Learning with Partition of Unity-based Kriging Estimators

R. Cavoretto[*,◇], A. De Rossi[*,◇], E. Perracchione[+,◇]

[*]*Dipartimento di Matematica "Giuseppe Peano", Università di Torino, Italy*
[+]*Dipartimento di Scienze Matematiche "Giuseppe Luigi Lagrange", Politecnico di Torino, Italy*
[◇]*Member of the INdAM Research group GNCS*

Abstract

For supervised regression tasks we propose and study a new tool, namely Kriging Estimator based on the Partition of Unity (KEPU) method. Its background belongs to the framework of local kernel-based interpolation methods. Indeed, even if the latter needs to be accurately tailored for Gaussian process regression, the KEPU scheme provides a global estimator which is constructed by gluing together the local Kriging predictors via compactly supported weights. The added value of this investigation is twofold. On the one hand, our theoretical studies about the propagation of the uncertainties from the local predictors to the global one offer the opportunity to define the PU method in a stochastic framework and hence to provide confidence intervals for the PU approximant. On the other hand, as confirmed by extensive numerical experiments, when the number of instances grows, such a method enables us to significantly reduce the usually high complexity cost of fitting via Gaussian processes.

Keywords: Kernel-based interpolation, Partition of Unity, Gaussian processes, uncertainty estimation
2010 MSC: 65D15, 41A05, 68Q32

1. Introduction

Recently, many sophisticated algorithms for regression models, as random forests [9] and neural networks [21], have been developed. They are able to learn very elaborated tasks. However, because of their complex architecture, they are not easy to work with in practice. Therefore, *kernel machines* (refer to [17, 43] for a general overview), as Support Vector Regression (SVR) and Kriging or Gaussian process regression, are still rather popular. The main advantage of

Email addresses: roberto.cavoretto@unito.it (R. Cavoretto[*,◇]), alessandra.derossi@unito.it (A. De Rossi[*,◇]), emma.perracchione@polito.it (E. Perracchione[+,◇])

the Kriging model, which has been introduced by D.G. Krige in 1951 [29] and later has become renowned in the field of geosciences [16], is that it is able to provide not only the prediction at a query data, but also a measure of the related uncertainty, known as the *Kriging variance*.

Without loosing generality, we might think of the Kriging method as a stochastic interpolation scheme, as well as regression models can be thought as interpolants of *smoothed* data. In view of this, it requires to solving a $n \times n$ linear system, where n denotes the number of measurements. Hence, when the number of examples grows, both the Kriging complexity costs and memory requirements (for the kernel matrix allocation) become prohibitive. To overcome such issues, which also arise in the context of Support Vector Machines (SVMs), several works deal with selecting (possibly randomly) a *representative* subset of measurements [31], as well as with approximating the usually full kernel matrix with a sparse one or via low-rank techniques [18, 27, 28, 32].

Moreover, most of recent research focuses on *local* learning algorithms (see e.g. [7] for a general overview) that construct several local models which are then combined together via some weights. In this direction Bayesian committee machines [48] provide local Kriging estimators and use weights depending on the inverse covariance of the predictions. A second class of local schemes that is worth to mention, as it shows similarities with the method proposed in this paper, is the one of clustering Kriging schemes [39, 40, 41, 49], which exhibit as well some analogies with localized procedures proposed for SVMs [33, 37, 44]. In [40] the predictors are pasted together using a distance metric, while in [41], after clustering data with $k-$means algorithms, the local Kriging predictors are combined together via weights selected so that the Kriging variance is minimized.

In this paper we look for an efficient computation of the Kriging Estimator (KE) whose background lies in the context of approximation theory. Precisely, the so-called Partition of Unity (PU) scheme, first introduced in the mid 1990s in [3], is nowadays a well-established method for approximating large data sets and it is also rather popular for researchers working on collocation schemes or numerical solution of Partial Differential Equations (PDEs); refer e.g. to [2, 13, 30, 38]. We then make use of such a partitioning scheme to compute both the Kriging predictions and uncertainties. The resulting method, namely KEPU, drastically reduces the computation complexity of global Gaussian processes, as numerically shown, and inherits properties from the local Kriging estimators. Indeed, we prove that it is unbiased and that the Kriging uncertainty of the KEPU is a squared weighted sum of the local Mean Squared Errors (MSEs). Hence, as a benefit of this study we are able to include the PU method in a machine learning and stochastic framework, providing predictions at query data and related Kriging variances.

The paper is organized as follows. In Section 2 we briefly review the kernel-based PU method for interpolating scattered data, i.e. from a deterministic point of view. Section 3 presents the theoretical study about the proposed localized Gaussian process, whose complexity analysis and computational details are studied in Section 4. Numerical experiments are presented in Section 5, while conclusions are offered in Section 6.

2. Preliminaries

In this section we present the kernel-based PU method, and in doing so we focus on interpolation; any extension to regression schemes is straightforward and will be commented later.

2.1. Kernel-based interpolation

We consider a function $f : \Omega \longrightarrow \mathbb{R}$ with $\Omega \subset \mathbb{R}^d$, i.e. a function depending on d features, and an associated set of function values $\mathcal{F} = \{f(\boldsymbol{x_i})\}_{i=1}^n$ sampled at a data point set $\mathcal{X}_n = \{\boldsymbol{x}_i\}_{i=1}^n \subset \Omega$. Given the examples $\{(\boldsymbol{x}_i, f(\boldsymbol{x}_i))\}_{i=1}^n$, our goal is to construct an approximation of the unknown function f, namely $\tilde{f}$. Thus, in order to construct an interpolant we have to impose n interpolation constraints, i.e.

$$\tilde{f}(\boldsymbol{x}_i) = f(\boldsymbol{x_i}), \quad i = 1, \ldots, n. \tag{1}$$

Then, given a normed linear space of functions defined on Ω, and an associated basis $\{b_j\}_{j=1}^n \subset C(\Omega)$, an interpolant $\tilde{f} : \Omega \longrightarrow \mathbb{R}$ may be defined as

$$\tilde{f}(\boldsymbol{x}) = \sum_{i=1}^n \alpha_i b_i(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega, \tag{2}$$

where $\alpha_1, \ldots, \alpha_n$ need to be determined by imposing the interpolation conditions (1). Such an interpolant is unique as long as $\{b_i\}_{i=1}^n$ forms a Haar system. For $d > 1$ this condition holds true only for trivial Haar spaces, i.e. spaces spanned by a single function (see e.g. [51, Theorem 2.3, p. 19]). Nevertheless, if we consider data-dependent basis, as for kernel-based interpolation, the existence and uniqueness of the interpolant might be ensured. Precisely, let $H_\kappa(\Omega)$ be a Hilbert space equipped with an inner product $(\cdot,\cdot)_{H_\kappa(\Omega)}$, we consider symmetric reproducing kernels $\kappa : \Omega \times \Omega \longrightarrow \mathbb{R}$ for $H_\kappa(\Omega)$, i.e. so that (refer e.g. to [23, Definition 2.6, p. 32]) $\kappa(\cdot, \boldsymbol{x}) \in H_\kappa(\Omega)$, $\kappa(\boldsymbol{x}, \boldsymbol{z}) = \kappa(\boldsymbol{z}, \boldsymbol{x})$ for all $\boldsymbol{x}, \boldsymbol{z} \in \Omega$, and

$$(f, \kappa(\cdot, \boldsymbol{x}))_{H_\kappa(\Omega)} = f(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega.$$

As a consequence, we have that each reproducing kernel is identified by an inner product, i.e.

$$(\kappa(\cdot, \boldsymbol{x}), \kappa(\cdot, \boldsymbol{z}))_{H_\kappa(\Omega)} = \kappa(\boldsymbol{x}, \boldsymbol{z}), \quad \boldsymbol{x}, \boldsymbol{z} \in \Omega. \tag{3}$$

Equivalently, $\kappa : \Omega \times \Omega \longrightarrow \mathbb{R}$ is a reproducing kernel if there exists a mapping $\Phi : \Omega \longrightarrow H_\kappa(\Omega)$, usually referred to as *feature map* [46, §5], so that:

$$\kappa(\boldsymbol{x}, \boldsymbol{z}) = (\Phi_{\boldsymbol{x}}, \Phi_{\boldsymbol{z}})_{H_\kappa(\Omega)}, \quad \boldsymbol{x}, \boldsymbol{z} \in \Omega. \tag{4}$$

In what follows we focus on radial kernels; refer to [22] for a general overview. They are kernels for whom there exists a Radial Basis Function (RBF) $\varphi : \mathbb{R}_+ \longrightarrow \mathbb{R}$, where $\mathbb{R}_+ = [0, \infty)$, and (possibly) two parameters $\ell > 0$ and

$\sigma > 0$ (known in machine learning literature as *length scale* and *process variance*, respectively) such that, for all $\boldsymbol{x}, \boldsymbol{z} \in \Omega$,

$$\kappa(\boldsymbol{x}, \boldsymbol{z}) = \sigma^2 \kappa_\ell(\boldsymbol{x}, \boldsymbol{z}) = \sigma^2 \varphi_\ell(||\boldsymbol{x} - \boldsymbol{z}||_2) = \sigma^2 \varphi(r), \tag{5}$$

where $r = ||\boldsymbol{x} - \boldsymbol{z}||_2$. Scaling the kernel with σ^2 will not change the interpolation setting, but here, we need to incorporate it into the kernel for defining later the *Kriging variance*.

With these preliminaries, from the expansion (2) we may derive the following representation

$$\tilde{f}(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \kappa(\boldsymbol{x}, \boldsymbol{x}_i), \quad \boldsymbol{x} \in \Omega,$$

where $\kappa : \Omega \times \Omega \longrightarrow \mathbb{R}$ is a positive definite radial kernel. Indeed for those kernels the solution of the interpolation problem is unique (refer e.g. to [23, Definition 2.2, p. 18]). Precisely, letting $\boldsymbol{f} = (f(\boldsymbol{x}_1), \dots, f(\boldsymbol{x}_n))^\intercal$ and $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n)^\intercal$, the scattered data interpolation problem reduces to solving the linear system of the form:

$$\mathsf{K}\boldsymbol{\alpha} = \boldsymbol{f}, \tag{6}$$

where the matrix K with entries $\mathsf{K}_{ik} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_k)$ is non-singular.

2.2. Partition of unity method

Since the interpolation matrix in (6) is typically full, such a meshfree approach works efficiently as long as we have a reduced number of data. Conversely, when the number of examples grows, local methods, as the PU ones, might be helpful. We thus consider a partition of the open and bounded domain Ω into m subdomains Ω_j, such that $\Omega \subseteq \cup_{j=1}^{m} \Omega_j$, with some mild overlap among them [52]. Associated with this partition, we take a family of compactly supported, non-negative, continuous functions w_j, $j = 1, \dots, m$, which form a partition of unity, i.e.

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

One possible solution is to consider the so-called Shepard's weights [45], which are defined as

$$w_j(\boldsymbol{x}) = \frac{\bar{w}_j(\boldsymbol{x})}{\displaystyle\sum_{k=1}^{m} \bar{w}_k(\boldsymbol{x})}, \quad j = 1, \dots, m, \quad \boldsymbol{x} \in \Omega,$$

where $\bar{w}_j$ are compactly supported functions with support on Ω_j, as Wendland's functions [53].

With these ingredients, we can define the PU interpolant as

$$\bar{f}(\boldsymbol{x}) = \sum_{j=1}^{m} \tilde{f}_j(\boldsymbol{x}) \, w_j(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega,$$

where $\tilde{f}_j$ denotes a kernel-based approximant defined on a subdomain Ω_j of the form

$$\tilde{f}_j(\boldsymbol{x}) = \sum_{k=1}^{n_j} \alpha_k^j \kappa(\boldsymbol{x}, \boldsymbol{x}_k^j), \quad \boldsymbol{x} \in \Omega,$$

being n_j the number of data points belonging to Ω_j and $\boldsymbol{x}_k^j \in \mathcal{X}_j = \mathcal{X}_n \cap \Omega_j$, with $k = 1, \dots, n_j$.

Then, the construction of the PU interpolant consists in solving m (invertible) linear systems of the form:

$$\mathsf{K}_j \boldsymbol{\alpha}_j = \boldsymbol{f}_j, \tag{7}$$

where $\boldsymbol{\alpha}_j = (\alpha_1^j, \dots, \alpha_{n_j}^j)^\intercal$, $\boldsymbol{f}_j = (f(\boldsymbol{x}_1^j), \dots, f(\boldsymbol{x}_{n_j}^j))^\intercal$ and K_j is the local interpolation matrix whose entries are given by

$$(\mathsf{K}_j)_{ik} = \kappa(\boldsymbol{x}_i^j, \boldsymbol{x}_k^j), \quad i, k = 1, \dots, n_j.$$

Note that, if we formally solve the above system, we get $\boldsymbol{\alpha}_j = \mathsf{K}_j^{-1} \boldsymbol{f}_j$ and thus the *nodal* values are given by

$$\bar{f}(\boldsymbol{x}) = \sum_{j=1}^{m} \tilde{f}_j(\boldsymbol{x}) w_j(\boldsymbol{x}) = \sum_{j=1}^{m} \boldsymbol{k}_j(\boldsymbol{x})^\intercal \mathsf{K}_j^{-1} \boldsymbol{f}_j w_j(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega, \tag{8}$$

where $\boldsymbol{k}_j(\boldsymbol{x}) = (\kappa(\boldsymbol{x}, \boldsymbol{x}_1^j), \dots, \kappa(\boldsymbol{x}, \boldsymbol{x}_{n_j}^j))$.

For the PU scheme, only deterministic error bounds (see [51, Theorem 5]) are available and there are no studies on the uncertainty associated to the PU approximation. This might limit the popularity of such method in other settings, as in machine learning and statistics literature.

In the remaining part of this work, therefore, we will investigate how the PU method can be efficiently built for Gaussian process or *simple* Kriging. To introduce and study in the next section the Kriging predictors in a local context, we will mainly follow the exposition line provided in [23, §5].

3. Kriging estimator based on partition of unity

For a fixed $\boldsymbol{x} \in \Omega$, the main requirement in the Kriging prediction is the one of assuming that the value $f(\boldsymbol{x})$ is a realisation of a random variable $F_{\boldsymbol{x}}$ belonging to a zero-mean Gaussian random field F. We first note that, letting $H_F(\Omega)$ be the Hilbert space generated by F, the following representation holds true (see e.g. [4, §2])

$$(F_{\boldsymbol{x}}, F_{\boldsymbol{z}})_{H_F(\Omega)} = \mathbb{E}\left(F_{\boldsymbol{x}}, F_{\boldsymbol{z}}\right) = \kappa(\boldsymbol{x}, \boldsymbol{z}) = \left(\kappa(\cdot, \boldsymbol{x}), \kappa(\cdot, \boldsymbol{z})\right)_{H_\kappa(\Omega)}, \quad \boldsymbol{x}, \boldsymbol{z} \in \Omega.$$

The above equation, together with (3) and (4), shows the analogies between the deterministic, the machine learning and the stochastic point of view; we also refer the reader to [10, 25, 26, 34].

3.1. Localized Gaussian fitting

When many examples are given, the main drawback of the classical, i.e. global, Kriging prediction is the allocation of the kernel matrix K as in (6) and the solution of the associated linear system. Therefore, thanks to the PU method, we define Gaussian processes that are weighted sums of local Kriging estimators.

For the localized approach we think of $f(\boldsymbol{x}_i^j)$, $i = 1, \dots, n$, as realisations of random variables $F_{\boldsymbol{x}_i^j}$, $i = 1, \dots, n_j$, with the property that for any given distinct data point set $\mathcal{X}_j = \{\boldsymbol{x}_1^j, \dots, \boldsymbol{x}_{n_j}^j\} \subset \Omega_j$, $\boldsymbol{F}_j = (F_{\boldsymbol{x}_1^j}, \dots, F_{\boldsymbol{x}_{n_j}^j})^\intercal$ has a multivariate normal distribution with mean vector $\boldsymbol{\mu}_j = \mathbb{E}(\boldsymbol{F}_j)$ and covariance matrix K_j. For clarity in the exposition and without loosing generality, we now fix $\boldsymbol{\mu}_j = \boldsymbol{0}$, $j = 1, \dots, m$. With such assumptions, we define the localized Kriging *predictor* as

$$\bar{F}_{\boldsymbol{x}} = \sum_{j=1}^{m} \tilde{F}_{\boldsymbol{x}}^j w_j(\boldsymbol{x}) = \sum_{j=1}^{m} \left(\boldsymbol{k}_j(\boldsymbol{x})^\intercal \mathsf{K}_j^{-1} \boldsymbol{F}_j\right) w_j(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega, \tag{9}$$

whose realizations provide us the Kriging *predictions* as in (8). To study such a localized Gaussian process, we introduce the following random variables:

$$Y_{\boldsymbol{x}}^j = (F_{\boldsymbol{x}} | \boldsymbol{F}_j = \boldsymbol{f}_j), \quad j = 1, \dots, m,$$

whose distributions are given by (see e.g. [23, Equation (5.18), p. 103]):

$$Y_{\boldsymbol{x}}^j \sim \mathcal{N}\left(\boldsymbol{k}_j(\boldsymbol{x})^\intercal \mathsf{K}_j^{-1} \boldsymbol{f}_j, \kappa(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}_j(\boldsymbol{x})^\intercal \mathsf{K}_j^{-1} \boldsymbol{k}_j(\boldsymbol{x})\right). \tag{10}$$

Given such variables, we are able to characterize the KEPU as follows.

Proposition 3.1 *For a given $\boldsymbol{x} \in \Omega$, letting*

$$Y_{\boldsymbol{x}} = \sum_{j=1}^{m} Y_{\boldsymbol{x}}^j w_j(\boldsymbol{x}) = \sum_{j=1}^{m} (F_{\boldsymbol{x}} | \boldsymbol{F}_j = \boldsymbol{f}_j)\, w_j(\boldsymbol{x}), \tag{11}$$

the KEPU defined in (9) *is the expected value of $Y_{\boldsymbol{x}}$, i.e.*

$$\mathbb{E}(Y_{\boldsymbol{x}}) = \bar{F}_{\boldsymbol{x}}.$$

Proof: Because of the linearity of the expectation, given $\boldsymbol{x} \in \Omega$, we have that that:

$$\begin{aligned} \mathbb{E}(Y_{\boldsymbol{x}}) &= \mathbb{E}\left(\sum_{j=1}^{m} (F_{\boldsymbol{x}} | \boldsymbol{F}_j = \boldsymbol{f}_j)\, w_j(\boldsymbol{x})\right) = \sum_{j=1}^{m} \mathbb{E}\left(F_{\boldsymbol{x}} | \boldsymbol{F}_j = \boldsymbol{f}_j\right) w_j(\boldsymbol{x}) \\ &= \sum_{j=1}^{m} (\boldsymbol{k}_j(\boldsymbol{x})^\intercal \mathsf{K}_j^{-1} \boldsymbol{f}_j) w_j(\boldsymbol{x}) = \sum_{j=1}^{m} \tilde{F}_{\boldsymbol{x}}^j w_j(\boldsymbol{x}). \end{aligned}$$

Then, Equation (9) concludes the proof. ∎

The global PU predictor is then a weighted sum of m local Kriging estimators of $F_{\boldsymbol{x}}$. A similar idea is presented in [41], indeed the authors use weights that form a partition of unity and they are selected so that the Kriging variance is minimized.

We further note that each of the local estimators is unbiased, i.e., $\mathbb{E}(F_{\boldsymbol{x}}) = \mathbb{E}(F_{\boldsymbol{x}}^j)$, $j = 1, \ldots, m$, and such a property is inherited by the PU scheme, as shown in the following result.

Proposition 3.2 *Let $\boldsymbol{x} \in \Omega$, the* KEPU $\bar{F}_{\boldsymbol{x}}$, *defined in* (9)*, is unbiased.*

Proof: We have that:

$$\mathbb{E}(\bar{F}_{\boldsymbol{x}}) = \mathbb{E}\left(\sum_{j=1}^{m} \tilde{F}_{\boldsymbol{x}}^j w_j(\boldsymbol{x})\right) = \sum_{j=1}^{m} \mathbb{E}(\tilde{F}_{\boldsymbol{x}}^j) w_j(\boldsymbol{x}) = \sum_{j=1}^{m} \mathbb{E}(F_{\boldsymbol{x}}) w_j(\boldsymbol{x}).$$

Then, since for $\boldsymbol{x} \in \Omega$, $\sum_{j=1}^{m} w_j(\boldsymbol{x}) = 1$, the thesis follows. Indeed,

$$\mathbb{E}(\bar{F}_{\boldsymbol{x}}) = \sum_{j=1}^{m} \mathbb{E}(F_{\boldsymbol{x}}) w_j(\boldsymbol{x}) = \mathbb{E}(F_{\boldsymbol{x}}) \sum_{j=1}^{m} w_j(\boldsymbol{x}) = \mathbb{E}(F_{\boldsymbol{x}}).$$

■

Note that, until this moment, we just recovered the classical PU interpolant, seen in a stochastic setting. Now, we introduce the associated Kriging variance that represents the main difference between the stochastic and the deterministic point of view.

3.2. Localized Gaussian uncertainties

We then have to investigate how the local uncertainties propagate towards the global KEPU. To this aim, we assume that the variables $Y_{\boldsymbol{x}}^j$, $j = 1, \ldots, m$, are uncorrelated. This is not so restrictive, because all the local Kriging predictors are constructed independently of each other. Then, as a consequence of Proposition 3.1, we have the following result which makes use of the so-called *power function*; see [51, Definition 11.2, p. 174] and [20].

Corollary 3.2.1 *If* $\mathrm{Cov}(Y_{\boldsymbol{x}}^j, Y_{\boldsymbol{x}}^k) = 0$*, for $j \neq k$, $j, k = 1, \ldots, m$, then*

$$Y_{\boldsymbol{x}} \sim \mathcal{N}\left(\bar{F}_{\boldsymbol{x}}, \sum_{j=1}^{m} \mathcal{P}_j^2(\boldsymbol{x}) w_j^2(\boldsymbol{x})\right),$$

where $Y_{\boldsymbol{x}}$ is defined as in (11) *and*

$$\mathcal{P}_j(\boldsymbol{x}) = \sqrt{\kappa(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}_j(\boldsymbol{x})^{\intercal} \mathsf{K}_j^{-1} \boldsymbol{k}_j(\boldsymbol{x})}, \quad \boldsymbol{x} \in \Omega,$$

is the power function (computed on Ω_j).

Proof: Given $\boldsymbol{x} \in \Omega$, since for each subdomain Equation (10) holds true and because the random variables $Y_{\boldsymbol{x}}^j$, $j = 1, \dots, m$, are uncorrelated their sum follows a normal distribution whose mean is provided by Proposition 3.1 and whose variance is given by

$$\mathrm{Var}(Y_{\boldsymbol{x}}) = \sum_{j=1}^{m} \left(\kappa(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}_j(\boldsymbol{x})^\intercal \mathsf{K}_j^{-1} \boldsymbol{k}_j(\boldsymbol{x})\right) w_j^2(\boldsymbol{x}) = \sum_{j=1}^{m} \mathcal{P}_j^2(\boldsymbol{x}) w_j^2(\boldsymbol{x}).$$

■

For each $\boldsymbol{x} \in \Omega$, being $\bar{F}_{\boldsymbol{x}}$ an estimator of $F_{\boldsymbol{x}}$, we now have to compute its MSE. To reach such scope, we refer the reader to [23, p. 97] and we assume that $\mathrm{Cov}(F_{\boldsymbol{x}} - \tilde{F}_{\boldsymbol{x}}^j, F_{\boldsymbol{x}} - \tilde{F}_{\boldsymbol{x}}^k) = 0$ for $j \neq k$ and $j, k = 1, \dots, m$. Again, being the local predictors constructed independently of each other, this requirement is not too demanding.

Proposition 3.3 *For a given $\boldsymbol{x} \in \Omega$, if* $\mathrm{Cov}((F_{\boldsymbol{x}} - \tilde{F}_{\boldsymbol{x}}^j), (F_{\boldsymbol{x}} - \tilde{F}_{\boldsymbol{x}}^k)) = 0$ *for $j \neq k$ and $j, k = 1, \dots, m$, the* MSE *of the KEPU is so that*

$$\mathrm{MSE}(\bar{F}_{\boldsymbol{x}}) = \sum_{j=1}^{m} \left(\kappa(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}_j(\boldsymbol{x})^\intercal \mathsf{K}_j^{-1} \boldsymbol{k}_j(\boldsymbol{x})\right) w_j^2(\boldsymbol{x}).$$

Proof: Since $\{w_j\}_{j=1}^m$ form a partition of unity, for $\boldsymbol{x} \in \Omega$, we note that

$$\begin{aligned}
\mathrm{MSE}(\bar{F}_{\boldsymbol{x}}) &= \mathbb{E}\left((F_{\boldsymbol{x}} - \bar{F}_{\boldsymbol{x}})^2\right) \\
&= \mathbb{E}\left(\left(F_{\boldsymbol{x}} \sum_{j=1}^{m} w_j(\boldsymbol{x}) - \sum_{j=1}^{m} \tilde{F}_{\boldsymbol{x}}^j w_j(\boldsymbol{x})\right)^2\right) \\
&= \mathbb{E}\left(\left(\sum_{j=1}^{m} \left(F_{\boldsymbol{x}} - \tilde{F}_{\boldsymbol{x}}^j\right) w_j(\boldsymbol{x})\right)^2\right) \\
&= \mathbb{E}\left(\sum_{j=1}^{m} \left(F_{\boldsymbol{x}} - \tilde{F}_{\boldsymbol{x}}^j\right)^2 w_j^2(\boldsymbol{x})\right) + \\
&+ 2\mathbb{E}\left(\sum_{j<k} \left(F_{\boldsymbol{x}} - \tilde{F}_{\boldsymbol{x}}^j\right)\left(F_{\boldsymbol{x}} - \tilde{F}_{\boldsymbol{x}}^k\right) w_j(\boldsymbol{x}) w_k(\boldsymbol{x})\right).
\end{aligned}$$

Then, thanks to the local properties of the Kriging predictor we observe that:

$$\mathbb{E}\left(\sum_{j=1}^{m}\left(F_{\boldsymbol{x}}-\tilde{F}_{\boldsymbol{x}}^{j}\right)^{2} w_{j}^{2}(\boldsymbol{x})\right)=\sum_{j=1}^{m}\mathbb{E}\left(\left(F_{\boldsymbol{x}}-\tilde{F}_{\boldsymbol{x}}^{j}\right)^{2}\right) w_{j}^{2}(\boldsymbol{x})$$
$$=\sum_{j=1}^{m}\left(\kappa(\boldsymbol{x},\boldsymbol{x})-\boldsymbol{k}_{j}(\boldsymbol{x})^{\intercal}\mathsf{K}_{j}^{-1}\boldsymbol{k}_{j}(\boldsymbol{x})\right) w_{j}^{2}(\boldsymbol{x}).$$

For the second term, being $\mathrm{Cov}(F_{\boldsymbol{x}}-\tilde{F}_{\boldsymbol{x}}^{j}, F_{\boldsymbol{x}}-\tilde{F}_{\boldsymbol{x}}^{k})=0$, for $j\neq k$ and $j,k=1,\dots,m$, we have that

$$\mathbb{E}\left(\sum_{j<k}\left(F_{\boldsymbol{x}}-\tilde{F}_{\boldsymbol{x}}^{j}\right)\left(F_{\boldsymbol{x}}-\tilde{F}_{\boldsymbol{x}}^{k}\right) w_{j}(\boldsymbol{x})w_{k}(\boldsymbol{x})\right)$$
$$=\sum_{j<k}\mathbb{E}\left(\left(F_{\boldsymbol{x}}-\tilde{F}_{\boldsymbol{x}}^{j}\right)\left(F_{\boldsymbol{x}}-\tilde{F}_{\boldsymbol{x}}^{k}\right)\right) w_{j}(\boldsymbol{x})w_{k}(\boldsymbol{x})$$
$$=\sum_{j<k}\mathbb{E}\left(F_{\boldsymbol{x}}-\tilde{F}_{\boldsymbol{x}}^{j}\right)\mathbb{E}\left(F_{\boldsymbol{x}}-\tilde{F}_{\boldsymbol{x}}^{k}\right) w_{j}(\boldsymbol{x})w_{k}(\boldsymbol{x})=0,$$

where the last equality follows from the fact that all the local estimators are unbiased. ∎

Being the localized Kriging predictor unbiased (see Proposition 3.2), Proposition 3.3 tells us how to compute the variance of $\bar{F}$, i.e. $\mathrm{Var}(\bar{F}_{\boldsymbol{x}})=\mathrm{MSE}(\bar{F}_{\boldsymbol{x}})$. Then, we are able to introduce confidence intervals. Precisely, given $\delta\in[0,1]$, we obtain

$$P\left(F_{\boldsymbol{x}}\in\bar{F}_{\boldsymbol{x}}\pm z_{\delta}\sqrt{\sum_{j=1}^{m}\mathcal{P}_{j}^{2}(\boldsymbol{x})w_{j}^{2}(\boldsymbol{x})}\right)=1-\delta, \tag{12}$$

where z_{δ} is the quantile location of the normal distribution. Note that, the above equation allows us to understand how the local uncertainties propagate via the PU scheme. We conclude this section with a few remarks.

Remark 3.1 (Zero-mean) *In our presentation we have supposed to deal with zero-mean Gaussian random fields. Such hypothesis might appear restrictive. However, with some pre-processing on the data one can always consider the proposed simple Kriging approach. As an alternative, for $\boldsymbol{x}\in\Omega$, letting $\boldsymbol{\mu}_j$ the local mean on Ω_j, one can define the local predictors as (see e.g. [23, Remark 5.5, p. 101])*

$$\tilde{F}_{\boldsymbol{x}}^{j}=\boldsymbol{\mu}_{j}+\boldsymbol{k}_{j}(\boldsymbol{x})^{\intercal}\mathsf{K}_{j}^{-1}(\boldsymbol{F}_{j}-\boldsymbol{\mu}_{j}),\quad j=1,\dots,m,\quad \boldsymbol{x}\in\Omega.$$

Remark 3.2 (Noise) *Kriging predictions are frequently associated to noisy data. In this study we deliberately focused (without any restrictions) only on* *interpolation. Precisely, tools known as* smooting splines, ridge regression *and* Tikhonov regularization, *which are typically used for Kriging regression, are based on solving for each subdomain [24, 47, 50]*

$$(\mathsf{K}_j + \lambda \mathsf{I})\boldsymbol{\alpha}_j = \boldsymbol{f}_j, \quad j = 1, \ldots, m,$$

instead of (7)*, where* $\lambda \in \mathbb{R}_+$ *and* I *is the* $n_j \times n_j$ *identity matrix. Nevertheless, this is equivalent to interpolating the* smoothed *data* $\hat{\boldsymbol{f}}_j = (\mathsf{K}_j + \lambda \mathsf{I})^{-1}\mathsf{K}_j \boldsymbol{f}_j$ *with the method previously described. Indeed, given* $\boldsymbol{x} \in \Omega$*, and by applying the push-through the identity [5, Fact 2.16.16] for matrix inverses we can define the local estimators as*

$$\tilde{f}_j(\boldsymbol{x}) = \kappa(\boldsymbol{x})^\intercal (\mathsf{K}_j + \lambda \mathsf{I})^{-1}\boldsymbol{f}_j = \kappa(\boldsymbol{x})^\intercal \mathsf{K}_j^{-1}(\mathsf{K}_j + \lambda \mathsf{I})^{-1}\mathsf{K}_j \boldsymbol{f}_j = \kappa(\boldsymbol{x})^\intercal \mathsf{K}_j^{-1}\hat{\boldsymbol{f}}_j,$$

with $j = 1, \ldots, m$.

4. Complexity analysis and implementation

In the following we point out some computational details and we briefly analyze the complexity of the proposed KEPU method.

4.1. Complexity costs

For the proposed localized algorithm, we use Wendland's C^2 functions as PU weights and balls in $\mathbb{R}^d$ as patches whose radii are constant and fixed as ρ/m, with $\rho = \sqrt{2}$. Note that, if the subdomains centres are grid data then, to ensure that they form a covering of Ω, any $\rho \geq 1$ can be used. Once the PU structure is set, the first step of the proposed localized Kriging method consists in distributing the scattered data among the different subdomains. To this end, we consider the well-established data structure and sorting routine introduced in [11], further developed in [1, 14] for Shepard-type methods and used for other kernel bases in [19]. They respectively require

$$\mathcal{O}\left(n \log n + \sum_{k=1}^{d-1} \frac{d-k}{d} n \log n\right),$$

and $\mathcal{O}(n)$ operations. Once this issue is accomplished, the problem reduces to solving m linear systems whose size is $n_j \times n_j$. Since usually $n_j \ll n$ this leads to a saving in terms of computational times with respect to the classical Kriging implementation. Of course, since the localized Kriging prediction involves a pre-processing step for organizing the instances among the subdomains, we expect improvements in terms of computational complexity when n is *sufficiently large.*

4.2. Implementation details

For the KEPU implementation, we have to solve (7) and then compute the uncertainty via Corollary 3.2.1. This is compared in terms of efficiency and accuracy with the Global Kriging Estimator (GKE). Aside this kind of implementation, called in what follows *canonical*, we will make use of the MATLAB function `fitrgp.m` that belongs to the Statistics and Machine Learning toolbox [35]. Indeed, such a routine already offers an ad hoc implementation for huge values of n. Precisely, when $n > 10000$, it does not allocate the kernel matrix as in (6) (which, being a $n \times n$ matrix whose entries are in double-precision floating-point format, would require $8 \times n \times n$ bytes), but it takes advantage of a block coordinate descent algorithm. Nevertheless, such a strategy is not completely satisfying. Indeed, when $n > 10000$, `fitrgp.m` does not return the Kriging variance, which plays an important role in the stochastic predictions. On the opposite, with our KEPU, we are able to allocate the local matrices and hence compute the Kriging uncertainties also for large data sets (using the `fitrgp.m` on each subdomain). This is a consequence of the fact that the largest matrix that the algorithm needs to store has size $n_s \times n_s$, where $n_s = \max_{j=1,\ldots,m} \mathrm{card}(\Omega_j)$ and it is so that $n_s \ll n$.

As far as the kernels are concerned, we consider the Gaussian (GA) function, which is also known as Squared Exponential, and the family of Matérn functions [36]. The former is defined as

$$\kappa_\ell^{\mathrm{GA}}(\boldsymbol{x},\boldsymbol{z}) = \exp\left(-\frac{1}{2\ell^2}||\boldsymbol{x}-\boldsymbol{z}||_2^2\right), \quad \boldsymbol{x},\boldsymbol{z} \in \Omega,$$

and it is infinitely smooth. The Matérn kernels are instead characterized by a finite regularity. Indeed, for $\boldsymbol{x},\boldsymbol{z} \in \Omega$, such functions are given by

$$\kappa_\ell^{\mathrm{M}}(\boldsymbol{x},\boldsymbol{z}) = \frac{1}{\Gamma(\nu)2^{\nu-1}}\left(\frac{\sqrt{2\nu}}{\ell}||\boldsymbol{x}-\boldsymbol{z}||_2\right)^{\nu} B_\nu\left(\frac{\sqrt{2\nu}}{\ell}||\boldsymbol{x}-\boldsymbol{z}||_2\right), \tag{13}$$

where B_ν is a modified Bessel function and Γ is the gamma function (see e.g. [51] for further details). Among this family, we take the Matérn C^2 (M2) and C^4 (M4) kernels that, for $\boldsymbol{x},\boldsymbol{z} \in \Omega$, are respectively given by:

$$\kappa_\ell^{\mathrm{M2}}(\boldsymbol{x},\boldsymbol{z}) = \left(1+\frac{\sqrt{3}}{\ell}||\boldsymbol{x}-\boldsymbol{z}||_2\right)\exp\left(\frac{\sqrt{3}}{\ell}||\boldsymbol{x}-\boldsymbol{z}||_2\right),$$

and

$$\kappa_\ell^{\mathrm{M4}}(\boldsymbol{x},\boldsymbol{z}) = \left(1+\frac{\sqrt{5}}{\ell}||\boldsymbol{x}-\boldsymbol{z}||_2+\frac{\sqrt{5}}{3\ell}||\boldsymbol{x}-\boldsymbol{z}||_2^2\right)\exp\left(\frac{\sqrt{5}}{\ell}||\boldsymbol{x}-\boldsymbol{z}||_2\right),$$

which are recovered from (13) by fixing $\nu = 3/2$ and $\nu = 5/2$.

In our setting, aside the length scale kernel parameter $\ell \in \mathbb{R}_+$, we further consider the process variance $\sigma \in \mathbb{R}_+$. Indeed, even if the latter is irrelevant for computing the deterministic interpolant, it plays a role in defining the Kriging variance. For this reason, in (5) we defined $\kappa(\boldsymbol{x},\boldsymbol{z}) = \sigma^2\kappa_\ell(\boldsymbol{x},\boldsymbol{z})$, $\boldsymbol{x},\boldsymbol{z} \in \Omega$. Both the parameters σ and ℓ affect the accuracy of the approximation and associated uncertainty. In the following tests we might employ the same default parameters used by `fitrgp.m`, i.e.

$$\ell = \text{mean}\left(\text{std}(\{\boldsymbol{x}_i\}_{i=1}^n)\right), \quad \sigma = \frac{\text{std}(\{f_i\}_{i=1}^n)}{\sqrt{2}}, \tag{14}$$

where the mean is along the dimensions. Then, for the KEPU implementation with *default* parameters we select

$$\ell = \frac{\sum_{j=1}^m \ell_j}{m}, \quad \sigma = \frac{\sum_{j=1}^m \sigma_j}{m}, \tag{15}$$

where ℓ_j and σ_j, $j = 1,\ldots,m$, are computed as in (14) on each Ω_j.

Nevertheless, one could optimize the length scale and the process variance on each patch. In that case, following [23, §14], we minimize a profile likelihood function, where $\sigma_j = \sigma_j(\ell_j)$, $j = 1,\ldots,m$, and this immediately gives the *optimal* process standard deviation as:

$$\sigma_j^* = \sqrt{\frac{1}{n_j}\boldsymbol{f}_j^\intercal \mathsf{K}_j^{-1}\boldsymbol{f}_j}, \tag{16}$$

while the optimal local length scale is the minimum of

$$n_j \log(\boldsymbol{f}_j^\intercal \mathsf{K}_j^{-1}\boldsymbol{f}_j) + \log\det \mathsf{K}_j, \quad j = 1,\ldots,m. \tag{17}$$

The above minimum problem is computationally addressed via the `fminbnd.m` function that belongs to the Matlab Optimization toolbox. An application can also be found in [12].

To conclude this section, we point out that the `fitrgp.m` routine offers the opportunity to tune both the kernel parameters, and the default way to achieve this is based on cross-validation strategies and quasi-Newton optimization techniques. Hence, in the numerical experiments that follow, we will consider this option as well and carry out some comparisons.

5. Numerical experiments

The key feature of our KEPU is that the Kriging approximant and its variance can be computed for large data sets on *standard* calculators. Hence, tests are carried out on an Intel(R) Core(TM) i5-6400 CPU 2.70GHz processor (64 bit); 8 GB RAM. Referring to the previous section, in the following experiments, we make use of both the canonical implementation and of the one based on the `fitrgp.m` routine. The former Matlab software is available at:

`https://github.com/emmaA89/KEPU`.

Such a free code could be further speed up by running the KEPU method in parallel; to achieve this we refer the reader to [15].

In the incoming experiments, we play with different kernels with default and/or optimized parameters. In doing so, we take both univariate and bivariate data sets (with and without noise), and to test the efficiency we let n vary. Specifically, n Halton, grid and random training data on Ω, with $\sqrt{n} = 2^p + 1$, $p = 3, \dots, 9$, are considered. We point out that for huge values of n we *stop* the computation of the global Kriging estimator when either the software returns an out-of-memory message or when it requires more than eight hours of calculations.

To check the accuracy, the KEPU is evaluated on grid test sets $\Xi = \{\boldsymbol{\xi}_i, i = 1, \dots, v\} \subseteq \Omega$. Then, letting $\bar{f}$ be the KEPU approximant of a function f, we might compute the following quantities:

- Root Mean Squared Error:

$$\text{RMSE} = \sqrt{\frac{1}{v}\sum_{i=1}^{v} \left(\bar{f}(\boldsymbol{\xi}_i) - f(\boldsymbol{\xi}_i)\right)^2};$$

- Absolute Error:

$$\mathbf{AE} = \left(\left|\bar{f}(\boldsymbol{\xi}_1) - f(\boldsymbol{\xi}_1)\right|, \dots, \left|\bar{f}(\boldsymbol{\xi}_v) - f(\boldsymbol{\xi}_v)\right|\right);$$

- Mean of the Kriging Variance:

$$\text{MKV} = \frac{1}{v}\sum_{i=1}^{v}\left(\sum_{j=1}^{m} \mathcal{P}_j^2(\boldsymbol{\xi}_i) w_j^2(\boldsymbol{\xi}_i)\right);$$

- Absolute Kriging Variance:

$$\mathbf{AKV} = \left(\sum_{j=1}^{m} \mathcal{P}_j^2(\boldsymbol{\xi}_1) w_j^2(\boldsymbol{\xi}_1), \dots, \sum_{j=1}^{m} \mathcal{P}_j^2(\boldsymbol{\xi}_v) w_j^2(\boldsymbol{\xi}_v)\right).$$

5.1. Test 1d: canonical implementation without noise

We consider noise-free samples on Halton data of the following test function

$$f_1(x_1) = \sin(10\pi(x_1 + 0.1)), \quad x_1 \in \Omega = [0, 1].$$

The KEPU interpolant is constructed by taking $m = \sqrt{n}$ equispaced subdomain centres and is then evaluated on $v = 100$ equispaced points. The experiments are performed using a canonical implementation for the KEPU interpolant with the M2 kernel and the parameters are set as in (15). The results are compared with the global Kriging method. For the latter, the memory requirement becomes prohibitive when n is greater than about 10000. In Figure 1 (top left) we report the CPU times and, since the samples are not affected by noise, the RMSEs in Figure 1 (bottom left). We observe that, while the RMSEs returned by the global and local Kriging estimators are comparable, the CPU times are significantly different. Precisely, consistently with what observed in Section 4, when n is larger than about 4000 data, the KEPU leads to a significant saving in terms of computational complexity and memory needs (the global interpolation matrix cannot be allocated for huge values of n).

As second test, we repeat this experiment by optimizing both the process variance and length scale parameters as in (16) and (17). The results are shown in Figure 1 (right). The CPU times are then the sum of the tuning and fitting phases. We note that the saving in terms of computational time offered by the KEPU scheme becomes even more evident, and that, as expected, for both methods the RMSEs are lower than the ones computed in the previous test.

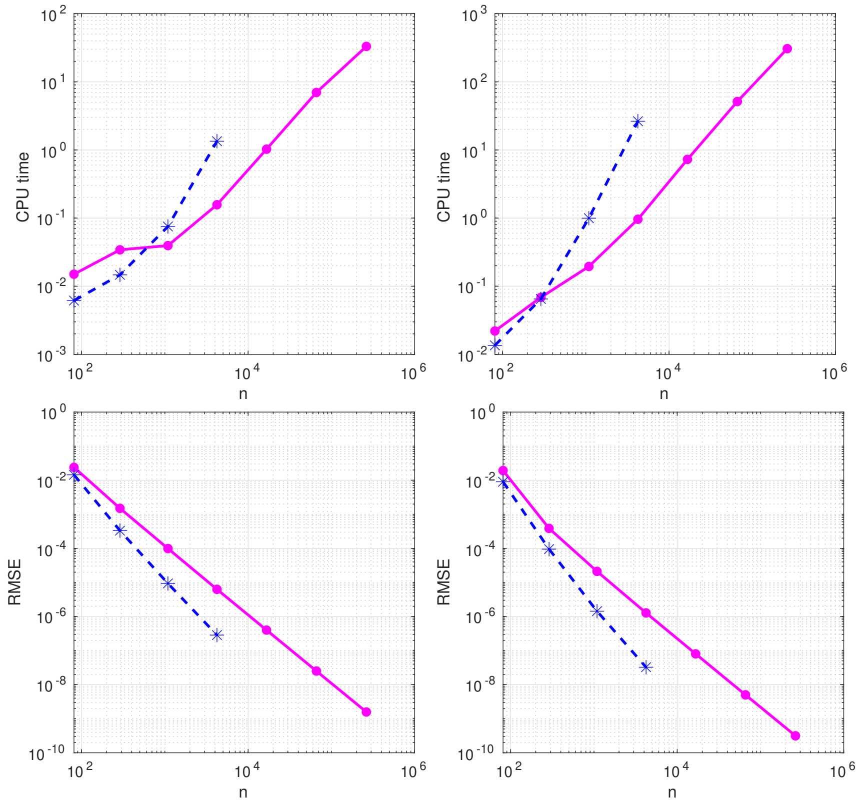

Figure 1: Results for the test function f_1 sampled without noise at Halton data: CPU times (top) and RMSEs (bottom) for the KEPU (magenta dots and solid line) and GKE (blue stars and dashed line). Left: both methods are computed with default parameters. Right: both methods are computed with optimized process variance and length scale parameters. Plots are in logarithmic scale.

We conclude this subsection pointing out that in our canonical implementation, if the samples are noisy, one might compute the Kriging coefficients as in Remark 3.2, and an effective choice for the regression parameter is to set it as $\lambda = 1/\ \mathrm{SNR}(\boldsymbol{f})$, where the Signal to Noise Ratio (SNR) can be computed via the `snr.m` MATLAB function that belongs to the Signal Processing Toolbox. As an alternative for regression purposes, one could use the more sophisticated `fitrgp.m` routine, as done in the next subsection.

5.2. *Test 1d:* `fitrgp.m` *implementation with noise*

We consider equispaced samples with, *for instance,* Gaussian withe noise of the following test function

$$f_2(x_1) = \frac{\cos(14\pi(x_1+0.5))}{2x_1+0.5} + (x_1-0.5)^4, \quad x_1 \in \Omega = [0,1],$$

i.e.

$$f_i = f_2(x_i) + 0.1\epsilon_i, \quad i = 1,\dots,n, \tag{18}$$

where ϵ_i are random perturbations obtained with the MATLAB `randn.m` routine. As in the previous case, the KEPU interpolant is constructed by taking $m = \sqrt{n}$ equispaced subdomain centres and is then evaluated on $v = 100$ equispaced points.

The first experiment for the KEPU is carried out by using the implementation offered by the `fitrgp.m` function with the GA kernel and the parameters set as in (15). The results are compared with the global Kriging algorithm (implemented with `fitrgp.m` as well) and are shown in Figure 2 (left) and in Table 1, where respectively the CPU times and the MKVs are reported. With the help of the `fitrgp.m` routine, since the global matrix is not explicitly stored, we observe that the global Kriging estimator is computed for about 60000 data. However, the same does not hold true for the Kriging variance. Indeed, the routine does not return the confidence intervals for $n > 10000$.

As further test (see Figures 2 (right) and Table 1), we repeat this experiment with the M4 kernel and random samples. Moreover, we optimize for the KEPU both the process variance and length scale parameters as in (16) and (17). This is compared with a global implementation of the Kriging estimator, where the optimal parameters are approximated by the `fitrgp.m` routine itself. In this case the computational effort for the global method becomes for n larger than about 5000 data. Moreover, the saving in terms of computational time offered by the KEPU implementation are meaningful also for relatively small data sets. We further observe that the MKVs of the two methods are comparable and are essentially dictated by the noise of data. For a graphical feedback, we refer the reader to Figure 3.

5.3. *Test 2d: applications to finance*

In this subsection we test the proposed tool in higher dimensions. We consider bivariate data that simulate the second derivative of an option price with respect to the stock price γ of an option. The example is directly taken by the MATLAB Financial toolbox. The following test shows how γ changes with

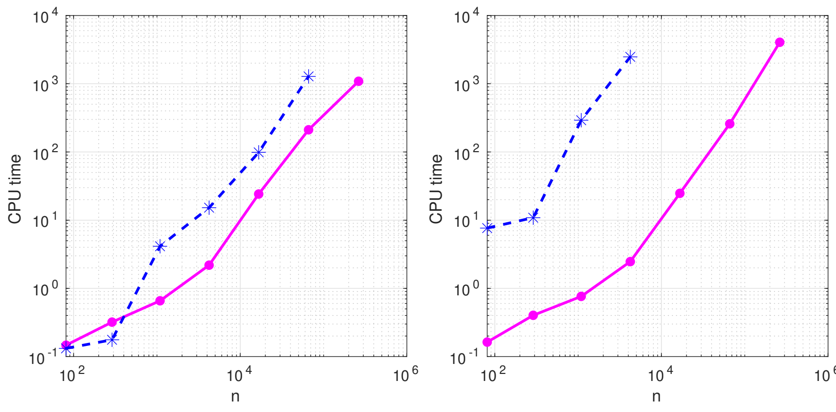

Figure 2: Results for the test function f_2 sampled (with noise) at equispaced (left) and random (right) data: CPU times for the KEPU (magenta dots and solid line) and for the GKE (blue stars and dashed line). Left: both methods are computed with default parameters for the GA kernel. Right: the KEPU is computed via the M4 kernel with optimized process variance and length scale parameters as in (16) and (17), while the GKE is computed with the process variance and length scale parameters optimized via the `fitrgp.m` itself. Plots are in logarithmic scale.

Table 1: MKVs for the test function f_2 sampled (with noise) at equispaced and random data. Second and third column: both methods are computed with default parameters for the GA kernel. Fourth and fifth column: the KEPU is computed via the M4 kernel with optimized process variance and length scale parameters as in (16) and (17), while the GKE is computed with process variance and length scale parameters optimized via the `fitrgp.m` itself.

	Non-optimized (GA)		Optimized (M4)	
n	KEPU	GKE	KEPU	GKE
81	8.97e − 03	9.83e − 01	8.84e − 02	6.63e − 03
289	1.44e − 02	1.14e + 00	1.37e − 02	2.03e − 02
1089	2.13e − 02	3.52e − 02	2.08e − 02	3.64e − 02
4225	2.39e − 02	3.77e − 02	2.39e − 02	3.57e − 02
16641	2.75e − 02	–	2.79e − 02	–
66049	3.16e − 02	–	3.13e − 02	–
263169	3.40e − 02	–	3.45e − 02	–

respect to the value of a price for a Black-Scholes option in time, see e.g. [6, 42]. The considered price (x_1-axis) varies from 10\$ to 70\$ and the time (x_2-axis) is one year, i.e. $\Omega = [10, 70] \times [1, 12]$. As training data we consider grids and we let n vary as in the previous tests. Then, the sampling data $\boldsymbol{f}$ (obtained via the `blsgamma.m` MATLAB routine) represent the value of γ, i.e. the gamma function, that is calculated by fixing the exercise price to 40\$, the risk-free interest rate to 10%, and the volatility to 0.35 for all prices and periods.

In the first test, we sample without noise the gamma-function on n grid points, with $\sqrt{n} = 2^p + 1$, $p = 3, \ldots, 9$, and to check the accuracy, the KEPU

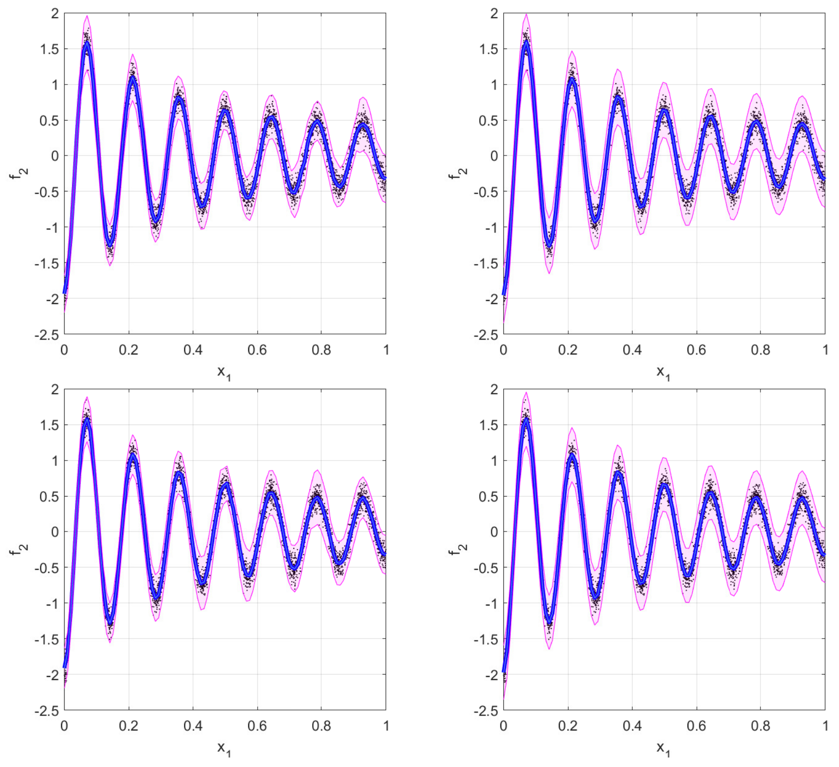

Figure 3: Graphical results ($n = 4225$) for the test function f_2 sampled (with noise) at equispaced (top) and random (bottom) data. Left: the KEPU. Right: the GKE. Top: both methods are computed with default parameters for the GA kernel. Bottom: the KEPU is computed via the M4 kernel with optimized process variance and length scale parameters as in (16) and (17), while the GKE is computed with process variance and length scale parameters optimized via the `fitrgp.m` itself. Training data are represented by black dots, the approximant by blue line and the shade magenta area denotes two times the Kriging standard deviation, i.e. the confidence intervals as in (12).

(constructed by fixing the default kernel parameters for the M4 function, by taking $m = \sqrt{n/2}$ patch centres and by using the `fitrgp.m` routine) is evaluated on grid test sets $\Xi = \{\boldsymbol{\xi}_i, i = 1, \dots, v\} \subseteq \Omega$, with $v = 40^2$. The CPU times and RMSEs are depicted in Figure 4. We note that the RMSEs of the KEPU and of the GKE are comparable. This is also confirmed by Figure 5 (top), where the reconstructed surfaces, false-colored with the **AE**, are reported. As far as the CPU times are concerned, we note that, since the selected subdomains are less than in the 1d test case and hence contain more points, the KEPU saving in terms of computational time is only apparently less evident. Indeed, for the bivariate case, the complexity of the global method is already for about 10000 data.

As second test, we repeat this experiment by perturbing the samples as in (18) with $\epsilon_i = 0.001$. The **AKV**s are reported in Table 2 and we observe that the results obtained with the KEPU are comparable with the ones returned by the GKE. Moreover, for a graphical feedback, refer to Figure 5 (bottom), where we show the reconstructed surfaces false-colored with the **AKV**. From that figure we further note that, both the **AE** and the **AKV** computed with the GKE are more uniformly distributed on Ω, while in the local case appear to vary more. This is a typical consequence of local computations, but the accuracy scales are comparable.

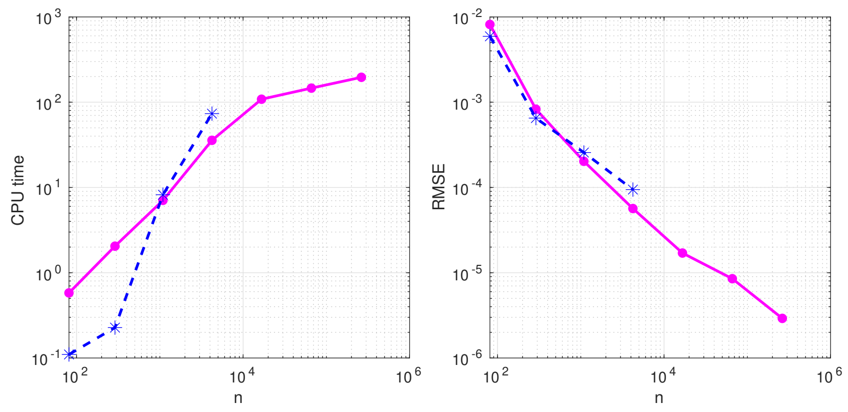

Figure 4: Results for the gamma function sampled without noise at grid data: CPU times (left) and RMSEs (right) for the KEPU (magenta dots and solid line) and GKE (blue stars and dashed line). Plots are in logarithmic scale.

Table 2: MKVs for the gamma function sampled with noise at grid data. Both methods are computed with default parameters for the M4 kernel.

n	KEPU	GKE
81	7.79e − 06	1.26e − 05
289	1.10e − 06	2.37e − 06
1089	8.39e − 07	4.28e − 06
4225	9.83e − 07	3.44e − 06
16641	1.03e − 06	−
66049	1.02e − 06	−
263169	1.14e − 06	−

6. Conclusions and work in progress

We have presented an efficient domain-decomposition algorithm for computing the Kriging estimator, namely the KEPU. The theoretical results show that the KEPU inherits many properties of the global Kriging predictor. As a consequence, its accuracy is comparable to the one of the global method. However,

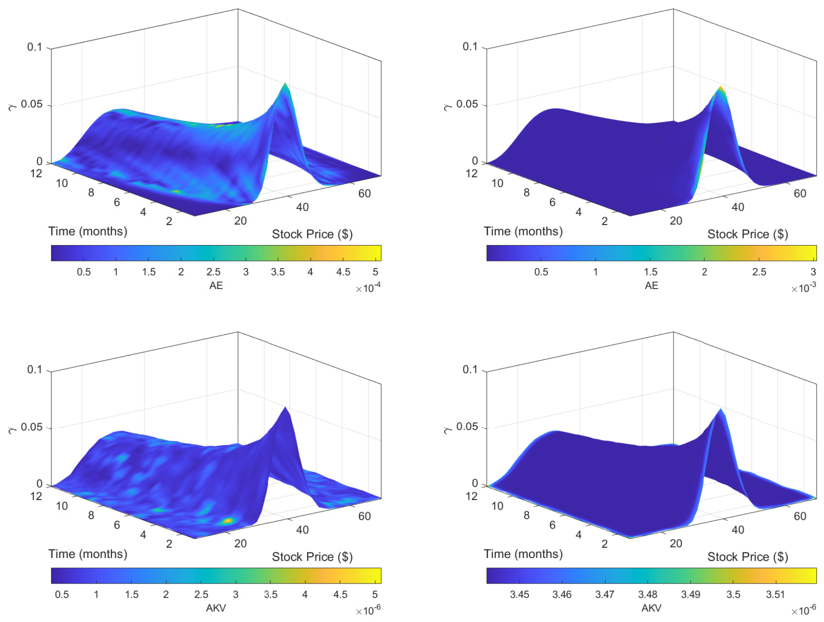

Figure 5: Graphical results ($n = 4225$) for the gamma function sampled without noise (top) and with noise (bottom) at grid data. Left: the KEPU. Right: the GKE. The reconstructed surfaces are false-colored with the **AE** (top) and **AKV** (bottom).

the main feature of the KEPU is that it is not so computationally demanding and hence fast.

Future work consists in extending the proposed idea to other kinds of kernel bases, as the variably scaled kernels (see [8]), and to use it for approximating surfaces defined by point cloud data.

Statements & Declarations

Funding & Acknowledgments

The work of the first and second author has been supported by the INdAM–GNCS 2022 Project “Computational methods for kernel-based approximation and its applications”, code CUP_E55F22000270001. Moreover, the first author has been supported by the 2020 Project “Mathematical methods in computational sciences” funded by the Department of Mathematics “Giuseppe Peano” of the University of Torino, while the work of the second author has also been supported by the Spoke 1 “FutureHPC & BigData” of the Italian Research Center on High-Performance Computing, Big Data and Quantum Computing (ICSC) funded by MUR Missione 4 Componente 2 Investimento 1.4: Potenziamento strutture di ricerca e creazione di “campioni nazionali di R&S (M4C2-19)” – Next Generation EU (NGEU). This research has been accomplished within the

RITA "Research ITalian network on Approximation" and the UMI Group TAA "Approximation Theory and Applications". We sincerely thank the reviewers for helping us to significantly improve the paper.

Competing Interests

The authors have no relevant financial or non-financial interests to disclose.

Author Contributions

All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Emma Perracchione, Roberto Cavoretto and Alessandra De Rossi. The first draft and experiments was carried out by Emma Perracchione and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Data Availability

The datasets generated during and/or analysed during the current study are publicly available and details are provided in the manuscripts.

References

[1] G. Allasia, R. Cavoretto, A. De Rossi, *Hermite-Birkhoff interpolation on scattered data on the sphere and other manifolds*, Appl. Math. Comput. **318** (2018), 35–50.

[2] S. Arefian, D. Mirzaei, *A compact radial basis function partition of unity method*, Comput. Math. Appl. **127** (2022), 1–11.

[3] I. Babuška, J.M. Melenk, *The partition of unity method*, Int. J. Numer. Meth. Eng. **40** (1997), 727–758.

[4] A. Berlinet, C. Thomas-Agnan, *Reproducing Kernel Hilbert Spaces in Probability and Statistics*, Kluwer, Dordrecht, 2004.

[5] D.S. Bernstein *Matrix Mathematics: Theory, Facts, and Formulas, 2nd edn.*, Princeton University Press, Princeton, N.J., 2009.

[6] F. Black, M. Scholes, *The pricing of options and corporate liabilities*, J. Polit. Econ. **81** (1973), 637–654.

[7] L. Bottou, V. Vapnik, *Local learning algorithms*, Neural Computation **4** (1992), 888–900.

[8] M. Bozzini, L. Lenarduzzi, M. Rossini, R. Schaback, *Interpolation with variably scaled kernels*, IMA J. Numer. Anal. **35** (2015), 199–219.

[9] L. Breiman, *Random forests*, Mach. Learn. **45** (2001), 5–32.

[10] C. Campi, F. Marchetti, E. Perracchione, *Learning via variably scaled kernels*, Adv. Comput. Math. **47** (2021), 51.

[11] R. Cavoretto, A. De Rossi, *A meshless interpolation algorithm using a cell-based searching procedure*, Comput. Math. Appl. **67** (2014), 1024–1038.

[12] R. Cavoretto, A. De Rossi, *An adaptive residual sub-sampling algorithm for kernel interpolation based on maximum likelihood estimations*, J. Comput. Appl. Math. **418** (2023), 114658.

[13] R. Cavoretto, A. De Rossi, *Error indicators and refinement strategies for solving Poisson problems through a RBF partition of unity collocation scheme*, Appl. Math. Comput. **369** (2020), 124824.

[14] R. Cavoretto, A. De Rossi, F. Dell'Accio, F. Di Tommaso, *An efficient trivariate algorithm for tetrahedral Shepard interpolation*, J. Sci. Comput. **82** (2020), 57.

[15] R. Cavoretto, T. Schneider, P. Zulian, *OpenCL based parallel algorithm for RBF-PUM interpolation*, J. Sci. Comput. **74** (2018), 267–289.

[16] J.P. Chilès, N. Desassis, *Fifty years of Kriging.* In: B. Daya Sagar et al. (eds), Handbook of Mathematical Geosciences, Springer, Cham, 2018, 589–612.

[17] C. Cortes, V.N. Vladimir, *Support-vector networks*, Machine Learning **20** (1995), 273–297.

[18] L. Csató, M. Opper, *Sparse On-Line Gaussian Processes*, Neural Comput. **14** (2002), 641–668.

[19] S. De Marchi, A. Martínez, E. Perracchione, *Fast and stable rational RBF-based partition of unity interpolation*, J. Comput. Appl. Math. **349** (2019), 331–343.

[20] S. De Marchi, R. Schaback, H. Wendland, *Near-optimal data-independent point locations for radial basis function interpolation*, Adv. Comput. Math. **23** (2005), 317–330.

[21] K.L. Du, M.N.S Swamy, *Neural Networks and Statistical Learning*, Springer, London, 2010.

[22] G.E. Fasshauer, *Meshfree Approximations Methods with Matlab*, World Scientific, Singapore, 2007.

[23] G.E. Fasshauer, M.J. McCourt, *Kernel-based Approximation Methods Using* Matlab, World Scientific, Singapore, 2015.

[24] M. Fuhry, L. Reichel, *A new Tikhonov regularization method*, Numer. Algorithms **59** (2012), 433–445.

[25] S. Guastavino, F. Benvenuto, *Convergence Rates of Spectral Regularization Methods: A Comparison between Ill-Posed Inverse Problems and Statistical Kernel Learning*, SIAM J. Num. Anal. **58** (2020), 3504–3529.

[26] S. Guastavino, F. Benvenuto, *A consistent and numerically efficient variable selection method for sparse Poisson regression with applications to learning and signal recovery*, Stat. Comput. **29** (2019), 501–516.

[27] L. Hartman, O. Hössjer, *Fast Kriging of large datasets with Gaussian Markov random fields*, Comput. Stat. Data Anal. **52** (2008), 2331–2349.

[28] T. Joachims, C.N.J. Yu, *Sparse kernel SVMs via cutting-plane training*, Machine Learning **76** (2009), 179–193.

[29] D.G. Krige, *A statistical approach to some basic mine valuation problems on the Witwatersrand*, J. Chem. Met. & Mining Soc., S. Africa **52** (1951), 119–139.

[30] E. Larsson, V. Shcherbakov, A. Heryudono, *A least squares radial basis function partition of unity method for solving PDEs*, SIAM J. Sci. Comput. **39** (2017), A2538–A2563.

[31] N.D. Lawrence, *Gaussian process latent variable models for visualisation of high dimensional data*, Adv. Neural. Inf. Proces. Syst. **16** (2004), 329–336.

[32] S. Maji, A.C. Berg, J. Malik, *Efficient classification for additive kernel SVMs*, in IEEE PAMI, vol. 35, 2013, 66–77.

[33] F. Marchetti, E. Perracchione, *Local-to-Global Support Vector Machines (LGSVMs)*, Pattern Recognit. **132** (2022), 108920.

[34] P. Massa, S. Garbarino, F. Benvenuto, *Approximation of discontinuous inverse operators with neural networks*, Inverse Probl. **38** (2022), 105001.

[35] Matlab R2021b and Statistics and Machine Learning Toolbox, The MathWorks, Inc., Natick, Massachusetts, USA.

[36] B. Matérn, *Spatial Variation*, Lecture Notes in Statistics, Springer-Verlag, vol. 36, 1986.

[37] A.K. Menon, *Large-scale support vector machines: Algorithms and theory*, technical report, University of California San Diego (2009).

[38] D. Mirzaei, *The direct radial basis function partition of unity (D-RBF-PU) method for solving PDEs*, SIAM J. Sci. Comput. **43** (2021), A54–A83.

[39] A. Naish-Guzman, S. Holden, *The generalized FITC approximation* In: J. Platt et al. (eds), Advances in neural information processing systems, vol. 4, 2008, 1057–1064.

[40] D. Nguyen-Tuong, M. Seeger, J. Peters, *Model learning with local Gaussian process regression*, Adv. Robot. **23** (2009), 2015–2034.

[41] D. Rullière, N. Durrande, F. Bachoc, C. Chevalier, *Nested kriging predictions for datasets with a large number of observations*, Stat. Comput. **28** (2018), 849–867.

[42] A. Safdari-Vaighani, A. Heryudono, E. Larsson, *A radial basis function partition of unity collocation method for convection-diffusion equations*, J. Sci. Comput. **64** (2015), 341–367.

[43] B. Schölkopf, A.J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, Cambridge, MA, USA, 2002.

[44] N. Segata, E. Blanzieri, *Fast local support vector machines for large datasets*, In: P. Perner P. (eds) Machine Learning and Data Mining in Pattern Recognition. MLDM 2009. Lecture Notes in Computer Science, vol. 5632, 2009, 295–310.

[45] D. Shepard, *A two-dimensional interpolation function for irregularly spaced data*, in: Proceedings of 23-rd National Conference, Brandon/Systems Press, Princeton, 1968, 517–524.

[46] J. Shawe-Taylor, N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge Univ. Press, 2004.

[47] A.N. Tikhonov, *Solution of incorrectly formulated problems and the regularization method*, Sov. Math. Dokl. **4** (1963), 1035–1038.

[48] V. Tresp, *A Bayesian Committee Machine*, Neural Comput. **12** (2000), 2719–2741.

[49] B. van Stein, H. Wang, W. Kowalczyk, M. Emmerich, Thomas Bäck, *Cluster-based Kriging approximation algorithms for complexity reduction*, Appl. Intell. **50** (2020), 778–791.

[50] G. Wahba *Spline Models for Observational Data*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1990.

[51] H. Wendland, *Scattered Data Approximation*, Cambridge Monogr. Appl. Comput. Math., vol. 17, Cambridge Univ. Press, Cambridge, 2005.

[52] H. Wendland, *Fast evaluation of radial basis functions: Methods based on partition of unity*, in: C.K. Chui et al. (eds), Approximation Theory X: Wavelets, Splines, and Applications, Vanderbilt Univ. Press, Nashville, 2002, 473–483.

[53] H. Wendland, *Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree*, Adv. Comput. Math. **4** (1995), 389–396.