

Learning with Partition of Unity-based Kriging Estimators

Original

Learning with Partition of Unity-based Kriging Estimators / Cavoretto, R.; De Rossi, A.; Perracchione, E.. - In: APPLIED MATHEMATICS AND COMPUTATION. - ISSN 0096-3003. - 448:(2023), pp. 1-14. [10.1016/j.amc.2023.127938]

Availability:

This version is available at: 11583/2977112 since: 2023-03-16T10:54:59Z

Publisher:

Elsevier

Published

DOI:10.1016/j.amc.2023.127938

Terms of use:

openAccess

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

Publisher copyright

(Article begins on next page)

Learning with Partition of Unity-based Kriging Estimators

R. Cavoretto^{*, \diamond} , A. De Rossi^{*, \diamond} , E. Perracchione^{+, \diamond}

^{*}*Dipartimento di Matematica “Giuseppe Peano”, Università di Torino, Italy*

⁺*Dipartimento di Scienze Matematiche “Giuseppe Luigi Lagrange”, Politecnico di Torino, Italy*

^{\diamond} *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^{*, \diamond}),
alessandra.derossi@unito.it (A. De Rossi^{*, \diamond}), emma.perracchione@polito.it (E. Perracchione^{+, \diamond})

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

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

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

34 In this paper we look for an efficient computation of the Kriging Estimator
35 (KE) whose background lies in the context of approximation theory. Precisely,
36 the so-called Partition of Unity (PU) scheme, first introduced in the mid 1990s
37 in [3], is nowadays a well-established method for approximating large data sets
38 and it is also rather popular for researchers working on collocation schemes or
39 numerical solution of Partial Differential Equations (PDEs); refer e.g. to [2, 13,
40 30, 38]. We then make use of such a partitioning scheme to compute both the
41 Kriging predictions and uncertainties. The resulting method, namely KEPU,
42 drastically reduces the computation complexity of global Gaussian processes, as
43 numerically shown, and inherits properties from the local Kriging estimators.
44 Indeed, we prove that it is unbiased and that the Kriging uncertainty of the
45 KEPU is a squared weighted sum of the local Mean Squared Errors (MSEs).
46 Hence, as a benefit of this study we are able to include the PU method in a
47 machine learning and stochastic framework, providing predictions at query data
48 and related Kriging variances.

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

54 conclusions are offered in Section 6.

55 2. Preliminaries

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

59 2.1. Kernel-based interpolation

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

$$\tilde{f}(\mathbf{x}_i) = f(\mathbf{x}_i), \quad i = 1, \dots, n. \quad (1)$$

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

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

where $\alpha_1, \dots, \alpha_n$ 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 $\kappa : \Omega \times \Omega \rightarrow \mathbb{R}$ for $H_\kappa(\Omega)$, i.e. so that (refer e.g. to [23, Definition 2.6, p. 32]) $\kappa(\cdot, \mathbf{x}) \in H_\kappa(\Omega)$, $\kappa(\mathbf{x}, \mathbf{z}) = \kappa(\mathbf{z}, \mathbf{x})$ for all $\mathbf{x}, \mathbf{z} \in \Omega$, and

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

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

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

70 Equivalently, $\kappa : \Omega \times \Omega \rightarrow \mathbb{R}$ is a reproducing kernel if there exists a mapping
 71 $\Phi : \Omega \rightarrow H_\kappa(\Omega)$, usually referred to as *feature map* [46, §5], so that:

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

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

75 $\sigma > 0$ (known in machine learning literature as *length scale* and *process variance*,
76 respectively) such that, for all $\mathbf{x}, \mathbf{z} \in \Omega$,

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

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

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

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

80 where $\kappa : \Omega \times \Omega \rightarrow \mathbb{R}$ is a positive definite radial kernel. Indeed for those kernels
81 the solution of the interpolation problem is unique (refer e.g. to [23, Definition
82 2.2, p. 18]). Precisely, letting $\mathbf{f} = (f(\mathbf{x}_1), \dots, f(\mathbf{x}_n))^\top$ and $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n)^\top$,
83 the scattered data interpolation problem reduces to solving the linear system of
84 the form:

$$\mathbf{K}\boldsymbol{\alpha} = \mathbf{f}, \quad (6)$$

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

86 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.

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

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

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

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

With these ingredients, we can define the PU interpolant as

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

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

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

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

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

$$\mathbf{K}_j \boldsymbol{\alpha}_j = \mathbf{f}_j, \quad (7)$$

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

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

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

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

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

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

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

104 3. Kriging estimator based on partition of unity

For a fixed $\mathbf{x} \in \Omega$, the main requirement in the Kriging prediction is the one of assuming that the value $f(\mathbf{x})$ is a realisation of a random variable $F_{\mathbf{x}}$ 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])

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

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

108 *3.1. Localized Gaussian fitting*

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

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

$$\bar{F}_{\mathbf{x}} = \sum_{j=1}^m \tilde{F}_{\mathbf{x}}^j w_j(\mathbf{x}) = \sum_{j=1}^m (\mathbf{k}_j(\mathbf{x})^\top \mathbf{K}_j^{-1} \mathbf{F}_j) w_j(\mathbf{x}), \quad \mathbf{x} \in \Omega, \quad (9)$$

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

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

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

$$Y_{\mathbf{x}}^j \sim \mathcal{N}(\mathbf{k}_j(\mathbf{x})^\top \mathbf{K}_j^{-1} \mathbf{f}_j, \kappa(\mathbf{x}, \mathbf{x}) - \mathbf{k}_j(\mathbf{x})^\top \mathbf{K}_j^{-1} \mathbf{k}_j(\mathbf{x})). \quad (10)$$

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

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

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

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

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

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

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

123 Then, Equation (9) concludes the proof. ■

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

128 We further note that each of the local estimators is unbiased, i.e., $\mathbb{E}(F_{\mathbf{x}}) =$
 129 $\mathbb{E}(F_{\mathbf{x}}^j)$, $j = 1, \dots, m$, and such a property is inherited by the PU scheme, as
 130 shown in the following result.

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

Proof: We have that:

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

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

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

132

■

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

137 3.2. Localized Gaussian uncertainties

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

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

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

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

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

144 *is the power function (computed on Ω_j).*

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

$$\text{Var}(Y_{\mathbf{x}}) = \sum_{j=1}^m (\kappa(\mathbf{x}, \mathbf{x}) - \mathbf{k}_j(\mathbf{x})^\top \mathbf{K}_j^{-1} \mathbf{k}_j(\mathbf{x})) w_j^2(\mathbf{x}) = \sum_{j=1}^m \mathcal{P}_j^2(\mathbf{x}) w_j^2(\mathbf{x}).$$

145

■

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

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

$$\text{MSE}(\bar{F}_{\mathbf{x}}) = \sum_{j=1}^m (\kappa(\mathbf{x}, \mathbf{x}) - \mathbf{k}_j(\mathbf{x})^\top \mathbf{K}_j^{-1} \mathbf{k}_j(\mathbf{x})) w_j^2(\mathbf{x}).$$

151

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

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

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

$$\begin{aligned} \mathbb{E} \left(\sum_{j=1}^m (F_{\mathbf{x}} - \tilde{F}_{\mathbf{x}}^j)^2 w_j^2(\mathbf{x}) \right) &= \sum_{j=1}^m \mathbb{E} \left((F_{\mathbf{x}} - \tilde{F}_{\mathbf{x}}^j)^2 \right) w_j^2(\mathbf{x}) \\ &= \sum_{j=1}^m (\kappa(\mathbf{x}, \mathbf{x}) - \mathbf{k}_j(\mathbf{x})^\top \mathbf{K}_j^{-1} \mathbf{k}_j(\mathbf{x})) w_j^2(\mathbf{x}). \end{aligned}$$

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

$$\begin{aligned} &\mathbb{E} \left(\sum_{j < k} (F_{\mathbf{x}} - \tilde{F}_{\mathbf{x}}^j) (F_{\mathbf{x}} - \tilde{F}_{\mathbf{x}}^k) w_j(\mathbf{x}) w_k(\mathbf{x}) \right) \\ &= \sum_{j < k} \mathbb{E} \left((F_{\mathbf{x}} - \tilde{F}_{\mathbf{x}}^j) (F_{\mathbf{x}} - \tilde{F}_{\mathbf{x}}^k) \right) w_j(\mathbf{x}) w_k(\mathbf{x}) \\ &= \sum_{j < k} \mathbb{E} (F_{\mathbf{x}} - \tilde{F}_{\mathbf{x}}^j) \mathbb{E} (F_{\mathbf{x}} - \tilde{F}_{\mathbf{x}}^k) w_j(\mathbf{x}) w_k(\mathbf{x}) = 0, \end{aligned}$$

152 where the last equality follows from the fact that all the local estimators are
153 unbiased. ■

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

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

158 where z_{δ} is the quantile location of the normal distribution. Note that, the
159 above equation allows us to understand how the local uncertainties propagate
160 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 $\mathbf{x} \in \Omega$, 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])*

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

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 smoothing splines, ridge regression and Tikhonov regularization, which are typically used for Kriging regression, are based on solving for each subdomain [24, 47, 50]

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

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

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

161 with $j = 1, \dots, m$.

162 4. Complexity analysis and implementation

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

165 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

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

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

172 *4.2. Implementation details*

173 For the KEPU implementation, we have to solve (7) and then compute the
 174 uncertainty via Corollary 3.2.1. This is compared in terms of efficiency and
 175 accuracy with the Global Kriging Estimator (GKE). Aside this kind of imple-
 176 mentation, called in what follows *canonical*, we will make use of the MATLAB
 177 function `fitrgp.m` that belongs to the Statistics and Machine Learning tool-
 178 box [35]. Indeed, such a routine already offers an ad hoc implementation for
 179 huge values of n . Precisely, when $n > 10000$, it does not allocate the kernel
 180 matrix as in (6) (which, being a $n \times n$ matrix whose entries are in double-
 181 precision floating-point format, would require $8 \times n \times n$ bytes), but it takes
 182 advantage of a block coordinate descent algorithm. Nevertheless, such a strat-
 183 egy is not completely satisfying. Indeed, when $n > 10000$, `fitrgp.m` does not
 184 return the Kriging variance, which plays an important role in the stochastic
 185 predictions. On the opposite, with our KEPU, we are able to allocate the local
 186 matrices and hence compute the Kriging uncertainties also for large data sets
 187 (using the `fitrgp.m` on each subdomain). This is a consequence of the fact
 188 that the largest matrix that the algorithm needs to store has size $n_s \times n_s$, where
 189 $n_s = \max_{j=1, \dots, m} \text{card}(\Omega_j)$ 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

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

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

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

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 $\mathbf{x}, \mathbf{z} \in \Omega$, are respectively given by:

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

and

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

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

193 In our setting, aside the length scale kernel parameter $\ell \in \mathbb{R}_+$, we further
 194 consider the process variance $\sigma \in \mathbb{R}_+$. Indeed, even if the latter is irrelevant for
 195 computing the deterministic interpolant, it plays a role in defining the Kriging

196 variance. For this reason, in (5) we defined $\kappa(\mathbf{x}, \mathbf{z}) = \sigma^2 \kappa_\ell(\mathbf{x}, \mathbf{z})$, $\mathbf{x}, \mathbf{z} \in \Omega$. Both
 197 the parameters σ and ℓ affect the accuracy of the approximation and associated
 198 uncertainty. In the following tests we might employ the same default parameters
 199 used by `fitrgp.m`, i.e.

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

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

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

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

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

$$\sigma_j^* = \sqrt{\frac{1}{n_j} \mathbf{f}_j^\top \mathbf{K}_j^{-1} \mathbf{f}_j}, \quad (16)$$

207 while the optimal local length scale is the minimum of

$$n_j \log(\mathbf{f}_j^\top \mathbf{K}_j^{-1} \mathbf{f}_j) + \log \det \mathbf{K}_j, \quad j = 1, \dots, m. \quad (17)$$

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

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

216 5. Numerical experiments

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

223 <https://github.com/emmaA89/KEPU>.

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

226 In the incoming experiments, we play with different kernels with default
 227 and/or optimized parameters. In doing so, we take both univariate and bivariate
 228 data sets (with and without noise), and to test the efficiency we let n vary.
 229 Specifically, n Halton, grid and random training data on Ω , with $\sqrt{n} = 2^p +$
 230 1 , $p = 3, \dots, 9$, are considered. We point out that for huge values of n we
 231 *stop* the computation of the global Kriging estimator when either the software
 232 returns an out-of-memory message or when it requires more than eight hours of
 233 calculations.

234 To check the accuracy, the KEPU is evaluated on grid test sets $\Xi = \{\xi_i, i =$
 235 $1, \dots, v\} \subseteq \Omega$. Then, letting \bar{f} be the KEPU approximant of a function f , we
 236 might compute the following quantities:

- Root Mean Squared Error:

$$\text{RMSE} = \sqrt{\frac{1}{v} \sum_{i=1}^v (\bar{f}(\xi_i) - f(\xi_i))^2};$$

- Absolute Error:

$$\mathbf{AE} = (|\bar{f}(\xi_1) - f(\xi_1)|, \dots, |\bar{f}(\xi_v) - f(\xi_v)|);$$

- Mean of the Kriging Variance:

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

- Absolute Kriging Variance:

$$\mathbf{AKV} = \left(\sum_{j=1}^m \mathcal{P}_j^2(\xi_1) w_j^2(\xi_1), \dots, \sum_{j=1}^m \mathcal{P}_j^2(\xi_v) w_j^2(\xi_v) \right).$$

237 5.1. Test 1d: canonical implementation without noise

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

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

238 The KEPU interpolant is constructed by taking $m = \sqrt{n}$ equispaced subdomain
 239 centres and is then evaluated on $v = 100$ equispaced points. The experiments are
 240 performed using a canonical implementation for the KEPU interpolant with the
 241 M2 kernel and the parameters are set as in (15). The results are compared with
 242 the global Kriging method. For the latter, the memory requirement becomes
 243 prohibitive when n is greater than about 10000. In Figure 1 (top left) we report

244 the CPU times and, since the samples are not affected by noise, the RMSEs
 245 in Figure 1 (bottom left). We observe that, while the RMSEs returned by
 246 the global and local Kriging estimators are comparable, the CPU times are
 247 significantly different. Precisely, consistently with what observed in Section 4,
 248 when n is larger than about 4000 data, the KEPU leads to a significant saving in
 249 terms of computational complexity and memory needs (the global interpolation
 250 matrix cannot be allocated for huge values of n).

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

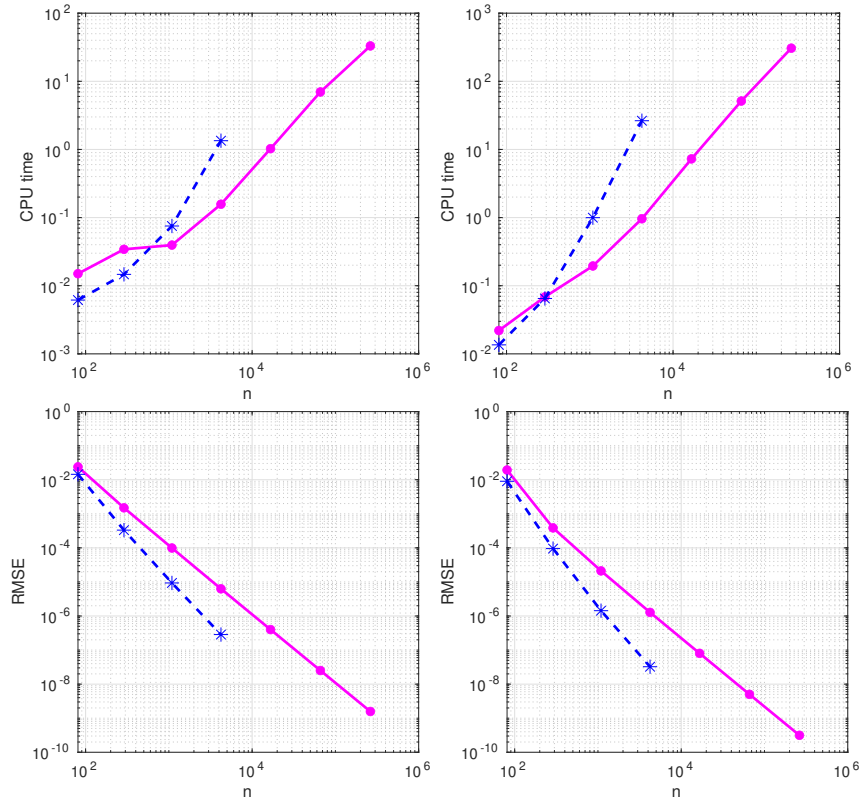


Figure 1: Results for the test function f_1 sampled without noise at Halton data: CPU times (top) and RMSEs (bottom) for the KEPU (magenta dots and solid line) and GKE (blue stars and dashed line). Left: both methods are computed with default parameters. Right: both methods are computed with optimized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

257 We conclude this subsection pointing out that in our canonical implemen-

258 tation, if the samples are noisy, one might compute the Kriging coefficients as
 259 in Remark 3.2, and an effective choice for the regression parameter is to set it
 260 as $\lambda = 1/\text{SNR}(\mathbf{f})$, where the Signal to Noise Ratio (SNR) can be computed
 261 via the `snr.m` MATLAB function that belongs to the Signal Processing Toolbox.
 262 As an alternative for regression purposes, one could use the more sophisticated
 263 `fitrgp.m` routine, as done in the next subsection.

264 *5.2. Test 1d: fitrgp.m implementation with noise*

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

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

265 i.e.

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

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

270 The first experiment for the KEPU is carried out by using the implementa-
 271 tion offered by the `fitrgp.m` function with the GA kernel and the parameters
 272 set as in (15). The results are compared with the global Kriging algorithm (im-
 273 plemented with `fitrgp.m` as well) and are shown in Figure 2 (left) and in Table
 274 1, where respectively the CPU times and the MKVs are reported. With the
 275 help of the `fitrgp.m` routine, since the global matrix is not explicitly stored,
 276 we observe that the global Kriging estimator is computed for about 60000 data.
 277 However, the same does not hold true for the Kriging variance. Indeed, the
 278 routine does not return the confidence intervals for $n > 10000$.

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

290 *5.3. Test 2d: applications to finance*

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

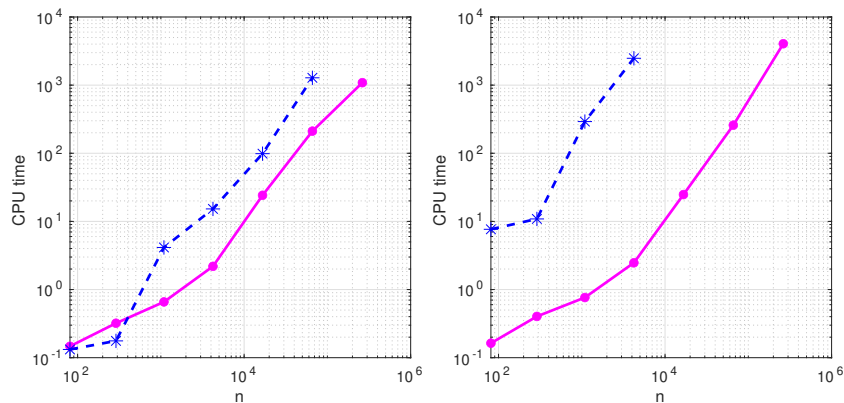


Figure 2: Results for the test function f_2 sampled (with noise) at equispaced (left) and random (right) data: CPU times for the KEPU (magenta dots and solid line) and for the GKE (blue stars and dashed line). Left: both methods are computed with default parameters for the GA kernel. Right: the KEPU 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.

| n | Non-optimized (GA) | | Optimized (M4) | |
|--------|--------------------|----------|----------------|----------|
| | 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 | - |

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

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

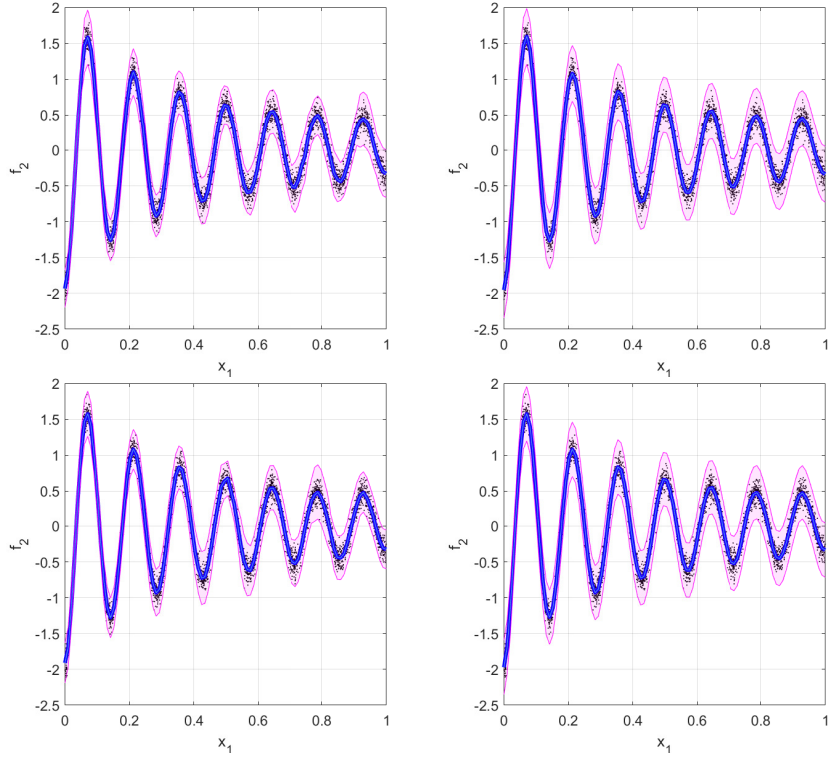


Figure 3: Graphical results ($n = 4225$) for the test function f_2 sampled (with noise) at equispaced (top) and random (bottom) data. Left: the KEPU. Right: the GKE. Top: both methods are computed with default parameters for the GA kernel. Bottom: the KEPU is computed via the M4 kernel with optimized 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).

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

315 As second test, we repeat this experiment by perturbing the samples as in
 316 (18) with $\epsilon_i = 0.001$. The **AKV**s are reported in Table 2 and we observe that

317 the results obtained with the KEPU are comparable with the ones returned
 318 by the GKE. Moreover, for a graphical feedback, refer to Figure 5 (bottom),
 319 where we show the reconstructed surfaces false-colored with the **AKV**. From
 320 that figure we further note that, both the **AE** and the **AKV** computed with
 321 the GKE are more uniformly distributed on Ω , while in the local case appear
 322 to vary more. This is a typical consequence of local computations, but the
 323 accuracy scales are comparable.

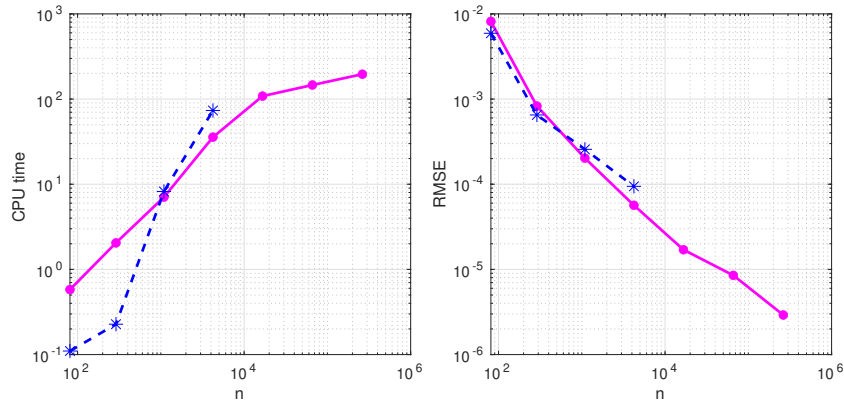


Figure 4: Results for the gamma function sampled without noise at grid data: CPU times (left) and RMSEs (right) for the KEPU (magenta dots and solid line) and GKE (blue stars and dashed line). Plots are in logarithmic scale.

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 | - |

324 6. Conclusions and work in progress

325 We have presented an efficient domain-decomposition algorithm for comput-
 326 ing the Kriging estimator, namely the KEPU. The theoretical results show that
 327 the KEPU inherits many properties of the global Kriging predictor. As a conse-
 328 quence, its accuracy is comparable to the one of the global method. However,

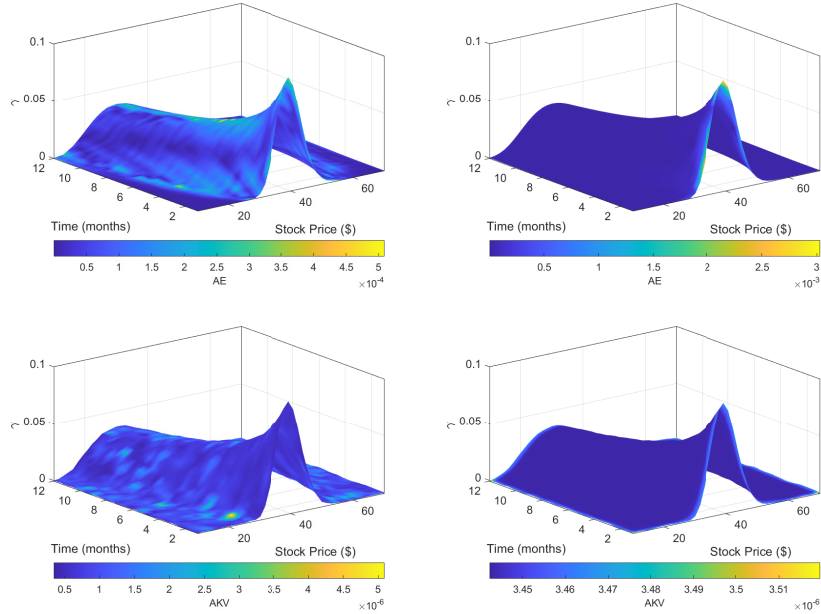


Figure 5: Graphical results ($n = 4225$) for the gamma function sampled without noise (top) and with noise (bottom) at grid data. Left: the KEPU. Right: the GKE. The reconstructed surfaces are false-colored with the **AE** (top) and **AKV** (bottom).

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

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

334 **Statements & Declarations**

335 *Funding & Acknowledgments*

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

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

350 *Competing Interests*

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

352 *Author Contributions*

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

358 *Data Availability*

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

361 **References**

- 362 [1] G. ALLASIA, R. CAVORETTO, A. DE ROSSI, *Hermite-Birkhoff interpolation*
363 *on scattered data on the sphere and other manifolds*, Appl. Math. Comput.
364 **318** (2018), 35–50.
- 365 [2] S. AREFIAN, D. MIRZAEI, *A compact radial basis function partition of unity*
366 *method*, Comput. Math. Appl. **127** (2022), 1–11.
- 367 [3] I. BABUŠKA, J.M. MELENK, *The partition of unity method*, Int. J. Numer.
368 Meth. Eng. **40** (1997), 727–758.
- 369 [4] A. BERLINET, C. THOMAS-AGNAN, *Reproducing Kernel Hilbert Spaces in*
370 *Probability and Statistics*, Kluwer, Dordrecht, 2004.
- 371 [5] D.S. BERNSTEIN *Matrix Mathematics: Theory, Facts, and Formulas, 2nd*
372 *edn.*, Princeton University Press, Princeton, N.J., 2009.
- 373 [6] F. BLACK, M. SHOLES, *The pricing of options and corporate liabilities*, J.
374 Polit. Econ. **81** (1973), 637–654.
- 375 [7] L. BOTTOU, V. VAPNIK, *Local learning algorithms*, Neural Computation **4**
376 (1992), 888–900.
- 377 [8] M. BOZZINI, L. LENARDUZZI, M. ROSSINI, R. SCHABACK, *Interpolation*
378 *with variably scaled kernels*, IMA J. Numer. Anal. **35** (2015), 199–219.
- 379 [9] L. BREIMAN, *Random forests*, Mach. Learn. **45** (2001), 5–32.

- 380 [10] C. CAMPI, F. MARCHETTI, E. PERRACCHIONE, *Learning via variably*
381 *scaled kernels*, Adv. Comput. Math. **47** (2021), 51.
- 382 [11] R. CAVORETTO, A. DE ROSSI, *A meshless interpolation algorithm using*
383 *a cell-based searching procedure*, Comput. Math. Appl. **67** (2014), 1024–1038.
- 384 [12] R. CAVORETTO, A. DE ROSSI, *An adaptive residual sub-sampling algo-*
385 *rithm for kernel interpolation based on maximum likelihood estimations*, J.
386 Comput. Appl. Math. **418** (2023), 114658.
- 387 [13] R. CAVORETTO, A. DE ROSSI, *Error indicators and refinement strategies*
388 *for solving Poisson problems through a RBF partition of unity collocation*
389 *scheme*, Appl. Math. Comput. **369** (2020), 124824.
- 390 [14] R. CAVORETTO, A. DE ROSSI, F. DELL’ACCIO, F. DI TOMMASO, *An ef-*
391 *ficient trivariate algorithm for tetrahedral Shepard interpolation*, J. Sci. Com-
392 put. **82** (2020), 57.
- 393 [15] R. CAVORETTO, T. SCHNEIDER, P. ZULIAN, *OpenCL based parallel algo-*
394 *rithm for RBF-PUM interpolation*, J. Sci. Comput. **74** (2018), 267–289.
- 395 [16] J.P. CHILÈS, N. DESASSIS, *Fifty years of Kriging*. In: B. Daya Sagar
396 et al. (eds), Handbook of Mathematical Geosciences, Springer, Cham, 2018,
397 589–612.
- 398 [17] C. CORTES, V.N. VLADIMIR, *Support-vector networks*, Machine Learning
399 **20** (1995), 273–297.
- 400 [18] L. CSATÓ, M. OPPER, *Sparse On-Line Gaussian Processes*, Neural Com-
401 put. **14** (2002), 641–668.
- 402 [19] S. DE MARCHI, A. MARTÍNEZ, E. PERRACCHIONE, *Fast and stable ratio-*
403 *nal RBF-based partition of unity interpolation*, J. Comput. Appl. Math. **349**
404 (2019), 331–343.
- 405 [20] S. DE MARCHI, R. SCHABACK, H. WENDLAND, *Near-optimal data-*
406 *independent point locations for radial basis function interpolation*, Adv. Com-
407 put. Math. **23** (2005), 317–330.
- 408 [21] K.L. DU, M.N.S SWAMY, *Neural Networks and Statistical Learning*,
409 Springer, London, 2010.
- 410 [22] G.E. FASSHAUER, *Meshfree Approximations Methods with MATLAB*, World
411 Scientific, Singapore, 2007.
- 412 [23] G.E. FASSHAUER, M.J. MCCOURT, *Kernel-based Approximation Methods*
413 *Using MATLAB*, World Scientific, Singapore, 2015.
- 414 [24] M. FUHRY, L. REICHEL, *A new Tikhonov regularization method*, Numer.
415 Algorithms **59** (2012), 433–445.

- 416 [25] S. GUASTAVINO, F. BENVENUTO, *Convergence Rates of Spectral Regularization Methods: A Comparison between Ill-Posed Inverse Problems and*
417 *Statistical Kernel Learning*, SIAM J. Num. Anal. **58** (2020), 3504–3529.
- 419 [26] S. GUASTAVINO, F. BENVENUTO, *A consistent and numerically efficient*
420 *variable selection method for sparse Poisson regression with applications to*
421 *learning and signal recovery*, Stat. Comput. **29** (2019), 501–516.
- 422 [27] L. HARTMAN, O. HÖSSJER, *Fast Kriging of large datasets with Gaussian*
423 *Markov random fields*, Comput. Stat. Data Anal. **52** (2008), 2331–2349.
- 424 [28] T. JOACHIMS, C.N.J. YU, *Sparse kernel SVMs via cutting-plane training*,
425 Machine Learning **76** (2009), 179–193.
- 426 [29] D.G. KRIGE, *A statistical approach to some basic mine valuation problems*
427 *on the Witwatersrand*, J. Chem. Met. & Mining Soc., S. Africa **52** (1951),
428 119–139.
- 429 [30] E. LARSSON, V. SHCHERBAKOV, A. HERYUDONO, *A least squares radial*
430 *basis function partition of unity method for solving PDEs*, SIAM J. Sci. Com-
431 put. **39** (2017), A2538–A2563.
- 432 [31] N.D. LAWRENCE, *Gaussian process latent variable models for visualisation*
433 *of high dimensional data*, Adv. Neural. Inf. Proces. Syst. **16** (2004), 329–336.
- 434 [32] S. MAJI, A.C. BERG, J. MALIK, *Efficient classification for additive kernel*
435 *SVMs*, in IEEE PAMI, vol. 35, 2013, 66–77.
- 436 [33] F. MARCHETTI, E. PERRACCHIONE, *Local-to-Global Support Vector Ma-*
437 *chines (LGSVMs)*, Pattern Recognit. **132** (2022), 108920.
- 438 [34] P. MASSA, S. GARBARINO, F. BENVENUTO, *Approximation of discontinu-*
439 *ous inverse operators with neural networks*, Inverse Probl. **38** (2022), 105001.
- 440 [35] MATLAB R2021b and Statistics and Machine Learning Toolbox, The Math-
441 Works, Inc., Natick, Massachusetts, USA.
- 442 [36] B. MATÉRN, *Spatial Variation*, Lecture Notes in Statistics, Springer-
443 Verlag, vol. 36, 1986.
- 444 [37] A.K. MENON, *Large-scale support vector machines: Algorithms and theory*,
445 technical report, University of California San Diego (2009).
- 446 [38] D. MIRZAEI, *The direct radial basis function partition of unity (D-RBF-*
447 *PU) method for solving PDEs*, SIAM J. Sci. Comput. **43** (2021), A54–A83.
- 448 [39] A. NAISH-GUZMAN, S. HOLDEN, *The generalized FITC approximation* In:
449 J. Platt et al. (eds), Advances in neural information processing systems, vol.
450 4, 2008, 1057–1064.

- 451 [40] D. NGUYEN-TUONG, M. SEEGER, J. PETERS, *Model learning with local*
452 *Gaussian process regression*, Adv. Robot. **23** (2009), 2015–2034.
- 453 [41] D. RULLIÈRE, N. DURRANDE, F. BACHOC, C. CHEVALIER, *Nested kriging*
454 *predictions for datasets with a large number of observations*, Stat. Comput.
455 **28** (2018), 849–867.
- 456 [42] A. SAFDARI-VAIGHANI, A. HERYUDONO, E. LARSSON, *A radial basis func-*
457 *tion partition of unity collocation method for convection-diffusion equations*,
458 J. Sci. Comput. **64** (2015), 341–367.
- 459 [43] B. SCHÖLKOPF, A.J. SMOLA, *Learning with Kernels: Support Vector Ma-*
460 *chines, Regularization, Optimization, and Beyond*, MIT Press, Cambridge,
461 MA, USA, 2002.
- 462 [44] N. SEGATA, E. BLANZIERI, *Fast local support vector machines for large*
463 *datasets*, In: P. Perner P. (eds) Machine Learning and Data Mining in Pattern
464 Recognition. MLDM 2009. Lecture Notes in Computer Science, vol. 5632,
465 2009, 295–310.
- 466 [45] D. SHEPARD, *A two-dimensional interpolation function for irregularly*
467 *spaced data*, in: Proceedings of 23-rd National Conference, Brandon/Systems
468 Press, Princeton, 1968, 517–524.
- 469 [46] J. SHAWE-TAYLOR, N. CRISTIANINI, *Kernel Methods for Pattern Analysis*,
470 Cambridge Univ. Press, 2004.
- 471 [47] A.N. TIKHONOV, *Solution of incorrectly formulated problems and the reg-*
472 *ularization method*, Sov. Math. Dokl. **4** (1963), 1035–1038.
- 473 [48] V. TRESP, *A Bayesian Committee Machine*, Neural Comput. **12** (2000),
474 2719–2741.
- 475 [49] B. VAN STEIN, H. WANG, W. KOWALCZYK, M. EMMERICH, THOMAS
476 BÄCK, *Cluster-based Kriging approximation algorithms for complexity reduc-*
477 *tion*, Appl. Intell. **50** (2020), 778–791.
- 478 [50] G. WAHBA *Spline Models for Observational Data*, Society for Industrial
479 and Applied Mathematics, Philadelphia, PA, 1990.
- 480 [51] H. WENDLAND, *Scattered Data Approximation*, Cambridge Monogr. Appl.
481 Comput. Math., vol. 17, Cambridge Univ. Press, Cambridge, 2005.
- 482 [52] H. WENDLAND, *Fast evaluation of radial basis functions: Methods based*
483 *on partition of unity*, in: C.K. Chui et al. (eds), Approximation Theory X:
484 Wavelets, Splines, and Applications, Vanderbilt Univ. Press, Nashville, 2002,
485 473–483.
- 486 [53] H. WENDLAND, *Piecewise polynomial, positive definite and compactly sup-*
487 *ported radial functions of minimal degree*, Adv. Comput. Math. **4** (1995),
488 389–396.