f_N$, where $f_i = f(x_i)$
2. Approximation setting: kernel K_ε, $\mathcal{N}_K(\Omega)$, $\mathcal{N}_K(X) \subset \mathcal{N}_K(\Omega)$
   - kernel $K = K_\varepsilon$, positive definite and radial
     examples:
     - globally supported: $K_\varepsilon(x, y) = e^{-(\varepsilon\|x-y\|)^2}$ (*gaussian*),
     - locally supported: $K_\varepsilon(x, y) = (1 - \varepsilon^2\|x - y\|^2)_+^4[4\varepsilon^2\|x - y\|^2 + 1]$ ($C^2(\mathbb{R}^2)$ *Wendland*)
   - native space $\mathcal{N}_K(\Omega)$ (where K is the reproducing kernel)
   - finite subspace $\mathcal{N}_K(X) = \operatorname{span}\{K(\cdot, x) : x \in X\} \subset \mathcal{N}_K(\Omega)$

Aim

Find $P_f \in \mathcal{N}_K(X)$ s.t. $P_f \approx f$

The rescaled RBF interpolant

The rescaled interpolant

Deparis, Forti and Quarteroni [DFQ14, SISC 36(6)], proposed a new consistent Rescaled Localized RBF (RL-RBF) interpolant for large-scale problems based on Compactly Supported RBF (CSRBF) (with parallel implementation on 2d-3d non cartesian unstructered meshes).

The rescaled interpolant

Deparis, Forti and Quarteroni [DFQ14, SISC 36(6)], proposed a new consistent Rescaled Localized RBF (RL-RBF) interpolant for large-scale problems based on Compactly Supported RBF (CSRBF) (with parallel implementation on 2d-3d non cartesian unstructered meshes).

Construction

On X, let us consider the constant function $g(\boldsymbol{x}) = 1$ and let $P_g(\boldsymbol{x})$ be the corresponding kernel-based interpolant. Letting $P_f(\boldsymbol{x})$ the interpolant of f, then the rescaled interpolant is

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

The rescaled interpolant

Deparis, Forti and Quarteroni [DFQ14, SISC 36(6)], proposed a new consistent Rescaled Localized RBF (RL-RBF) interpolant for large-scale problems based on Compactly Supported RBF (CSRBF) (with parallel implementation on 2d-3d non cartesian unstructered meshes).

Construction

On X, let us consider the constant function $g(\boldsymbol{x}) = 1$ and let $P_g(\boldsymbol{x})$ be the corresponding kernel-based interpolant. Letting $P_f(\boldsymbol{x})$ the interpolant of f, then the rescaled interpolant is

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

OBS: obviuosly

$$\hat{P}_f(X) = f(X).$$

Properties

- The interpolant is smoother even for small radii of the support (↪ next slide)

Properties

- The interpolant is smoother even for small radii of the support ($\hookrightarrow$ next slide)
- Thanks to the normalization, for each $\boldsymbol{x}_m$ they provide a strategy to select locallythe shape parameter ϵ that considers the data distribution ($\hookrightarrow$ i.e the local radius of the CSRBF gives constant neighbor points) (cf. [DFQ14]).

Properties

- The interpolant is smoother even for small radii of the support ($\hookrightarrow$ next slide)
- Thanks to the normalization, for each $\boldsymbol{x}_m$ they provide a strategy to select locallythe shape parameter ϵ that considers the data distribution ($\hookrightarrow$ i.e the local radius of the CSRBF gives constant neighbor points) (cf. [DFQ14]).
- When points are uniformly distributed the previous strategy falls in a priori selection of the radius.

Properties

- The interpolant is smoother even for small radii of the support ($\hookrightarrow$ next slide)
- Thanks to the normalization, for each $\boldsymbol{x}_m$ they provide a strategy to select locallythe shape parameter ϵ that considers the data distribution ($\hookrightarrow$ i.e the local radius of the CSRBF gives constant neighbor points) (cf. [DFQ14]).
- When points are uniformly distributed the previous strategy falls in a priori selection of the radius.
- When points are not uniformly distributed the previous strategy gives better results (using different circles diameters and neighbor points).

Properties

- The interpolant is smoother even for small radii of the support ($\hookrightarrow$ next slide)
- Thanks to the normalization, for each $\boldsymbol{x}_m$ they provide a strategy to select locallythe shape parameter ϵ that considers the data distribution ($\hookrightarrow$ i.e the local radius of the CSRBF gives constant neighbor points) (cf. [DFQ14]).
- When points are uniformly distributed the previous strategy falls in a priori selection of the radius.
- When points are not uniformly distributed the previous strategy gives better results (using different circles diameters and neighbor points).

Remark

No theoretical study of the properties of the new interpolant were provided!

First example

As starting simple illustrative example we want to interpolate $f(x) = x$ on the interval $[0, 1]$ by using the W2 function at the points set $X = \{1/3, 1/2, 5/6\}$, $\varepsilon = 5$. Here the radius of the corresponding radial basis function is $\delta = 1/\epsilon$ so that on $[0, 1]$ the rescaled interpolant never vanishes

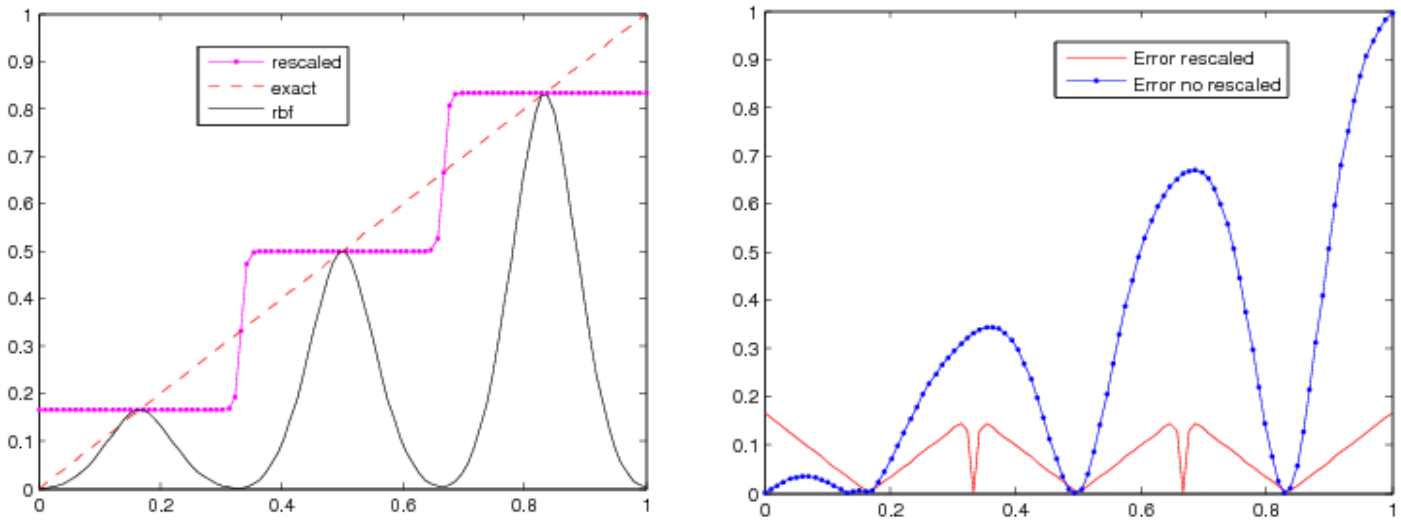


Figure: (left) interpolants and (right) the abs error

Second example

Again, $f(x) = x$ on $[0, 1]$ by using W2 at the points set $X = \{0, 1/6, 1/3, 1/2, 2/3, 5/6, 1\}$, $\varepsilon = 5$ ($\epsilon = 1/\delta$).

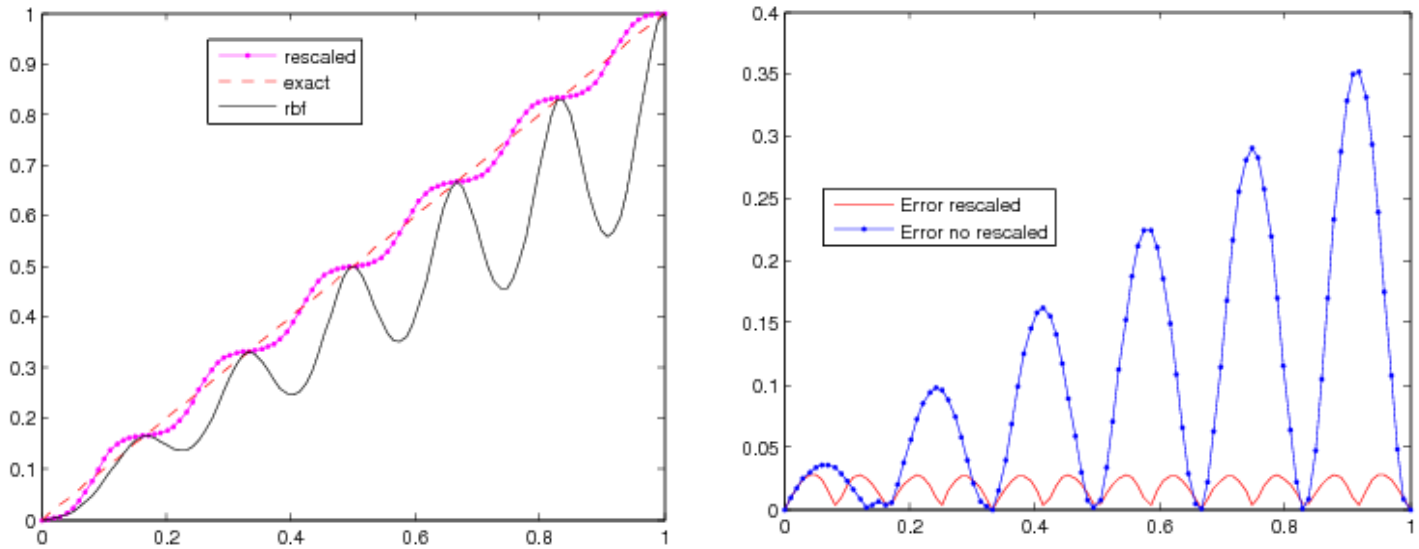


Figure: (left) interpolants and (right) the abs error

For more results see [DFQ14, Idda15].

An example on 2d

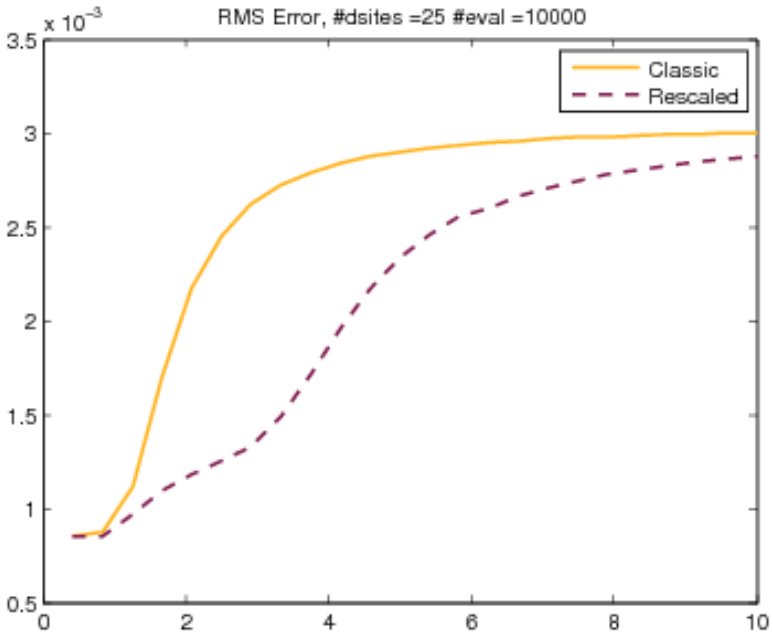


Figure: RMSE behavior at 30 values of the shape parameter in $[0.1, 10]$ for interpolation of the 2d Franke function (stationary) on a grid 5×5, again with W2.

The rescaled kernel

The rescaled kernel

1. Assume that $P_f(\boldsymbol{x})$ and $P_g(\boldsymbol{x})$ be constructed by using the kernel K (with associate native space $\mathcal{N}_K$).

The rescaled kernel

1. Assume that $P_f(\boldsymbol{x})$ and $P_g(\boldsymbol{x})$ be constructed by using the kernel K (with associate native space $\mathcal{N}_K$).
2. Writing

$$\hat{P}_f(\boldsymbol{x}) = \frac{P_f(\boldsymbol{x})}{P_g(\boldsymbol{x})} = \sum_{j=1}^{N} c_j \frac{K(\boldsymbol{x}, \boldsymbol{x}_j)}{\sum_{i=1}^{N} d_i K(\boldsymbol{x}, \boldsymbol{x}_i)}, \tag{2}$$

since $P_g(\boldsymbol{x}_j) = 1$, we have

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

The rescaled kernel

1. Assume that $P_f(\boldsymbol{x})$ and $P_g(\boldsymbol{x})$ be constructed by using the kernel K (with associate native space $\mathcal{N}_K$).
2. Writing

$$\hat{P}_f(\boldsymbol{x}) = \frac{P_f(\boldsymbol{x})}{P_g(\boldsymbol{x})} = \sum_{j=1}^{N} c_j \frac{K(\boldsymbol{x}, \boldsymbol{x}_j)}{\sum_{i=1}^{N} d_i K(\boldsymbol{x}, \boldsymbol{x}_i)}, \tag{2}$$

since $P_g(\boldsymbol{x}_j) = 1$, we have

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

But $P_g(\boldsymbol{x}) = \sum_{i=1}^{N} d_i K(\boldsymbol{x}, \boldsymbol{x}_i)$ then $\left\{ \frac{K(\boldsymbol{x}, \boldsymbol{x}_j)}{P_g(\boldsymbol{x}) \cdot P_g(\boldsymbol{x}_j)}, \quad j = 1, \ldots, N \right\}$ can be interpreted as a (new) basis for the rescaled interpolant.

The rescaled kernel (cont')

Theorem (Aronszajn50)

Let $K : \Omega \times \Omega \to \mathbb{R}$ be a (strictly) positive definite kernel. Let $s : \Omega \to \mathbb{R}$ a continuous and nonvanishing function on Ω. Then

$$K_s(\boldsymbol{x}, \boldsymbol{y}) = s(\boldsymbol{x})s(\boldsymbol{y})K(\boldsymbol{x}, \boldsymbol{y}) \tag{3}$$

is (strictly) positive definite.

The rescaled kernel (cont')

Theorem (Aronszajn50)

Let $K : \Omega \times \Omega \to \mathbb{R}$ be a (strictly) positive definite kernel. Let $s : \Omega \to \mathbb{R}$ a continuous and nonvanishing function on Ω. Then

$$K_s(\boldsymbol{x}, \boldsymbol{y}) = s(\boldsymbol{x}) s(\boldsymbol{y}) K(\boldsymbol{x}, \boldsymbol{y}) \tag{3}$$

is (strictly) positive definite.

Setting $s = 1/P_g$, which is continuous, non-vanishing on Ω, the rescaled kernel is then

$$\boxed{K_R(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{P_g(\boldsymbol{x})} \frac{1}{P_g(\boldsymbol{y})} K(\boldsymbol{x}, \boldsymbol{y})} \tag{4}$$

which turns out to be (strictly) positive definite and then we can consider its associate native space $\mathcal{N}_{K_R}$.

Native space (ideas)

Definition

A function $m \in \mathcal{N}_K$ is called a multiplier of $\mathcal{N}_K$ if the product $m \cdot f \in \mathcal{N}_K, \ \forall \ f \in \mathcal{N}_K$.

Native space (ideas)

Definition

A function $m \in \mathcal{N}_K$ is called a multiplier of $\mathcal{N}_K$ if the product $m \cdot f \in \mathcal{N}_K$, $\forall\ f \in \mathcal{N}_K$.

This definition provide the inclusion $\mathcal{N}_{K_R} \subset \mathcal{N}_K$ (since $1/P_g$ is a multiplier).

Native space (ideas)

Definition

A function $m \in \mathcal{N}_K$ is called a multiplier of $\mathcal{N}_K$ if the product $m \cdot f \in \mathcal{N}_K,\ \forall\ f \in \mathcal{N}_K$.

This definition provide the inclusion $\mathcal{N}_{K_R} \subset \mathcal{N}_K$ (since $1/P_g$ is a multiplier). The other inclusion could come from the following

Theorem (Aronszajn50)

Let K_1 and K_2 be two kernels on $\Omega \times \Omega$. Then $\mathcal{N}_{K_1} \subset \mathcal{N}_{K_2}$ if and only if for any $N \in \mathbb{N}$, $\boldsymbol{c} \in \mathbb{R}^N$ and points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ of Ω we have

$$\boldsymbol{c}^T A_1 \boldsymbol{c} \leq \boldsymbol{c}^T A_2 \boldsymbol{c}$$

with $(A_i)_{j,k} = K_i(\boldsymbol{x}_j, \boldsymbol{x}_k),\ i = 1, 2.$

Native space (ideas)

Definition

A function $m \in \mathcal{N}_K$ is called a multiplier of $\mathcal{N}_K$ if the product $m \cdot f \in \mathcal{N}_K,\ \forall\ f \in \mathcal{N}_K$.

This definition provide the inclusion $\mathcal{N}_{K_R} \subset \mathcal{N}_K$ (since $1/P_g$ is a multiplier). The other inclusion could come from the following

Theorem (Aronszajn50)

Let K_1 and K_2 be two kernels on $\Omega \times \Omega$. Then $\mathcal{N}_{K_1} \subset \mathcal{N}_{K_2}$ if and only if for any $N \in \mathbb{N}$, $\boldsymbol{c} \in \mathbb{R}^N$ and points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ of Ω we have

$$\boldsymbol{c}^T A_1 \boldsymbol{c} \leq \boldsymbol{c}^T A_2 \boldsymbol{c}$$

with $(A_i)_{j,k} = K_i(\boldsymbol{x}_j, \boldsymbol{x}_k),\ i = 1, 2.$

The Theorem in our case means to show that (or the reverse inequality)

$$\boldsymbol{c}^T A_K \boldsymbol{c} \leq \boldsymbol{c}^T A_{K_R} \boldsymbol{c}\,, \tag{5}$$

Rescaling gives a Shepard method

We start by writing the interpolant of a function $f \in \mathcal{N}_K$ using the cardinals $u_j(x_i) = \delta_{i,j}$, $\boxed{P_f = \sum_{j=1}^{N} f(\boldsymbol{x}_j) u_j}$, so that for $g \equiv 1$ we get

$$\boxed{P_g = \sum_{j=1}^{N} u_j}.$$

Rescaling gives a Shepard method

We start by writing the interpolant of a function $f \in \mathcal{N}_K$ using the cardinals $u_j(x_i) = \delta_{i,j}$, $\boxed{P_f = \sum_{j=1}^{N} f(\boldsymbol{x}_j) u_j}$, so that for $g \equiv 1$ we get $\boxed{P_g = \sum_{j=1}^{N} u_j}$. The rescaled interpolant is then

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

where we introduced the new cardinal functions $\hat{u}_j := \dfrac{u_j}{\sum_{k=1}^{N} u_k}$.

Rescaling gives a Shepard method

We start by writing the interpolant of a function $f \in \mathcal{N}_K$ using the cardinals $u_j(x_i) = \delta_{i,j}$, $\boxed{P_f = \sum_{j=1}^{N} f(\boldsymbol{x}_j) u_j}$, so that for $g \equiv 1$ we get $\boxed{P_g = \sum_{j=1}^{N} u_j}$. The rescaled interpolant is then

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

where we introduced the new cardinal functions $\hat{u}_j := \dfrac{u_j}{\sum_{k=1}^{N} u_k}$.

Corollary

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$\{K(\cdot, x), x \in X\}$.

Stability estimate

With the usual notation, we can define the Lebesgue function and constant for the rescaled interpolant

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

getting the stability bound

$$\boxed{\|\hat{P}_f\|_{\infty,\Omega} \leq \hat{\lambda}_N \|f\|_{\infty,X}.} \tag{6}$$

Stability estimate

With the usual notation, we can define the Lebesgue function and constant for the rescaled interpolant

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

getting the stability bound

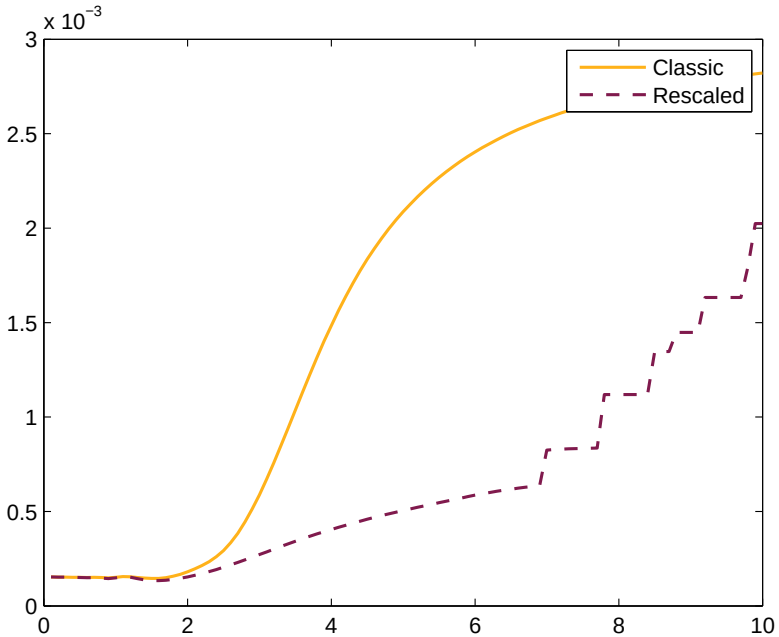
$$\boxed{\|\hat{P}_f\|_{\infty,\Omega} \leq \hat{\lambda}_N \|f\|_{\infty,X}.} \tag{6}$$

Hence, to quantify the stability gain of the rescaled interpolation process over the standard one, we can then compare the behavior of $\hat{\lambda}_N$ and λ_N ($\hookrightarrow$ see experiments)

Applications and numerical experiments

A first test

Comparison of RMSE for the classical and the rescaled interpolant on varying ε by *Trial&Error*



The stair behaviour of the rescaled interpolant is due to the "patching" algorithm used to avoid that $P_g(\mathbf{x}) = 0$

Application to PUM

$\Omega = \cup_{k=1}^{n} \Omega_k$ and compactly supported functions, $supp(w_k) \subseteq \Omega_k$,

$$\sum_{i=1}^{n} w_i(\boldsymbol{x}) = 1\,, \quad \forall \boldsymbol{x} \in \Omega\,. \tag{7}$$

The application of the rescaling to every local interpolant gives a global rescaled interpolant of the form

$$\boxed{P_f(\boldsymbol{x}) = \sum_{i=1}^{n} \hat{P}_i(\boldsymbol{x}) w_i(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega\,,} \tag{8}$$

with

$$\hat{P}_i(\boldsymbol{x}) = \sum_{j=1}^{n_i} c_j^{(i)} \frac{K^{(i)}(\boldsymbol{x}, \boldsymbol{x}_j)}{P_g^{(i)}(\boldsymbol{x})} = \sum_{j=1}^{n_i} c_j^{(i)} \frac{K^{(i)}(\boldsymbol{x}, \boldsymbol{x}_j)}{\sum_{l=1}^{n_i} d_l^{(i)} K^{(i)}(\boldsymbol{x}, \boldsymbol{x}_l)}\,, \text{with} \quad n_i = \#(X \cap \Omega_i)$$

where the coefficients $d_l^{(i)}$ are chosen so that
$\sum_{l=1}^{n_i} d_l^{(i)} K^{(i)}(\boldsymbol{x}, \boldsymbol{x}_l) = 1, \quad \forall \mathbf{x} \in \Omega_i\,.$

Rescaled PU (RPU): example

Consider the 2d *Askley's test function* [R15]

$$f(x,y) = -20\, e^{-0.2\sqrt{0.5(x^2+y^2)}} - e^{-0.5(\cos(2\pi x)+\cos(2\pi y))} + 20 + e \quad (9)$$

interpolated on 1000 Halton points on the disk centered in $(0.5, 0.5)$ and radius 0.5 with W2. As evalution points we took 10000 uniformly distributed points of the disk.

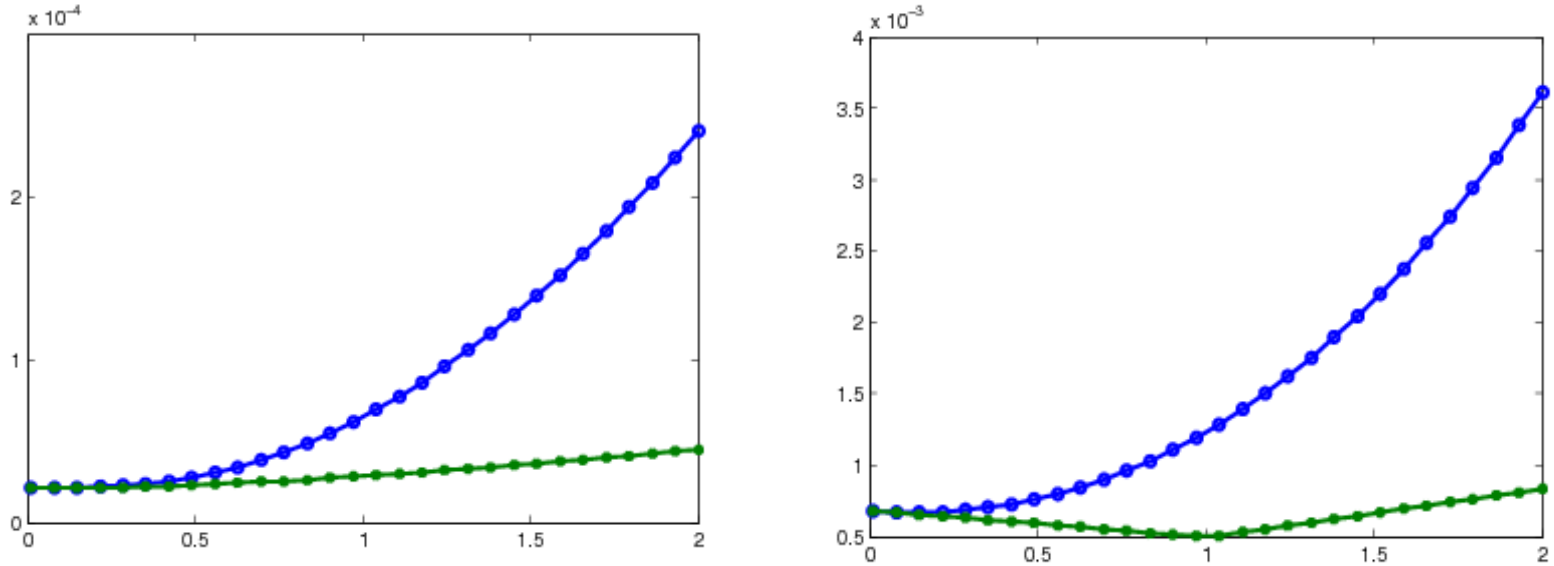


Figure: RMSE (Left) and MAXERR (Right) for the classical (blue) and the rescaled PU (green) for $\varepsilon \in [0.01, 2]$.

Remarks

Some numerical evidences

- RPU reaches the same precision of PU, but using a “thinner” point set X
- The evaluation time heuristically is $T_{RPU} < cT_{PU}$ with $c \approx 1.05$
- P_f and P_g share the same collocation matrix, so that the linear systems differ only by constant terms. Hence one can use specific algorithms to speed-up the evaluation step

Variably Scaled Kernels

- VSK, introduced in [BLRS, IMA JNA15], are intended to give more flexibility to RBF approximations.
- Let $c : \mathbb{R}^d \to (0, \infty)$ be a scale function. A VSK on $\mathbb{R}^d$ is

$$K_c(\boldsymbol{x}, \boldsymbol{y}) = K((\boldsymbol{x}, c(\boldsymbol{x})), (\boldsymbol{y}, c(\boldsymbol{y}))$$

If Φ is radial, the new kernel takes the form

$$K_c(\boldsymbol{x}, \boldsymbol{y}) := \Phi(\|\boldsymbol{x} - \boldsymbol{y}\|^2 + (c(\boldsymbol{x}) - c(\boldsymbol{y}))^2)$$

VSK (cont')

The scale function transforms a problem with data $\boldsymbol{x}_j \in \mathbb{R}^d$ to data location $(\boldsymbol{x}_j, c(\boldsymbol{x}_j)) \in \mathbb{R}^{d+1}$ and then use a fixed-scale kernel on $\mathbb{R}^{d+1}$.

VSK (cont')

The scale function transforms a problem with data $\boldsymbol{x}_j \in \mathbb{R}^d$ to data location $(\boldsymbol{x}_j, c(\boldsymbol{x}_j)) \in \mathbb{R}^{d+1}$ and then use a fixed-scale kernel on $\mathbb{R}^{d+1}$.

Letting $\sigma : \boldsymbol{x} \to (\boldsymbol{x}, c(\boldsymbol{x}))$ map from $\mathbb{R}^d$ into a d-dimensional submanifold $\sigma(\mathbb{R}^d)$ of $\mathbb{R}^{d+1}$

$$K_c(\boldsymbol{x}, \boldsymbol{y}) = K(\underbrace{(\boldsymbol{x}, c(\boldsymbol{x}))}_{\sigma(\boldsymbol{x})}, \underbrace{(\boldsymbol{y}, c(\boldsymbol{y}))}_{\sigma(\boldsymbol{y})})$$

VSK (cont')

The scale function transforms a problem with data $\boldsymbol{x}_j \in \mathbb{R}^d$ to data location $(\boldsymbol{x}_j, c(\boldsymbol{x}_j)) \in \mathbb{R}^{d+1}$ and then use a fixed-scale kernel on $\mathbb{R}^{d+1}$.

Letting $\sigma : \boldsymbol{x} \to (\boldsymbol{x}, c(\boldsymbol{x}))$ map from $\mathbb{R}^d$ into a d-dimensional submanifold $\sigma(\mathbb{R}^d)$ of $\mathbb{R}^{d+1}$

$$K_c(\boldsymbol{x}, \boldsymbol{y}) = K(\underbrace{(\boldsymbol{x}, c(\boldsymbol{x}))}_{\sigma(\boldsymbol{x})}, \underbrace{(\boldsymbol{y}, c(\boldsymbol{y}))}_{\sigma(\boldsymbol{y})})$$

Hence the interpolant satisfies

$$\boxed{P_{\sigma,f,X}(\boldsymbol{x}) = P_{1,f,\sigma(X)}(\boldsymbol{x}, c(\boldsymbol{x})) = P_{1,f,\sigma(X)}(\sigma(\boldsymbol{x}))\,.} \qquad (10)$$

Example: Classical vs VSK vs Rescaled

Franke test function, W2 sampled on 200 e.s.points of half unit sphere centered in $(0, 0, 0)$. The nodes in $\mathbb{R}^2$ are the projections on the unit disk, i.e. $c(\boldsymbol{x}) = \sqrt{1 - x_1^2 - x_2^2}$. The evaluation points in $\mathbb{R}^2$ are obtained by restricting the grid 100×100 of the square $[-1, 1]^2$ to the unit disk, while the points in $\mathbb{R}^3$ are obtained by the map $\sigma(\boldsymbol{x}) = (\boldsymbol{x}, c(\boldsymbol{x}))$ (in Figure 5 we show only 100 points). The shape parameter is $\varepsilon = 5$

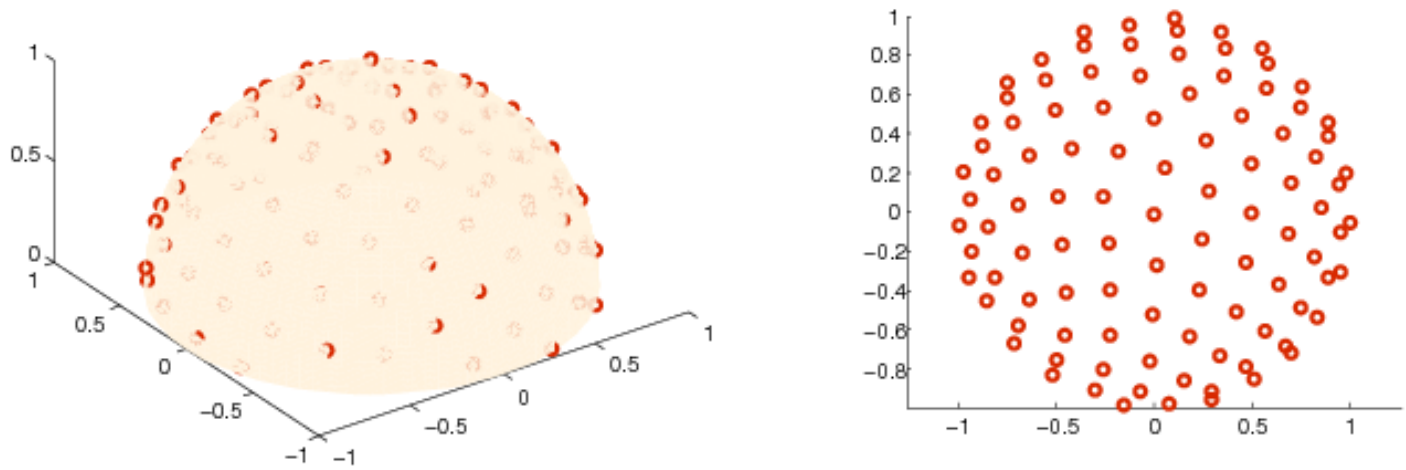


Figure: The points of the VSK example

Example (cont')

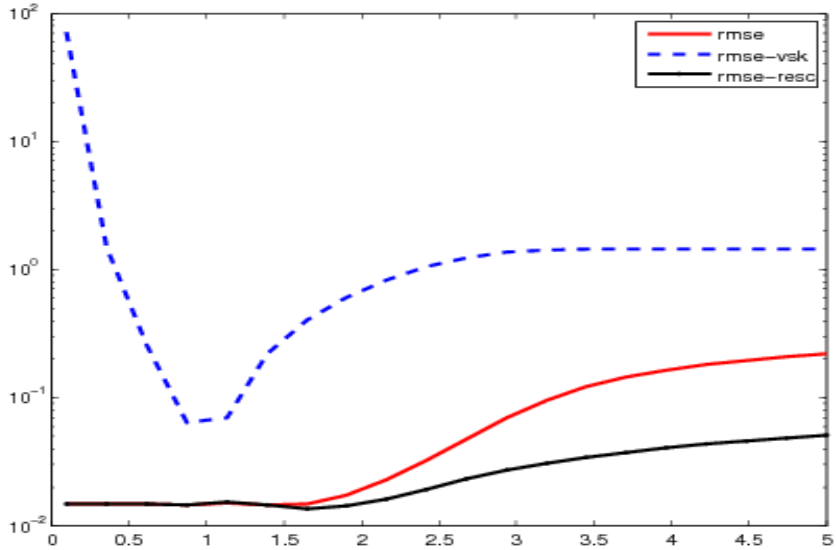


Figure: Classical, VSK and Rescaled method for W2 on varying the shape parameter

	Standard	+R	Cond(A)
Standard	$2.2e-01$	$5.1e-02$	$\approx 10^2$
+VS	$1.5e+00$	$9.6e-02$	$\approx 10^0$

Table: RMSE with/without rescaling applied to VSK

Comparison to accurate PUM

- In the recent work [CDeRP, DRNA16] a accurate PUM (A-PUM), combined with an optimal local RBF approximation via a *priori* error estimates (cf. DRWA15 poster) has been presented.
- The a method enables to select both suitable sizes of the different PU subdomains and shape parameters, i.e. the optimal couple (r_j^*, ϵ_j^*) for the subdomain Ω_j.
- the method uses a Bivariate LOOCV strategy.
- the method is suitable for data with non-homogeneous density.

#DataP	#EvalP	Method	RMSE	CPU time (sec)
289	1600	A-PUM	5.72e-4	3.44
		PU	4.34e-2	0.32
		RPUM	1.50e-2	0.32
1024	2500	A-PUM	1.32e-4	13.66
		PU	1.54e-2	0.63
		RPUM	7.55e-3	0.66
2500	6400	A-PUM	6.67e-5	32.42
		PU	6.14e-3	1.32
		RPUM	2.89e-3	1.30

Table: Comparison between A-PUM, PUM and RPUM with $\epsilon = 5$ with W2, on various grids on the square $[0,1]^2$, for interpolation of $f(x_1, x_2) = (x_1^2 + x_2^2 - 1)^9$.

Lebesgue functions

Domain: cardiod contained in $[-1,1]^2$. Data set: grid of 5×5 points.

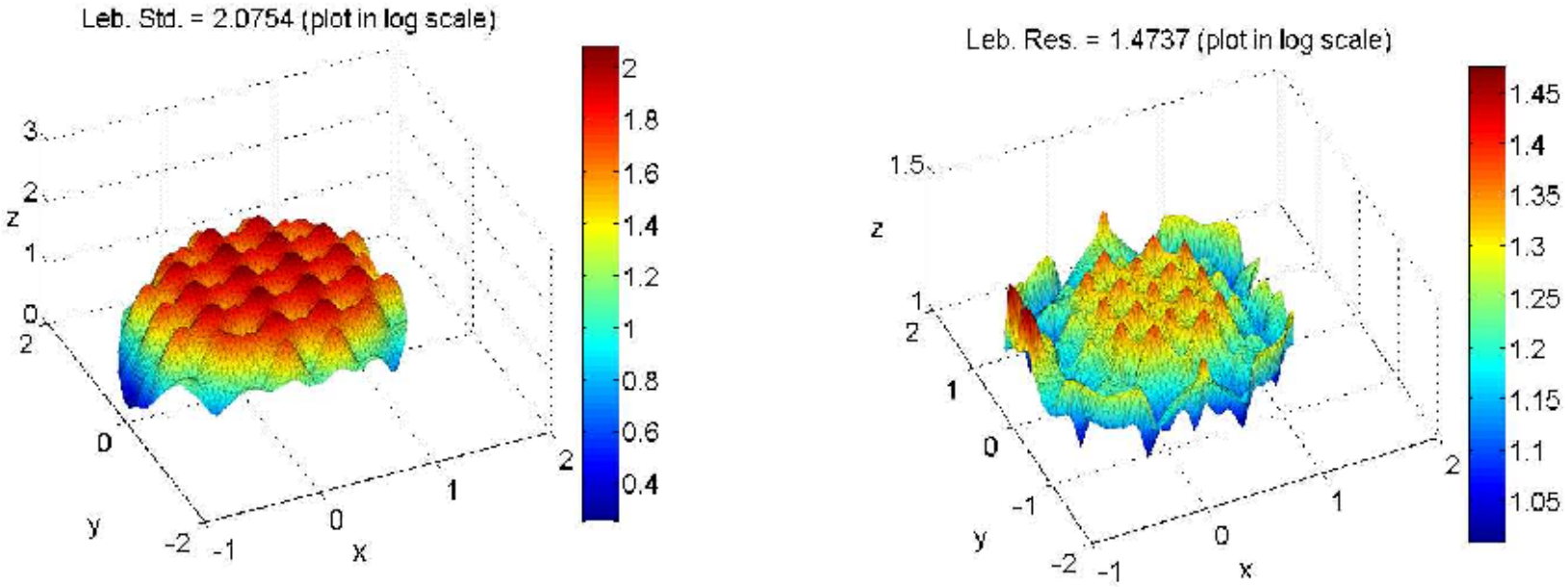


Figure: Comparison between the Lebesgue function with standard basis (Left) and the rescaled one (right) for the C^2 Wendland kernel on the cardiod with $\varepsilon = 3$.

Lebesgue functions (cont')

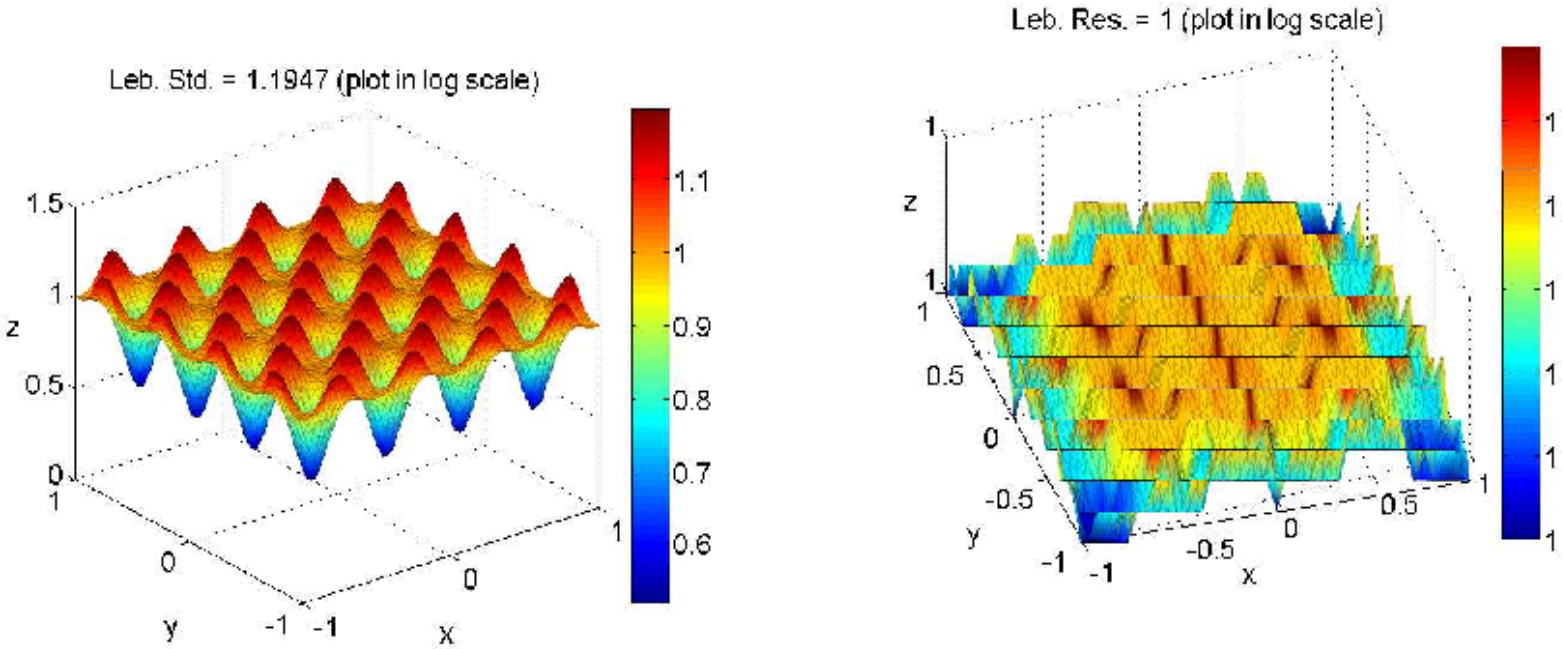


Figure: As in the previous slide with $\varepsilon = 3.85$.

Notice: for values of $\varepsilon \geq 2\varepsilon_M$ (in the example $\epsilon_M \approx 2$) the cardinal functions have disjoint supports.

Lebesgue constants

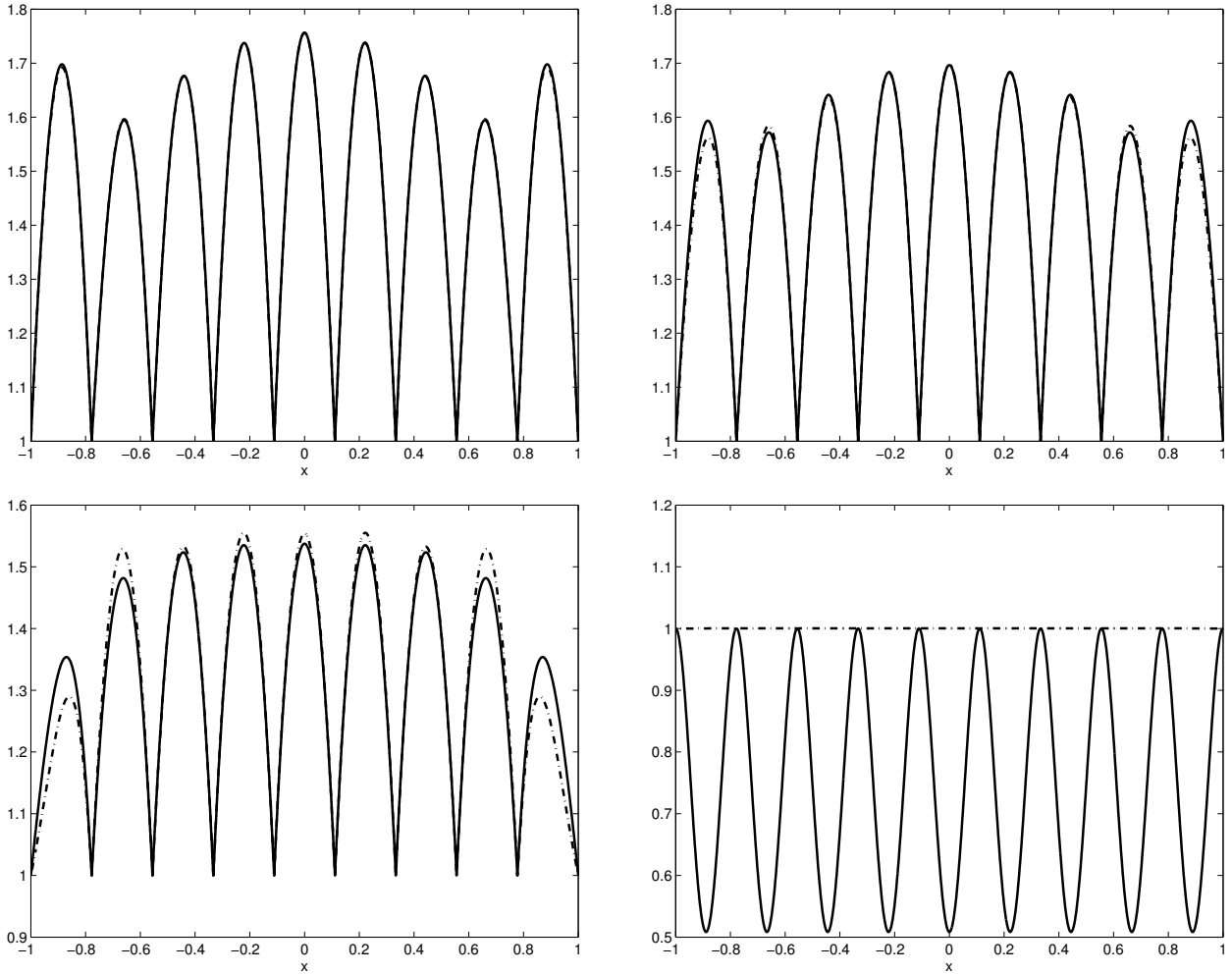


Figure: Comparison between the standard Lebesgue function (solid line) and the rescaled Lebesgue function (dotted line) for the C^2 Wendland kernel. From top left to bottom right, $\varepsilon = 0.5, 1, 2, 4$.

Generalized Rescaled Interpolant

$$K_r(\mathbf{x},\mathbf{y}) := s(\mathbf{x})s(\mathbf{y})K(\mathbf{x},\mathbf{y})$$

preserves the properties of the original Kernel.
Define s such that

$$s(\mathbf{x}) = s_\gamma(\mathbf{x}) := \frac{1}{P_g(\mathbf{x})^\gamma} = \frac{1}{(\sum_{j=1}^N c_j K(\mathbf{x},\mathbf{x}_j))^\gamma} \quad ,$$

γ is an "oscillation parameter".
The generalized rescaled kernel takes the form

$$K_{r,\gamma}(\mathbf{x},\mathbf{y}) := \frac{1}{(\sum_{j=1}^N c_j K(\mathbf{x},\mathbf{x}_j))^\gamma}\frac{1}{(\sum_{j=1}^N c_j K(\mathbf{y},\mathbf{x}_j))^\gamma}K(\mathbf{x},\mathbf{y}) =$$
$$= \frac{1}{\left(\sum_{j=1}^N c_j K(\mathbf{x},\mathbf{x}_j)\sum_{j=1}^N c_j K(\mathbf{y},\mathbf{x}_j)\right)^\gamma}K(\mathbf{x},\mathbf{y}) \quad .$$

Notice: $K_{r,0} = K$.

An example

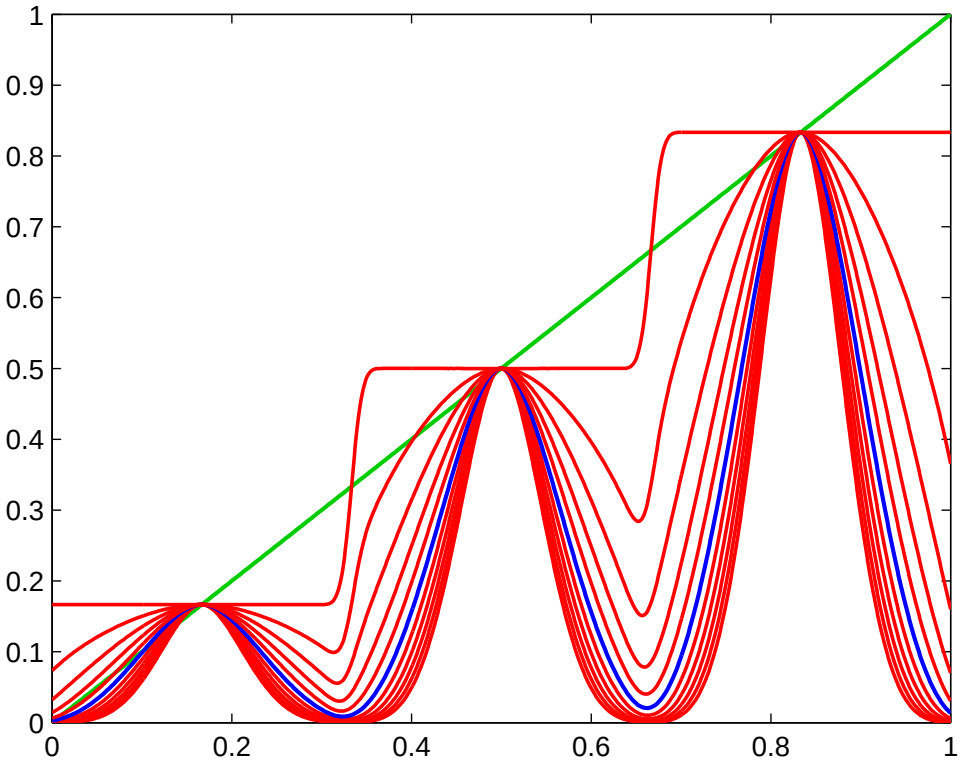


Figure: $X = \{\frac{1}{5}, \frac{1}{2}, \frac{5}{6}\}$, $y = x$ (green), Classic Interpolant (Blue), Generalized Interpolants (Red)

Curiosity

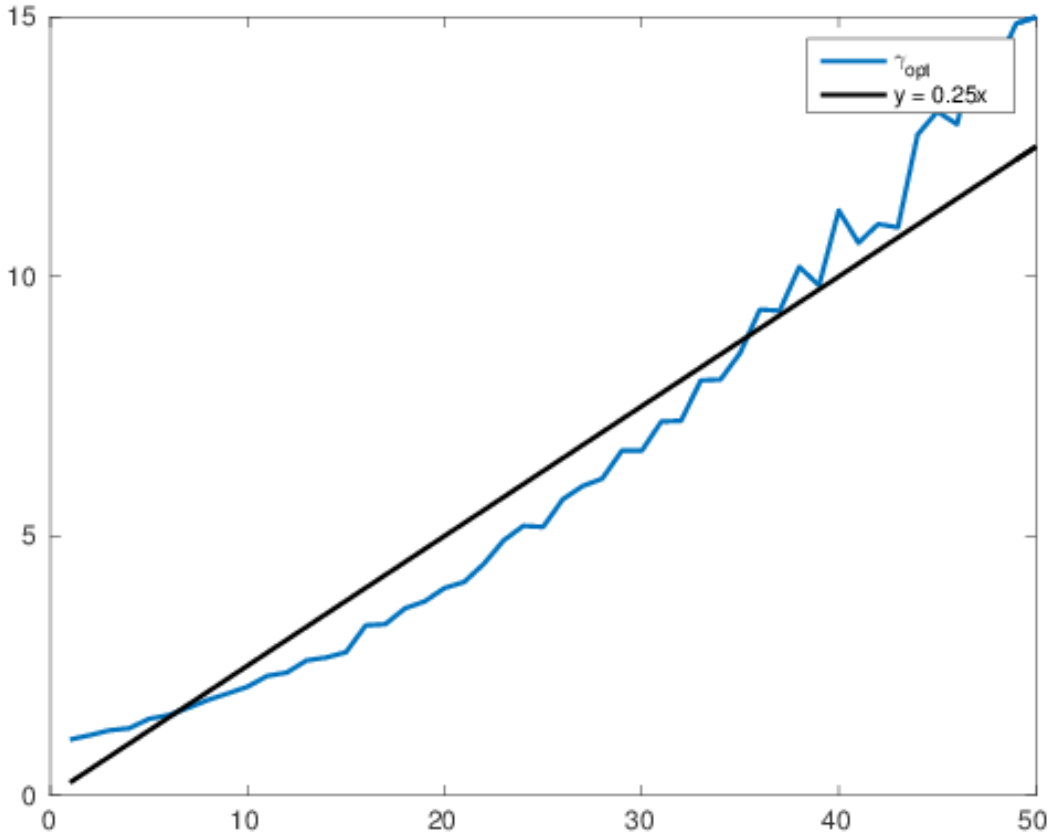


Figure: X=`linspace(-1,1,150)`, polynomial degree 1..50, $\epsilon = 3$, γ=`linspace(0,2*deg,100)`. Test repeated for any degree for 500 polynomials with random coefficients in $[-1, 1]$.

Summary

Done

- rescaled kernel and its properties
- rescaled kernel interpolation as a Shepard method
- rescaled kernel interpolant in cardinal form: Lebesgue constant behaviour
- application to PUM and SVK

To do

- error and stability analysis
- apply stable bases [DeMS13,15 and CDeMDeR+16] to RPUM?
- understanding the generalized kernels

References


Alessandra De Rossi, Emma Perracchione and Ezio Venturino: Fast strategy for PU interpolation: An application for the reconstruction of separatrix manifolds. Dolomites Res. Notes Approx.(2016), to appear.


Roberto Cavoretto, Alessandra De Rossi, Emma Perracchione: Fast partition of unity interpolation through block-based data structures. Poster session, Dolomites Research Week on Approximation, Canazei (2015).


Mira Bozzini, Licia Lenarduzzi, Milvia Rossini, Robert Shaback: Interpolation with Variably Scaled Kernels. IMA J. Numer. Anal. 35(1) (2015), pp. 199–219.


Oldenhuis Rody: Many test functions for global optimization algorithms. Matlab Central File Exchange (2015).


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), pp. A2745–A2762.


Gregory E. Fasshauer: *Meshfree Approximation Methods with Matlab*. World Scientific Publishing, Interdisciplinary Mathematical Sciences - Vol.6 (2007).


Holger Wendland: Fast evaluation of radial basis functions: methods based on partition of unity. Approximation theory, X, (St. Louis 2001) pp. 473–483, Innov. Appl. Math, Vanderbilt Univ. Press, Nashville, Tennessee (2002).


Nachman Aronszajn: Theory of Reproducing Kernels. Trans. Amer. Math. Soc., 68 (1950), pp. 337-404.


Andrea Idda: A comparison of some RBF interpolation method. Master's theis, University of Padova (2015)


Stefano De Marchi, Andrea Idda and Gabriele Santin: A rescaled method for RBF approximation. Draft (2016)

merci pour votre attention!

thanks for your attention!