Struct Multidisc Optim (2016) 53:935–952
DOI 10.1007/s00158-015-1395-9

REVIEW ARTICLE

Improving kriging surrogates of high-dimensional design models by Partial Least Squares dimension reduction

Mohamed Amine Bouhlel[1] · Nathalie Bartoli[2] · Abdelkader Otsmane[1] · Joseph Morlier[3]

Received: 20 June 2014 / Revised: 18 December 2015 / Accepted: 22 December 2015 / Published online: 14 January 2016

Abstract Engineering computer codes are often computationally expensive. To lighten this load, we exploit new covariance kernels to replace computationally expensive codes with surrogate models. For input spaces with large dimensions, using the kriging model in the standard way is computationally expensive because a large covariance matrix must be inverted several times to estimate the parameters of the model. We address this issue herein by constructing a covariance kernel that depends on only a few parameters. The new kernel is constructed based on information obtained from the Partial Least Squares method. Promising results are obtained for numerical examples with up to 100 dimensions, and significant computational gain is obtained while maintaining sufficient accuracy.

✉ Mohamed Amine Bouhlel
mohamed.bouhlel@onera.fr;
mohamed.amine.bouhlel@gmail.com

Nathalie Bartoli
nathalie.bartoli@onera.fr

Abdelkader Otsmane
abdelkader.otsmane@snecma.fr

Joseph Morlier
joseph.morlier@isae.fr

[1] SNECMA, Rond-point René Ravaud-Réau, 77550 Moissy-Cramayel, France

[2] ONERA, 2 Avenue Édouard Belin, 31055 Toulouse, France

[3] Université de Toulouse, CNRS, Institut Clément Ader, ISAE-SUPAERO, 10 Avenue Edouard Belin, 31055 Toulouse Cedex 4, France

Keywords Kriging · Partial Least Squares · Experiment design · Metamodels

Symbols and notation

Matrices and vectors are in bold type.

Symbol	Meaning
det	Determinant of a matrix
$\lvert\cdot\rvert$	Absolute value
$\mathbb{R}$	Set of real numbers
$\mathbb{R}^+$	Set of positive real numbers
n	Number of sampling points
d	Dimensions
h	Number of principal components retained
$\mathbf{x}$	$1 \times d$ vector
$\mathbf{x}_j$	j^{th} element of a vector $\mathbf{x}$
$\mathbf{X}$	$n \times d$ matrix containing sampling points
$\mathbf{y}$	$n \times 1$ vector containing simulation of $\mathbf{X}$
$\mathbf{x}^{(i)}$	i^{th} training point for $i = 1, \ldots, n$ (a $1 \times d$ vector)
$\mathbf{w}^{(l)}$	$d \times 1$ vector containing $\mathbf{X}$ weights given by the l^{th} PLS iteration for $l = 1, \ldots, h$
$\mathbf{X}^{(0)}$	$\mathbf{X}$
$\mathbf{X}^{(l-1)}$	Matrix containing residual of inner regression of $(l-1)^{\text{st}}$ PLS iteration for $l = 1, \ldots, h$
$k(\cdot, \cdot)$	Covariance function
$\mathcal{N}(\mathbf{0}, k(\cdot, \cdot))$	Distribution of a Gaussian process with mean function 0 and covariance function $k(\cdot, \cdot)$
$\mathbf{x}^t$	Superscript t denotes the transpose operation of the vector $\mathbf{x}$

1 Introduction and main contribution

In recent decades, because simulation models have striven to more accurately represent the true physics of phenomena, computational tools in engineering have become ever more complex and computationally expensive. To address this new challenge, a large number of input design variables, such as geometric representation, are often considered. Thus, to analyze the sensitivity of input design variables or to search for the best point of a physical objective under certain physical constraints (i.e., global optimization), a large number of computing iterations are required, which is impractical when using simulations in real time. This is the main reason that surrogate modeling techniques have been growing in popularity in recent years. Surrogate models, also called metamodels, are vital in this context and are widely used as substitutes for time-consuming high-fidelity models. They are mathematical tools that approximate coded simulations of a few well-chosen experiments that serve as models for the design of experiments. The main role of surrogate models is to describe the underlying physics of the phenomena in question. Different types of surrogate models can be found in the literature, such as regression, smoothing spline (Wahba and Craven 1978; Wahba 1990), neural networks (Haykin 1998), radial basis functions (Buhmann 2003) and Gaussian-process modeling (Rasmussen and Williams 2006).

In this article, we focus on the kriging model because it estimates the prediction error. This model is also referred to as the Gaussian-process model (Rasmussen and Williams 2006) and was presented first in geostatistics (see, e.g., Cressie 1988 or Goovaerts 1997) before being extended to computer experiments and machine learning (Schonlau 1998; Sasena 2002; Jones et al. 1998; Picheny et al. 2010). The kriging model has become increasingly popular due to its flexibility in accurately imitating the dynamics of computationally expensive simulations and its ability to estimate the error of the predictor. However, it suffers from some well-known drawbacks in high dimension, which may be due to multiple causes. For starters, the size of the covariance matrix of the kriging model may increase dramatically if the model requires a large number of sample points. As a result, inverting the covariance matrix is computationally expensive. The second drawback is the optimization of the subproblem, which involves estimating the hyper-parameters for the covariance matrix. This is a complex problem that requires inverting the covariance matrix several times. Some recent works have addressed the drawbacks of high-dimensional Gaussian processes (Hensman et al. 2013; Damianou and Lawrence 2013; Durrande et al. 2012) or the large-scale sampling of data (Sakata et al. 2004). One way to reduce CPU time when constructing a kriging model is to reduce the number of hyper-parameters, but this approach assumes that the kriging model exhibits the same dynamics in all directions (Mera 2007).

Thus, because estimating the kriging parameters can be time consuming, especially with dimensions as large as 100, we present herein a new method that combines the kriging model with the Partial Least Squares (PLS) technique to obtain a fast predictor. Like the method of principle components analysis (PCA), the PLS technique reduces dimension and reveals how inputs depend on outputs. PLS is used in this work because PCA only exposes dependencies between inputs. Information given by PLS is integrated in the covariance structure of the kriging model to reduce the number of hyper-parameters. The combination of kriging and PLS is abbreviated KPLS and allows us to build a fast kriging model because it requires fewer hyper-parameters in its covariance function; all without eliminating any input variables from the original problem. In general, the number of kriging parameters is equal to the number of dimension which is reduced to at maximum 4 parameters with our approach.

The KPLS methods is used for many academic and industrial verifications, and promising results have been obtained for problems with up to 100 dimensions. The cases used in this paper do not exceed 100 input variables, which should be quite sufficient for most engineering problems. Problems with more than 100 inputs may lead to memory difficulties with the toolbox Scikit-learn (version 0.14), on which the KPLS method is based.

This paper is organized as follows: Section 2 summarizes the theoretical basis of the universal kriging model, recalling the key equations. The proposed KPLS model is then described in detail in Section 3 by using the kriging equations. Section 4 compares and analyzes the results of the KPLS model with those of the kriging model when applied to classic analytical examples and some complex engineering examples. Finally, Section 5 concludes and gives some perspectives.

2 Universal kriging model

To understand the mathematics of the proposed methods, we first review the kriging equations. The objective is to introduce the notation and to briefly describe the theory behind the kriging model. Assume that we have evaluated a cost deterministic function at n points $\mathbf{x}^{(i)}$ $(i = 1, \ldots, n)$ with

$$\mathbf{x}^{(i)} = \left[\mathbf{x}_1^{(i)}, \ldots, \mathbf{x}_d^{(i)}\right] \in B \subset \mathbb{R}^d,$$

and we denote by $\mathbf{X}$ the matrix $[\mathbf{x}^{(1)t}, \ldots, \mathbf{x}^{(n)t}]^t$. For simplicity, B is considered to be a hypercube expressed by

the product between intervals of each direction space, i.e., $B = \prod_{j=1}^{d}[a_j, b_j]$, where $a_j, b_j \in \mathbb{R}$ with $a_j \leq b_j$ for $j = 1, \ldots, d$. Simulating these n inputs gives the outputs $\mathbf{y} = [y^{(1)}, \ldots, y^{(n)}]^t$ with $y^{(i)} = y(\mathbf{x}^{(i)})$ for $i = 1, \ldots, n$. We use $\hat{y}(\mathbf{x})$ to denote the prediction of the true function $y(\mathbf{x})$ which is considered as a realization of a stochastic process $Y(\mathbf{x})$ for all $\mathbf{x} \in B$. For the universal kriging model (Roustant et al. 2012; Picheny et al. 2010), Y is written as

$$Y(\mathbf{x}) = \sum_{j=1}^{m} \beta_j f_j(\mathbf{x}) + Z(\mathbf{x}), \tag{1}$$

where, for $j = 1, \ldots, m$, f_j is a known independent basis function, $\beta_j \in \mathbb{R}$ is an unknown parameter, and Z is a random variable defined by $Z(\mathbf{x}) \sim \mathcal{N}(0, k)$, with k being a stationary covariance function, also called a covariance kernel. The kernel function k can be written as

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 r(\mathbf{x}, \mathbf{x}') = \sigma^2 r_{\mathbf{x}\mathbf{x}'} \; \forall \, \mathbf{x}, \mathbf{x}' \in B, \tag{2}$$

where σ^2 is the process variance and $r_{\mathbf{x}\mathbf{x}'}$ is the correlation function between $\mathbf{x}$ and $\mathbf{x}'$. However, the correlation function r depends on some hyper-parameters θ and it is considered to be known. We also denote the $n \times 1$ vector as $\mathbf{r}_{\mathbf{x}\mathbf{X}} = [r_{\mathbf{x}\mathbf{x}^{(1)}}, \ldots, r_{\mathbf{x}\mathbf{x}^{(n)}}]^t$ and the $n \times n$ covariance matrix as $\mathbf{R} = [\mathbf{r}_{\mathbf{x}^{(1)}\mathbf{X}}, \ldots, \mathbf{r}_{\mathbf{x}^{(n)}\mathbf{X}}]$.

2.1 Derivation of prediction formula

Under the hypothesis above, the best linear unbiased predictor for $y(\mathbf{x})$, given the observations $\mathbf{y}$, is

$$\hat{y}(\mathbf{x}) = \mathbf{f}(\mathbf{x})^t \hat{\beta} + \mathbf{r}_{\mathbf{x}\mathbf{X}}^t \mathbf{R}^{-1} \left(\mathbf{y} - \mathbf{F}\hat{\beta} \right), \tag{3}$$

where $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})]^t$ is the $m \times 1$ vector of basis functions, $\mathbf{F} = \left[\mathbf{f}(\mathbf{x}^{(1)}), \ldots, \mathbf{f}(\mathbf{x}^{(n)})\right]^t$ is the $n \times m$ matrix, and $\hat{\beta}$ is the vector of generalized least-square estimates of $\beta = [\beta_1, \ldots, \beta_m]^t$, which is given by

$$\hat{\beta} = \begin{bmatrix} \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_m \end{bmatrix} = \left(\mathbf{F}^t \mathbf{R}^{-1} \mathbf{F} \right)^{-1} \mathbf{F}^t \mathbf{R}^{-1} \mathbf{y}. \tag{4}$$

Moreover, the universal kriging model provides an estimate of the variance of the prediction, which is given by

$$s^2(\mathbf{x}) = \hat{\sigma}^2 \left(1 - \mathbf{r}_{\mathbf{x}\mathbf{X}}^t \mathbf{R}^{-1} \mathbf{r}_{\mathbf{x}\mathbf{X}} \right), \tag{5}$$

with

$$\hat{\sigma}^2 = \frac{1}{n} \left(\mathbf{y} - \mathbf{F}\hat{\beta} \right)^t \mathbf{R}^{-1} \left(\mathbf{y} - \mathbf{F}\hat{\beta} \right). \tag{6}$$

For more details of the derivation of the prediction formula, see, for instance, Sasena (2002) or Schonlau (1998). The theory of the proposed method has been expressed in the same way as for the universal kriging model. The numerical examples in Section 4 use the ordinary kriging model, which is a special case of the universal model, but with $\mathbf{f}(\mathbf{x}) = \{1\}$ (and $m = 1$). For the ordinary kriging model, (3), (4), and (6) are then replaced by the equations given in Appendix A.

Note that the assumption of known covariance with known hyper-parameters θ is unrealistic in reality. For this reason, the covariance function is typically chosen from among a parametric family of kernels. Table 12 in Appendix B gives some examples of typical stationary kernels. The number of hyper-parameters required for the estimate is typically greater than (or equal to) the number of input variables. In this work, we use in the following a Gaussian exponential kernel:

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \prod_{i=1}^{d} \exp\left(-\theta_i \left(\mathbf{x}_i - \mathbf{x}'_i \right)^2 \right) \; \forall \, \theta_i \in \mathbb{R}^+.$$

By applying some elementary operations to existing kernels, we can construct new kernels. In this work, we use the property that the tensor product of covariances is a covariance kernel in the product space. More details are available in Rasmussen and Williams (2006), Durrande (2011), Bishop (2007), Liem and Martins (2014).

2.2 Estimation of hyper-parameters θ

The key point of the kriging approximation is how it estimates the hyper-parameters θ, so its main steps are recalled here, along with some mathematical details.

One of the major challenges when building a kriging model is the complexity and difficulty of estimating the hyper-parameters θ, in particular when dealing with problems with many dimensions or with a large number of sampling points. In fact, using (3) to make a kriging prediction requires inverting an $n \times n$ matrix, which typically has cost of $\mathcal{O}\left(n^3\right)$, where n is the number of sampling points (Braham et al. 2014). The hyper-parameters are estimated using maximum likelihood (ML) or cross validation (CV), which are based on observations. Bachoc compared the ML and CV techniques (Bachoc 2013) and concluded that, in most cases studied, the CV variance is larger. The ML method is widely used to estimate the hyper-parameters θ; it is also used in this paper. In practice, the following log-ML estimate is often used:

$$\begin{aligned} \log\text{-ML}(\theta) \; = \; & -\frac{1}{2} \Big[n \ln(2\pi\sigma^2) + \ln(\det \mathbf{R}(\theta)) \\ & + (\mathbf{y} - \mathbf{F}\beta)^t \mathbf{R}(\theta)^{-1} (\mathbf{y} - \mathbf{F}\beta)/\sigma^2 \Big]. \end{aligned} \tag{7}$$

Inserting $\hat{\beta}$ and $\hat{\sigma}^2$ given by (4) and (6), respectively, into the expression (7), we get the following so-called

concentrated likelihood function, which depends only on the hyper-parameters θ:

$$\begin{aligned}\text{log-ML}(\theta) &= -\frac{1}{2}[n \ln \hat{\sigma}^2 + \ln \det \mathbf{R}(\theta)] \\ &= -\frac{1}{2}\left[n \ln \left(\frac{1}{n}(\mathbf{y} - \mathbf{F}(\mathbf{F}^t\mathbf{R}^{-1}\mathbf{F})^{-1}\mathbf{F}^t\mathbf{R}^{-1}\mathbf{y})^t \right.\right. \\ &\quad \left. \times \mathbf{R}^{-1}(\mathbf{y} - \mathbf{F}(\mathbf{F}^t\mathbf{R}^{-1}\mathbf{F})^{-1}\mathbf{F}^t\mathbf{R}^{-1}\mathbf{y})\right) \\ &\quad \left. + \ln \det \mathbf{R}\right]. \end{aligned} \tag{8}$$

To facilitate reading, $\mathbf{R}(\theta)$ has been replaced by $\mathbf{R}$ in the last line of (8).

Maximizing (8) is very computationally expensive for high dimensions and when using a large number of sample points because the $(n \times n)$ matrix $\mathbf{R}(\theta)$ in (8) must be inverted. The maximization problem is often solved using genetic algorithms (see Forrester and Sobester 2008 for more details). In this work, we use the derivative-free optimization algorithm COBYLA that was developed by Powell (1994). COBYLA is a sequential trust-region algorithm that uses linear approximations for the objective and constraint functions.

Figure 1 recalls the principal stages of building a kriging model, and each step is briefly outlined below:

1. The user must provide the initial design of experiments $(\mathbf{X}, \mathbf{y})$ and the parametric family of the covariance function k.
2. To derive the prediction formula, the kriging algorithm assumes that all parameters of k are known.

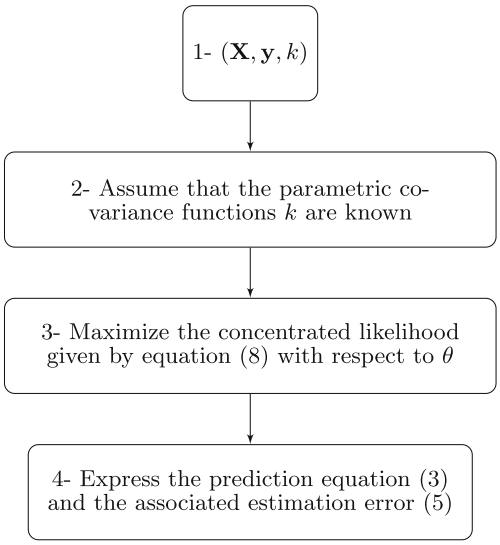


Fig. 1 The main steps for building an ordinary kriging model

3. Under the hypothesis of the kriging algorithm, we estimate hyper-parameters θ from the concentrated likelihood function given by (8) and by using the COBYLA algorithm.
4. Finally, we calculate the prediction (3) and the associated estimation error (5) after estimating all hyper-parameters of the kriging model.

3 Kriging model combined with Partial Least Squares

As explained above, maximizing the concentrated likelihood (8) can be time consuming when the number of covariance parameters is large, which typically occurs in large dimension. Solving this problem can be accelerated by combing the PLS method and the kriging model. The θ parameters from the kriging model represent the range in any spatial direction. Assuming, for instance, that certain values are less significant for the response, then the corresponding θ_i $(i = 1, \ldots, d)$ will be very small compared to the other θ parameters. The PLS method is a well-known tool for high-dimensional problems and consists of maximizing the variance by projecting onto smaller dimensions while monitoring the correlation between input variables and the output variable. In this way, the PLS method reveals the contribution of all variables—the idea being to use this information to scale the θ parameters.

In this section we propose a new method that can be used to build an efficient kriging model by using the information extracted from the PLS stage. The main steps for this construction are as follows:

1. Use PLS to define weight parameters.
2. To reduce the number of hyper-parameters, define a new covariance kernel by using the PLS weights.
3. Optimize the parameters.

The key mathematical details of this construction are explained in the following.

3.1 Linear transformation of covariance kernels

Let $\mathbf{x}$ be a vector space over the hypercube B. We define a linear map given by

$$\begin{aligned} F : B &\longrightarrow B', \\ \mathbf{x} &\longmapsto [\alpha_1 \mathbf{x}_1, \ldots, \alpha_d \mathbf{x}_d], \end{aligned} \tag{9}$$

where $\alpha_1, \ldots, \alpha_d \in \mathbb{R}$ and B' is a hypercube included in $\mathbb{R}^d$ (B' can be different from B). Let k be an isotropic covariance kernel with $k : B' \times B' \to \mathbb{R}$. Since k is isotropic, the covariance kernel $k(F(\cdot), F(\cdot))$

depends on a single parameter, which must be estimated. However, if $\alpha_1, \ldots, \alpha_d$ are well chosen, then we can use $k(F(\cdot), F(\cdot))$. In this case, the linear transformation F allows us to approach the isotropic case (Zimmerman and Homer 1991). In the present work, we choose $\alpha_1, \ldots, \alpha_d$ based on information extracted from the PLS technique.

3.2 Partial Least Squares

The PLS method is a statistical method that finds a linear relationship between input variables and the output variable by projecting input variables onto a new space formed by newly chosen variables, called principal components (or latent variables), which are linear combinations of the input variables. This approach is particularly useful when the original data are characterized by a large number of highly collinear variables measured on a small number of samples. Below, we briefly describe how the method works. For now, suffice it to say that only the weighting coefficients are central to understanding the new KPLS approach. For more details on the PLS method, please see Helland (1988), Frank and Friedman (1993), Alberto and González (2012).

The PLS method is designed to search out the best multidimensional direction in $\mathbf{X}$ space that explains the characteristics of the output $\mathbf{y}$. After centering and scaling the $(n \times d)$-sample matrix $\mathbf{X}$ and the response vector $\mathbf{y}$, the first principal component $\mathbf{t}^{(1)}$ is computed by seeking the best direction $\mathbf{w}^{(1)}$ that maximizes the squared covariance between $\mathbf{t}^{(1)} = \mathbf{X}\mathbf{w}^{(1)}$ and $\mathbf{y}$:

$$\mathbf{w}^{(1)} = \begin{cases} \arg\max\limits_{\mathbf{w}} \ \mathbf{w}^t \mathbf{X}^t \mathbf{y}\mathbf{y}^t \mathbf{X}\mathbf{w} \\ \text{such that } \mathbf{w}^t\mathbf{w} = 1. \end{cases} \tag{10}$$

The optimization problem (10) is maximized when $\mathbf{w}^{(1)}$ is the eigenvector of the matrix $\mathbf{X}^t\mathbf{y}\mathbf{y}^t\mathbf{X}$ corresponding to the eigenvalue with the largest absolute value; the vector $\mathbf{w}^{(1)}$ contains the $\mathbf{X}$ weights of the first component. The largest eigenvalue of problem (10) can be estimated by the power iteration method introduced by Lanczos (1950).

Next, the residual matrix from $\mathbf{X} = \mathbf{X}^{(0)}$ space and from $\mathbf{y} = \mathbf{y}^{(0)}$ are calculated; these are denoted $\mathbf{X}^{(1)}$ and $\mathbf{y}^{(1)}$, respectively:

$$\begin{aligned} \mathbf{X}^{(1)} &= \mathbf{X}^{(0)} - \mathbf{t}^{(1)}\mathbf{p}^{(1)}, \\ \mathbf{y}^{(1)} &= \mathbf{y}^{(0)} - c_1\mathbf{t}^{(1)}, \end{aligned} \tag{11}$$

where $\mathbf{p}^{(1)}$ (a $1 \times d$ vector) contains the regression coefficients of the local regression of $\mathbf{X}$ onto the first principal component $\mathbf{t}^{(1)}$, and c_1 is the regression coefficient of the local regression of $\mathbf{y}$ onto the first principal component $\mathbf{t}^{(1)}$.

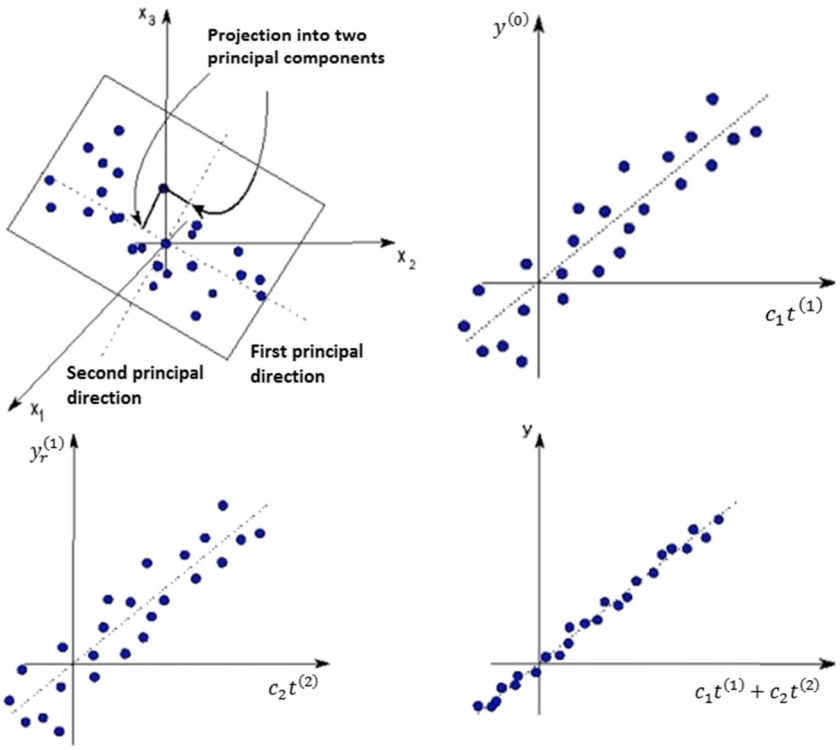


Fig. 2 *Upper left* shows construction of two principal components in $\mathbf{X}$ space. *Upper right* shows prediction of $y^{(0)}$. *Bottom left* shows prediction of $y^{(1)}$. *Bottom right* shows final prediction of y

The system (11) is the local regression of $\mathbf{X}$ and $\mathbf{y}$ onto the first principal component.

Next, the second principal component, which is orthogonal to the first, can be sequentially computed by replacing $\mathbf{X}$ by $\mathbf{X}^{(1)}$ and $\mathbf{y}$ by $\mathbf{y}^{(1)}$ to solve the system (10). The same approach is used to iteratively compute the other principal components. To illustrate this process, a simple three-dimensional (3D) example with two principal components is given in Fig. 2. In the following, we use h to denote the number of principal components retained.

The principal components represent the new coordinate system obtained upon rotating the original system with axes, $\mathbf{x}_1, \ldots, \mathbf{x}_d$ (Alberto and González 2012). For $l = 1, \ldots, h$, $\mathbf{t}^{(l)}$ can be written as

$$\mathbf{t}^{(l)} = \mathbf{X}^{(l-1)}\mathbf{w}^{(l)} = \mathbf{X}\mathbf{w}_*^{(l)}. \tag{12}$$

This important relationship is used for coding the method. The following matrix $\mathbf{W}_* = [\mathbf{w}_*^{(1)}, \ldots, .\mathbf{w}_*^{(h)}]$ is obtained by Manne (1987):

$$\mathbf{W}_* = \mathbf{W}\left(\mathbf{P}^t\mathbf{W}\right)^{-1},$$

where $\mathbf{W} = [\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(h)}]$ and $\mathbf{P} = [\mathbf{p}^{(1)^t}, \ldots, \mathbf{p}^{(h)^t}]$.

The vector $\mathbf{w}^{(l)}$ corresponds to the principal direction in X space that maximizes the covariance of $\mathbf{X}^{(l-1)^t}\mathbf{y}^{(l-1)}\mathbf{y}^{(l-1)^t}\mathbf{X}^{(l-1)}$. If $h = d$, the matrix $\mathbf{W}_* = [\mathbf{w}_*^{(1)}, \ldots, \mathbf{w}_*^{(d)}]$ rotates the coordinate space $(\mathbf{x}_1, \ldots, \mathbf{x}_d)$ to the new coordinate space $(t^{(1)}, \ldots, t^{(d)})$, which follow the principal directions $\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(d)}$.

As mentioned in the introduction, the PLS method is chosen instead of the PCA method because the PLS method shows how the output variable depends on the input variables, whereas the PCA method focuses only on how the input variables depend on each other. In fact, the hyper-parameters θ for the kriging model depend on how each input variable affects the output variable.

3.3 Construction of new kernels for KPLS models

Let B be a hypercube included in $\mathbb{R}^d$. As seen in the previous section, the vector $\mathbf{w}_*^{(1)}$ is used to build the first principal component $\mathbf{t}^{(1)} = \mathbf{X}\mathbf{w}_*^{(1)}$, where covariance between $\mathbf{t}^{(1)}$ and $\mathbf{y}$ is maximized. The scalars $\mathbf{w}_{*1}^{(1)}, \ldots, \mathbf{w}_{*d}^{(1)}$ can then be interpreted as measuring the importance of $\mathbf{x}_1, \ldots, \mathbf{x}_d$, respectively, for constructing the first principal component where its correlation with the output y is maximized. However, we know that the hyper-parameters $\theta_1, \ldots, \theta_d$ (see Table 12 in Appendix B) can be interpreted as measuring how strongly the variables $\mathbf{x}_1, \ldots, \mathbf{x}_d$, respectively, affect the output y. Thus, we define a new kernel $k_{kpls1} : B \times B \rightarrow \mathbb{R}$ given by $k_1(F_1(\cdot), F_1(\cdot))$ with $k_1 : B \times B \rightarrow \mathbb{R}$ being an isotropic stationary kernel and

$$\begin{aligned} F_1 : B &\longrightarrow B, \\ \mathbf{x} &\longmapsto \left[\mathbf{w}_{*1}^{(1)}\mathbf{x}_1, \ldots, \mathbf{w}_{*d}^{(1)}\mathbf{x}_d\right]. \end{aligned} \tag{13}$$

F_1 goes from B to B because it only works for the new coordinate system obtained by rotating the original coordinate axes, $\mathbf{x}_1, \ldots, \mathbf{x}_d$. Through the first component $\mathbf{t}^{(1)}$, the elements of the vector $\mathbf{w}_*^{(1)}$ reflect how $\mathbf{x}$ depends on y. However, such information is generally insufficient, so the elements of the vector $\mathbf{w}_*^{(1)}$ are supplemented by the information given by the other principal components $\mathbf{t}^{(2)}, \ldots, \mathbf{t}^{(h)}$. Thus, we build a new kernel $k_{kpls1:h}$ sequentially by using the tensor product of all kernels k_{kplsl}, which accounts for all this information in only a single covariance kernel:

$$k_{kpls1:h}(\mathbf{x}, \mathbf{x}') = \prod_{l=1}^{h} k_l(F_l(\mathbf{x}), F_l(\mathbf{x}')), \tag{14}$$

with $k_l : B \times B \rightarrow \mathbb{R}$ and

$$\begin{aligned} F_l : B &\longrightarrow B \\ \mathbf{x} &\longmapsto \left[\mathbf{w}_{*1}^{(l)}\mathbf{x}_1, \ldots, \mathbf{w}_{*d}^{(l)}\mathbf{x}_d\right]. \end{aligned} \tag{15}$$

If we consider the Gaussian kernel applied with this proposed approach, we get

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \prod_{l=1}^{h}\prod_{i=1}^{d} \exp\left[-\theta_l\left(\mathbf{w}_{*i}^{(l)}\mathbf{x}_i - \mathbf{w}_{*i}^{(l)}\mathbf{x}'_i\right)^2\right], \quad \forall\, \theta_l \in [0, +\infty[.$$

Table 13 in Appendix B presents new KPLS kernels based on examples from Table 12 (also in Appendix B) that contain fewer hyper-parameters because $h \ll d$. The number of principal components is fixed by the following leave-one-out cross-validation method:

(i) We build KPLS models based on $h = 1$, then $h = 2, \ldots$ principal components.
(ii) We choose the number of components that minimizes the leave-one-out cross-validation error.

The flowchart given in Fig. 3 shows how the information flows through the algorithm, from sample data, PLS algorithm, kriging hyper-parameters, to final predictor. With the same definitions and equations, almost all the steps for constructing the KPLS model are similar to the original steps for constructing the ordinary kriging model. The exception is the third step, which is highlighted in the solid-red box in Fig. 3. This step uses the PLS algorithm to define the new parametric kernel $k_{kpls1:h}$ as follows:

a. initialize the PLS algorithm with $l = 1$;
b. if $l \neq 1$, compute the residual of $\mathbf{X}^{(l-1)}$ and $\mathbf{y}^{(l-1)}$ by using system (11);

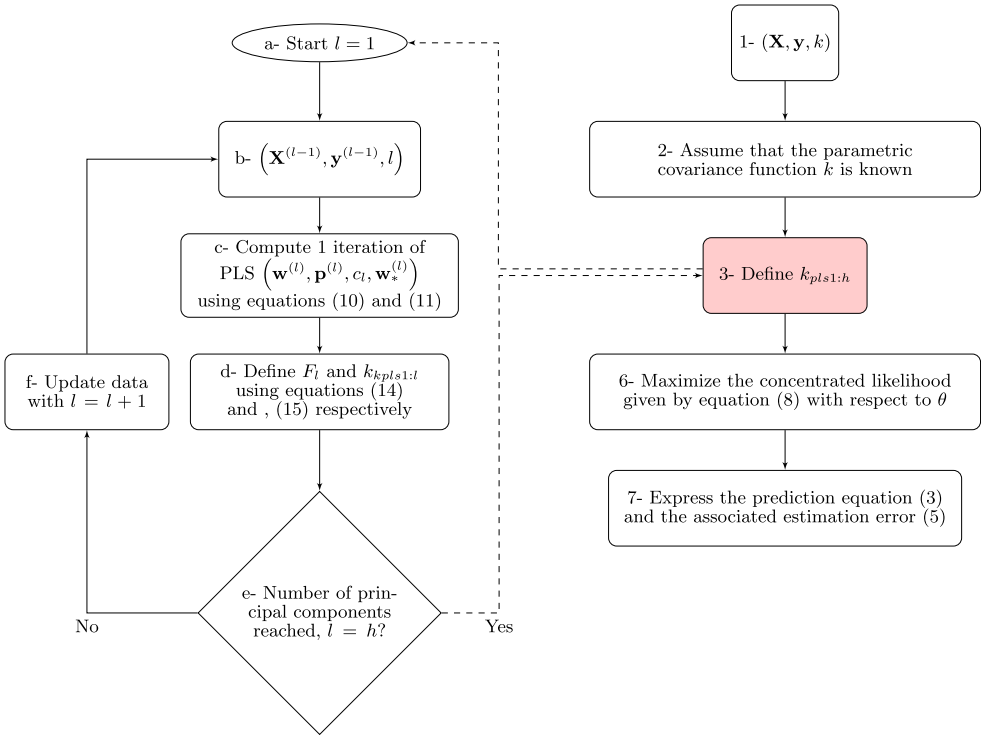


Fig. 3 Main steps for constructing KPLS model. The *solid-red box* (step 3 relative to PLS) is what differentiates this approach from the ordinary kriging approach

c. compute X weights for iteration l;
d. define a new kernel $k_{kpls1:h}$ by using (14);
e. if the number of iterations is reached, return to step 3, otherwise continue;
f. update data considering $l = l + 1$.

Note that, if kernels k_l are separable at this point, the new kernel given by (14) is also separable. In particular, if all kernels k_l are of the exponential type (e.g., all Gaussian exponentials), the new kernel given by (14) will be the same type as k_l. The proof is given in Appendix C.

4 Numerical examples

We now present a few analytical and engineering examples to verify the proper functioning of the proposed method. The ordinary kriging model with a Gaussian kernel provides the benchmark against which the results of the proposed combined approach are compared. The Python toolbox Scikit-learn v.014 (Pedregosa et al. 2011) is used to implement these numerical tests. This toolbox provides hyper-parameters for the ordinary kriging. The computations were done on an Intel® Celeron® CPU 900 2.20 GHz desktop PC. For the proposed method, we combined an ordinary kriging model with a Gaussian kernel with the PLS method with one to three principal components.

4.1 Analytical examples

We use two academic functions and vary the characteristics of these test problems to cover most of the difficulties faced in the field of substitution models. The first function is $g07$ (Michalewicz and Schoenauer 1996) with 10 dimensions,

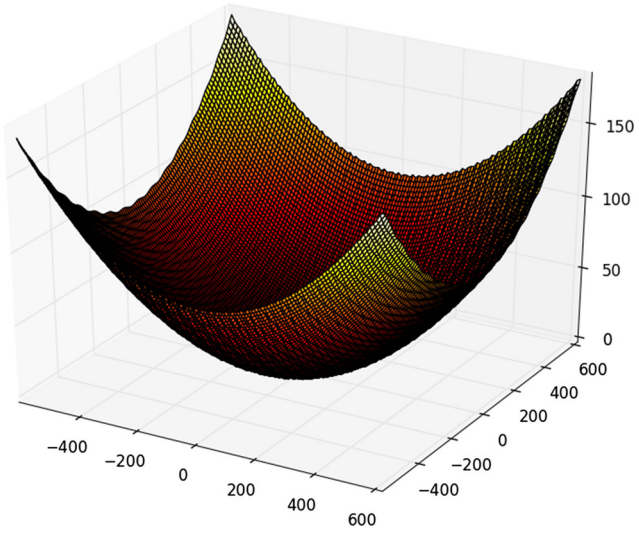


Fig. 4 A two-dimensional Griewank function over the interval $[-600, 600]$

Table 1 Number of data points used for latin hypercube design for the Griewank test function

$d = 2$	$d = 5$	$d = 7$	$d = 10$	$d = 20$	$d = 60$
$n = 70$	$n = 100$	$n = 200$	$n = 300$	$n = 400$	$n = 800$

which is close to what is required by industry in terms of dimensions,

$$\begin{aligned} y_{g07}(\mathbf{x}) = {} & x_1^2 + x_2^2 + x_1 x_2 - 14x_1 - 16x_2 \\ & + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 \\ & + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 \\ & + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45, \\ & -10 \leq x_i \leq 10, \ \text{for} i = 1, \ldots, 10. \end{aligned}$$

For this function, we use experiments based on a latin hypercube design with 100 data points to fit models.

The second function is the Griewank function (Regis and Shoemaker 2013), which is used because of its complexity, as illustrated in Fig. 4 for the two-dimensional (2D) case. The function is

$$\begin{aligned} y_{\text{Griewank}}(\mathbf{x}) = {} & \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \\ & -600 \leq x_i \leq 600, \ \text{for } i = 1, \ldots, d. \end{aligned}$$

Two types of experiments are done with this function. The first is defined over the interval $[-600, 600]$ and has varying dimensions (2, 5, 7, 10, 20, 60). This experiment serves to verify the effectiveness of the proposed approach in both low and high dimensions. It is based on the latin hypercube design and uses n data points to fit models, as mentioned in Table 1.

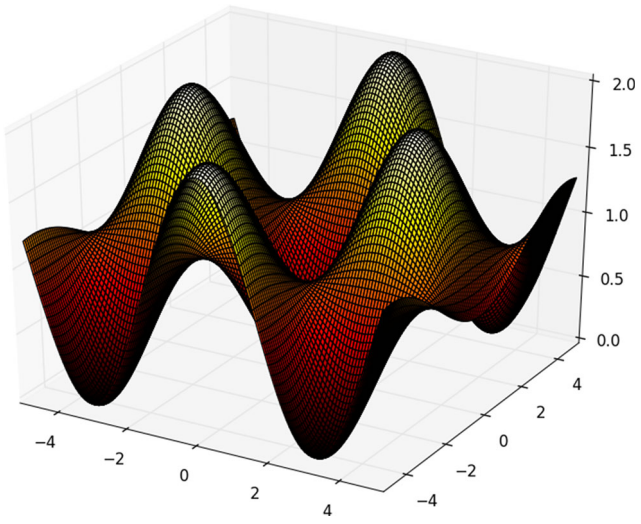


Fig. 5 A two-dimensional Griewank function over the interval $[-5, 5]$

Table 2 Results for $g07$ function in ten dimensions with 100-point latin hypercube

Surrogate	Error (%)	CPU time
Ordinary kriging	0.013	5.14 s
KPLS (1 component)	0.014	0.11 s
KPLS (2 components)	0.0015	0.43 s
KPLS (3 components)	0.0008	0.44 s

The second type of experiment is defined over the interval $[-5, 5]$, where the Griewank function is more complex than for the first type of experiment (cf. Figs. 4 and 5). Over this reduced interval, experiments are done with 20 and 40 dimensions (20D and 40D) and with 50, 100, 200, and 300 sampling points. To analyse the robustness of the method, ten experiments, each with a different latin hypercube design, are used for this case.

To compare the three approaches (i.e., the g07 function and the Griewank function of the intervals $[-600, 600]$ and $[-5, 5]$), 5000 random points are computed and the results are stored in a database. The following relative error is used to compare the performance of the ordinary kriging model with the KPLS model:

$$\text{Error} = \frac{||\hat{\mathbf{Y}} - \mathbf{Y}||_2}{||\mathbf{Y}||_2} 100, \tag{16}$$

where $||\cdot||_2$ represents the usual L^2 norm, and $\hat{\mathbf{Y}}$ and $\mathbf{Y}$ are the vectors containing the prediction and the real values of random points, respectively. The CPU time required to fit models is also noted ("h" refers to hours, "min" refers to minutes, and "s" refers to seconds).

4.1.1 Comparison with g07 function

The results listed in Table 2 show that the proposed KPLS surrogate model is more accurate than the ordinary kriging model when more than one component is used. Using just one component gives almost the same accuracy as the ordinary kriging model. In this case, only a single θ hyper-parameter from the space correlation needs be estimated compared to ten θ hyper-parameters for the

Table 3 Griewank function in two dimensions with 70-point latin hypercube over the interval $[-600, 600]$

Surrogate	Error (%)	CPU time
Ordinary kriging	5.50	0.09 s
KPLS (1 component)	7.23	0.04 s
KPLS (2 components)	5.50	0.10 s

Table 4 Griewank function in five dimensions with 100-point latin hypercube over the interval [−600, 600]

Surrogate	Error (%)	CPU time
Ordinary kriging	0.605	0.55 s
KPLS (1 component)	0.635	0.12 s
KPLS (2 components)	0.621	0.31 s
KPLS (3 components)	0.623	0.51 s

Table 6 Griewank function in ten dimensions with 300-point latin hypercube over the interval [−600, 600]

Surrogate	Error (%)	CPU time
Ordinary kriging	0.052	21 s
KPLS (1 component)	0.033	0.6 s
KPLS (2 components)	0.035	2.41 s
KPLS (3 components)	0.034	3.58 s

ordinary kriging model. Increasing the number of components improves the accuracy of the KPLS surrogate model. Whereas the PLS method treats only linearly related input and output variables, this example shows that the KPLS model can treat nonlinear problems. This result is not contradictory because (23) shows that the KPLS model is equivalent to the kriging model with specific hyper-parameters.

4.1.2 Comparison with complex Griewank function over interval [−600, 600]

Table 3 compares the ordinary kriging model and the KPLS model in two dimensions.

If two components are used for the KPLS, we expect to obtain the same accuracy and time cost for the two approaches because the difference between the two models consists only of a transformation of the search-space coordinates when a Gaussian kernel is used (the space in which the θ hyper-parameters exist). In this case, the KPLS model with only one component degrades the accuracy of the solution.

Tables 4, 5, and 6, show the results for 5, 7, and 10 dimensions, respectively.

Varying the number of principal components does not significantly affect the accuracy of the model. The gain in computation time does not appear upon increasing the number of principal components: the computation time is reduced when we use the KPLS model. Upon increasing the number of principal components, the CPU time for constructing the KPLS model increases but still remains lower than for ordinary kriging. For these three examples, the combined approach with only one PLS component offers sufficient accuracy with a CPU time reduced 35-fold for 10 dimensions (i.e., 21 s for the ordinary kriging model and 0.6 s for the combined model).

In the 20-dimension (20D) example (Table 7), using KPLS with only one principal component leads to a poor relative error (10.15 %) compared with other models. In this case, two principal components are required to build the combined model. The CPU time remains less than that for the ordinary kriging model (11.7 s vs 107 s).

The results in Table 8 for the KPLS model with 60 dimensions (60D) show that this model is faster than the ordinary kriging model. Compared with the kriging model, the CPU time is reduced 42-fold when one principal component is used and over 17-fold when three principal components are used.

Thus, for the Griewank function over the interval [−600, 600] and at the highest dimensions, the majority of the results obtained for the analytical examples are better when using the KPLS model than when using the ordinary kriging model. The proposed method thus appears interesting, particularly in terms of saving CPU time while maintaining good accuracy.

4.1.3 Comparison with complex Griewank function over interval [−5, 5]

As shown in Fig. 4, the Griewank function looks like a parabolic function. This is because, over the interval

Table 5 Griewank function in seven dimensions with 200-point latin hypercube over the interval [−600, 600]

Surrogate	Error (%)	CPU time
Ordinary kriging	0.138	3.09 s
KPLS (1 component)	0.141	0.25 s
KPLS (2 components)	0.138	0.52 s
KPLS (3 components)	0.141	0.94 s

Table 7 Griewank function in 20 dimensions with 400-point latin hypercube over the interval [−600, 600]

Surrogate	Error (%)	CPU time
Ordinary kriging	0.35	107 s
KPLS (1 component)	10.15	1.16 s
KPLS (2 components)	0.003	11.7 s
KPLS (3 components)	0.002	16.23 s

Table 8 Griewank function in 60 dimensions with 800-point latin hypercube over the interval $[-600, 600]$

Surrogate	Error (%)	CPU time
Ordinary kriging	11.47	293 s
KPLS (1 component)	7.4	6.88 s
KPLS (2 components)	6.04	12.57 s
KPLS (3 components)	5.23	16.82 s

$[-600, 600]$, the cosine part of the Griewank function does not contribute significantly compared with the sum of $x_i^2/4000$. This is especially true given that the cosine part is a product of factors each of which is less than unity. If we reduce the interval from $[-600, 600]$ to $[-5, 5]$, we can see why the Griewank function is widely used as a multimodal test function with a very rugged landscape and a large number of local optima (see Fig. 5). Compared with the interval $[-600, 600]$, the opposite happens for the interval $[-5, 5]$: the "cosine part" dominates; at least for moderate dimensions where the product contains few factors. For this case, which seems very difficult, we consider 20 and 60 input variables. For each problem, ten experiments based on the latin hypercube design are built with 50, 100, 200, and 300 sampling points. The mean and the standard error are given in Tables 14 and 15 in Appendix D. To better visualize the results, boxplots are used to show CPU time and the relative error RE given by Figs. 7, 8, 9, 10 and 11 in Appendix D.

For 20 input variables and 50 sampling points, the KPLS model gives a more accurate solution than the ordinary kriging model, as shown in Fig. 7a. The rate of improvement with respect to the number of sampling points is less for the KPLS model than for the kriging model (cf. Fig. 7b–d). Nevertheless, the results shown in Fig. 8 indicate that the KPLS model leads to an important reduction in CPU time for the various number of sampling points.

Similar results occur for the 60D Griewank function (Fig. 9). The mean RE obtained with the ordinary kriging model improves from approximately 1.39 to 0.65 % upon increasing the number of sampling points from 50 to 300 (cf. Fig. 9a, d). However, a very important reduction in CPU time is obtained, as shown in Fig. 10. The CPU time required for the KPLS model is hardly visible because it is much, much less than that required by the ordinary kriging model. We thus show in Fig. 11 the CPU time required by the KPLS model alone to show the different CPU times required for the various configurations (KPLS1, KPLS2, and KPLS3). For Griewank function over the interval $[-5, 5]$, the KPLS method seems to perform well when the number of observations is small compared to the dimension d. In this case, the standard separable covariance function for the ordinary kriging model is almost impossible to use because the number of parameters to be estimated is too large compared with the number of observations. Thus, the KPLS method seems more efficient in this case.

4.2 Industrial examples

The following engineering examples are based on results of numerical experiments done at SNECMA on multidisciplinary optimization. The results are stored in tables.

Aerospace turbomachinery consists of numerous blades that transfer energy between air and the rotor. The disks with compressor blades are particularly important because they must satisfy the dual criteria of aerodynamic performance and mechanical stress. Blades are mechanically and aerodynamically optimized by searching parameter space for an aerodynamic shape that ensures the best compromise that

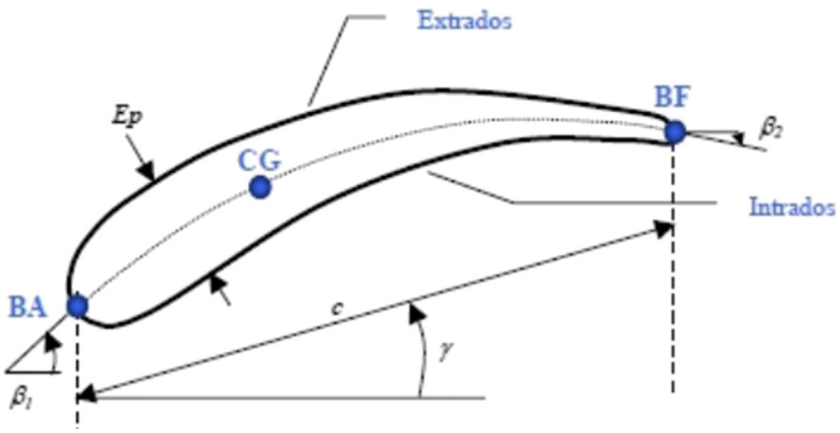


Fig. 6 Example of 2D cut of blade (c is chord; CG is gravity center; β_1 is angle for BA; β_2 is angle for BF; Ep is maximum thickness)

satisfies a set of constraints. The blade, which is a 3D entity, is first divided into a number of radial 2D profiles whose thickness is a given percentage of the distance from the hub to the shroud (see Fig. 6).

A new 3D blade is constructed by starting with the 2D profiles and then exporting them to various meshing tools before analyzing them in any specific way. The calculation is integrated into the Optimus platform (Noesis Solutions 2009), which makes it possible to integrate multiple engineering software tools (computational structural mechanics, computational fluid dynamics, ...) into a single automated work flow. Optimus, which is an industrial software package for screening variables, optimizing design, and analyzing the sensitivity and robustness, explores and analyzes the results of the work-flow to optimize product design. Toward this end, it uses high-fidelity codes or a reduced model of these codes. It also exploits a wide range of approximation models, including the ordinary kriging model.

Input variables designate geometric hyper-parameters at different percent height and outputs are related to aerodynamic efficiency, vibration criteria, mechanical stress, geometric constraints, and aerodynamic stress. Three numerical experiments are considered:

(i) The first experiment is denoted tab_1 and contains 24 input variables and 4 output variables. It has 99 training points and 52 validation points. The outputs are denoted y_1, y_2, y_3, and y_4.
(ii) The second experiment is denoted tab_2 and contains 10 input variables and only 1 output variable. It has 1295 training points and 500 validation points.
(iii) The third experiment is denoted tab_3 and contains 99 input variables and 1 output variable. It has 341 training points and 23 validation points.

Table 10 Results for tab_2 experiment data (10 input variables, 1 output variable y_1) obtained by using 1295 training points, 500 validation points, and error given by (16)

10D	Surrogate	Error (%)	CPU time
tab_2	Kriging	5.37	1 h 30 min
	KPLS1	5.07	11.69 s
	KPLS2	5.02	1 min 22 s
	KPLS3	5.34	7 min 34 s

"Kriging" refers to the ordinary kriging optimus solution and "KPLSh" refers to the KPLS model with h principal components

Points used in tab_1, tab_2, and tab_3 come from previous computationally expensive computer experiments done at SNECMA, which means that this separation between training points and verification points was imposed by SNECMA. The goal is to compare the ordinary kriging model that is implemented in the Optimus platform with the proposed KPLS model. The relative error given by (16) and the CPU time required to fit the model are reported in Tables 9–11.

The relative errors for the four models are very similar: the KPLS model results in a slightly improved accuracy for the solutions y_1, y_2, y_4 from tab_1, y_1 from tab_2, and y_1 from tab_3 but degrades slightly the solution y_3 from tab_1. The main improvement offered by the proposed model relates to the time required to fit the model, particularly for a large number of training points. Table 10 shows that, with only one principal component, the CPU time is drastically reduced compared with the Optimus model. More precisely, for tab_2, the ordinary kriging model requires 1 h 30 min whereas the KPLS1 model requires only 11 s and provides better accuracy. In addition, the results for KPLS2 and

Table 9 Results for tab_1 experiment data (24 input variables, 4 output variables y_1, y_2, y_3, y_4) obtained by using 99 training points, 52 validation points, and the error given by (16)

24D	Surrogate	y_1		y_2		y_3		y_4	
		Error (%)	CPU time	Error (%)	CPU time	Error (%)	CPU time	Error (%)	CPU time
tab_1	Kriging	0.082	8 s	4.45	8.4 s	8.97	8.17 s	6.27	8.12 s
	KPLS1	0.079	0.12 s	4.04	0.11 s	10.35	0.18 s	5.67	0.11 s
	KPLS2	0.079	0.43 s	4.06	0.69 s	10.33	0.42 s	5.67	0.19 s
	KPLS3	0.079	0.82 s	4.05	0.5 s	10.41	1.14 s	5.67	0.43 s

"Kriging" refers to the ordinary kriging Optimus solution and "KPLSh" refers to the KPLS model with h principal components

Table 11 Results for tab_3 experiment data (99 input variables, 1 output variable y_1) obtained by using 341 training points, 23 validation points, and error given by (16)

99D	Surrogate	Error (%)	CPU time
tab_3	Kriging	0.021	20 min 02 s
	KPLS1	0.19	46.6 s
	KPLS2	0.03	2 min 15 s
	KPLS3	0.02	4 min 56 s

"Kriging" refers to the ordinary kriging optimus solution and "KPLSh" refers to the KPLS model with h principal components

KPLS3 models applied to a 99D problem are very promising (see Table 11).

One other point of major interest for the proposed method is its natural compatibility with sequential enrichment techniques such as the efficient global optimization strategy (see Jones et al. 1998).

4.3 Dimensional limits

This project is financed by SNECMA and most of their design problems do not exceed 100 input variables. In addition, the toolbox Scikit-learn (version 0.14) may have memory problems when a very large number of input variables is considered. Thus, problems with more than 100 input variables are not investigated in this work. However, by optimizing memory access and storage, this limit could easily be increased.

5 Conclusion and future work

Engineering problems that require integrating surrogate models into an optimization process are receiving increasing interest within the multidisciplinary optimization community. Computationally expensive design problems can be solved efficiently by using, for example, a kriging model, which is an interesting method for approximating and replacing high-fidelity codes, largely because these models give estimation errors, which is an interesting way to solve optimization problems. The major drawback involves the construction of the kriging model and in particular the large number of hyper-parameters that must be estimated in high dimensions. In this work, we develop a new covariance kernel for handling this type of higher-dimensional problem (up to 100 dimensions). Although the PLS method requires a very short computation time to estimate θ, the estimate is often difficult to execute and computationally expensive when the number of input variables is greater than 10. The proposed KPLS model was tested by applying it to two analytic functions and by comparing its results to those tabulated in three industrial databases. The comparison highlights the efficiency of this model for up to 99 dimensions. The advantage of the KPLS models is not only the reduced CPU time, but also in that it reverts to the kriging model when the number of observations is small relative to the dimensions of the problem. Before using the KPLS model, however, the number of principal components should be tested to ensure a good balance between accuracy and CPU time.

An interesting direction for future work is to study how the design of the experiment (e.g., factorial) affects the KPLS model. Furthermore, other verification functions and other types of kernels can be used. In all cases studied herein, the first results with this proposed method reveal significant gains in terms of computation time while still ensuring good accuracy for design problems with up to 100 dimensions. The implementation of the proposed KPLS method requires minimal modifications of the classic kriging algorithm and offers further interesting advantages that can be exploited by methods of optimization by enrichment.

Acknowledgments The authors thank the anonymous reviewers for their insightful and constructive comments. We also extend our grateful thanks to A. Chiplunkar from ISAE SUPAERO, Toulouse and R. G. Regis from Saint Joseph's University, Philadelphia for their careful correction of the manuscript and to SNECMA for providing the tables of experiment results. Finally, B. Kraabel is gratefully acknowledged for carefully reviewing the paper prior to publication.

Appendix A: Equations for ordinary kriging model

The expression (3) for the ordinary kriging model is transformed into (see Forrester and Sobester 2008)

$$y(\mathbf{x}) = \hat{\beta} + \mathbf{r}_{\mathbf{xX}}^t \mathbf{R}^{-1} \left(\mathbf{y} - \mathbf{1}\hat{\beta} \right), \tag{17}$$

where $\mathbf{1}$ denotes an n-vector of ones and

$$\hat{\beta} = (\mathbf{1}^t \mathbf{R}^{-1} \mathbf{1})^{-1} \mathbf{1}^t \mathbf{R}^{-1} \mathbf{y}. \tag{18}$$

In addition, (6) is written as

$$\hat{\sigma}^2 = \frac{1}{n}\left(\mathbf{y} - \mathbf{1}\hat{\beta}\right)^t \mathbf{R}^{-1}\left(\mathbf{y} - \mathbf{1}\hat{\beta}\right). \tag{19}$$

Appendix B: Examples of kernels

Table 12 presents the most popular examples of stationary kernels. Table 13 presents the new KPLS kernels based on the examples given in Table 12.

Appendix C: Proof of equivalence kernel

For $l = 1, \ldots, h$, k_l are separable kernels (or a d-dimensional tensor product) of the same type, so $\exists\, \phi_{l1}, \ldots, \phi_{ld}$ such that

$$k_l\left(\mathbf{x}, \mathbf{x}'\right) = \prod_{i=1}^{d} \phi_{li}\left(F_l\left(\mathbf{x}\right)_i, F_l\left(\mathbf{x}'\right)_i\right), \tag{20}$$

Table 12 Examples of commonly used stationary covariance functions

Covariance functions	Expression	Hyper-parameters θ	Number of hyper-parameters to estimate
Generalized exponential	$\sigma^2 \prod_{i=1}^{d} \exp(-\theta_i \mathbf{m}_i^{p_i})$	$(\theta_1, \ldots, \theta_d, p_1, \ldots, p_d)$	$2d$
Gaussian exponential	$\sigma^2 \prod_{i=1}^{d} \exp(-\theta_i \mathbf{m}_i^2)$	$(\theta_1, \ldots, \theta_d)$	d
Matern $\frac{5}{2}$	$\sigma^2 \prod_{i=1}^{d} \left(1 + \sqrt{5}\theta_i \mathbf{m}_i + \frac{5}{3}\theta_i^2 \mathbf{m}_i^2\right) \exp(-\sqrt{5}\theta_i \mathbf{m}_i)$	$(\theta_1, \ldots, \theta_d)$	d
Matern $\frac{3}{2}$	$\sigma^2 \prod_{i=1}^{d} \left(1 + \sqrt{3}\theta_i \mathbf{m}_i\right) \exp(-\sqrt{3}\theta_i \mathbf{m}_i)$	$(\theta_1, \ldots, \theta_d)$	d

The covariance functions are written as functions of the ith component $\mathbf{m}_i = |\mathbf{x}_i - \mathbf{x}'_i|$ with $\theta_i \geq 0$ and $p_i \in [0, 2]$ for $i = 1, \ldots, d$

Table 13 Examples of KPLS covariance functions

Covariance functions	Expression	Hyper-parameters θ	Number of hyper-parameters to estimate
Generalized exponential	$\sigma^2 \prod_{l=1}^{h} \prod_{i=1}^{d} \exp\left[-\theta_l \left(\mathbf{m}_i^{(l)}\right)^{p_l}\right]$	$(\theta_1, \ldots, \theta_h, p_1, \ldots, p_h)$	$2h \ll 2d$
Gaussian exponential	$\sigma^2 \prod_{l=1}^{h} \prod_{i=1}^{d} \exp\left[-\theta_l \left(\mathbf{m}_i^{(l)}\right)^2\right]$	$(\theta_1, \ldots, \theta_h)$	$h \ll d$
Matern $\frac{5}{2}$	$\sigma^2 \prod_{l=1}^{h} \prod_{i=1}^{d} \left[1 + \sqrt{5}\theta_l \mathbf{m}_i^{(l)} + \frac{5}{3}\theta_l^2 \left(\mathbf{m}_i^{(l)}\right)^2\right] \exp\left(-\sqrt{5}\theta_l \mathbf{m}_i^{(l)}\right)$	$(\theta_1, \ldots, \theta_h)$	$h \ll d$
Matern $\frac{3}{2}$	$\sigma^2 \prod_{l=1}^{h} \prod_{i=1}^{d} \left(1 + \sqrt{3}\theta_l \mathbf{m}_i^{(l)}\right) \exp\left(-\sqrt{3}\theta_l \mathbf{m}_i^{(l)}\right)$	$(\theta_1, \ldots, \theta_h)$	$h \ll d$

The covariance functions are written as functions of the ith component $\mathbf{m}_i^{(l)} = |\mathbf{w}_{*i}^{(l)}(\mathbf{x}_i - \mathbf{x}'_i)|$ with $\theta_l \geq 0$ and $p_l \in [0, 2]$ for $l = 1, \ldots, h$

where $F_l(\mathbf{x})_i$ is the ith coordinate of $F_l(\mathbf{x})$. If we insert (20) in (14) we get

$$\begin{aligned} k_{kpls1:h}(\mathbf{x}, \mathbf{x}') &= \prod_{l=1}^{h} k_l(F_l(\mathbf{x}), F_l(\mathbf{x}')) \\ &= \prod_{l=1}^{h}\prod_{i=1}^{d} \phi_{li}(F_l(\mathbf{x})_i, F_l(\mathbf{x}')_i) \\ &= \prod_{i=1}^{d}\prod_{l=1}^{h} \phi_{li}(F_l(\mathbf{x})_i, F_l(\mathbf{x}')_i) \\ &= \prod_{i=1}^{d} \psi_i(\mathbf{x}_i, \mathbf{x}'_i), \end{aligned} \tag{21}$$

with

$$\psi_i(\mathbf{x}_i, \mathbf{x}'_i) = \prod_{l=1}^{h} \phi_{li}(F_l(\mathbf{x})_i, F_l(\mathbf{x}')_i),$$

corresponding to an one-dimensional kernel. Hence, $k_{kpls1:h}$ is a separable kernel. In particular, if we consider a generalized exponential kernel with $p_1 = \cdots = p_h = p \in [0, 2]$, we obtain

$$\begin{aligned} \psi_i(\mathbf{x}_i, \mathbf{x}'_i) &= \sigma^{\frac{2}{d}} \exp\left(-\sum_{l=1}^{h} \theta_l \left|\mathbf{w}_{*i}^{(l)}\right|^p |\mathbf{x}_i - \mathbf{x}'_i|^p\right) \\ &= \sigma^{\frac{2}{d}} \exp\left(-\eta_i |\mathbf{x}_i - \mathbf{x}'_i|^p\right), \end{aligned} \tag{22}$$

with

$$\eta_i = \sum_{l=1}^{h} \theta_l \left|\mathbf{w}_{*i}^{(l)}\right|^p.$$

We thus obtain

$$k_l(\mathbf{x}, \mathbf{x}') = \sigma^2 \prod_{i=1}^{d} \exp\left(-\eta_i |\mathbf{x}_i - \mathbf{x}'_i|^p\right). \tag{23}$$

Appendix D: Results of Griewank function in 20D and 60D over interval [−5, 5]

In Tables 14 and 15, the mean and standard deviation (std) of the numerical experiments with the Griewank function are given for 20 and 60 dimensions, respectively. To better visualize the results, boxplots are used in Figs. 7–11.

Table 14 Results for Griewank function in 20D over interval [−5, 5]

Surrogate	Statistic	50 points		100 points		200 points		300 points	
		Error (%)	CPU time	Error (%)	CPU time	Error (%)	CPU time	Error (%)	CPU time
Kriging	Mean	0.62	30.43 s	0.43	40.09 s	0.15	120.74 s	0.16	94.31 s
	std	0.03	9.03 s	0.04	11.96 s	0.02	27.49 s	0.06	21.92 s
KPLS1	Mean	0.54	0.05 s	0.53	0.12 s	0.48	0.43 s	0.45	0.89 s
	std	0.03	0.007 s	0.03	0.02 s	0.03	0.08 s	0.03	0.02 s
KPLS2	Mean	0.52	0.11 s	0.48	1.04 s	0.42	1.14 s	0.38	2.45 s
	std	0.03	0.05 s	0.04	0.97 s	0.04	0.92 s	0.04	1 s
KPLS3	Mean	0.51	1.27 s	0.46	3.09 s	0.37	3.56 s	0.35	3.52 s
	std	0.03	1.29 s	0.06	3.93 s	0.03	2.75 s	0.06	1.38 s

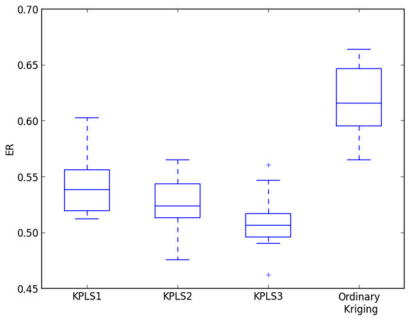
Ten trials are done for each test (50, 100, 200, and 300 training points)

Table 15 Results for Griewank function in 60D over interval [−5, 5]

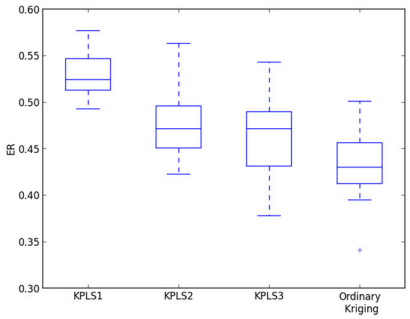
Surrogate	Statistic	50 points		100 points		200 points		300 points	
		Error (%)	CPU time	Error (%)	CPU time	Error (%)	CPU time	Error (%)	CPU time
Kriging	Mean	1.39	560.19 s	1.04	920.41 s	0.83	2015.39 s	0.65	2894.56 s
	std	0.15	200.27 s	0.05	231.34 s	0.04	239.11 s	0.03	728.48 s
KPLS1	Mean	0.92	0.07 s	0.87	0.10 s	0.82	0.37 s	0.79	0.86 s
	std	0.02	0.02 s	0.02	0.007 s	0.02	0.02 s	0.03	0.04 s
KPLS2	Mean	0.91	0.43 s	0.87	0.66 s	0.78	2.92 s	0.74	1.85 s
	std	0.03	0.54s	0.02	1.06 s	0.02	2.57 s	0.03	0.51 s
KPLS3	Mean	0.92	1.57 s	0.86	3.87 s	0.78	6.73 s	0.70	20.01 s
	std	0.04	1.98 s	0.02	5.34 s	0.02	10.94 s	0.03	26.59 s

Ten trials are done for each test (50, 100, 200, and 300 training points)

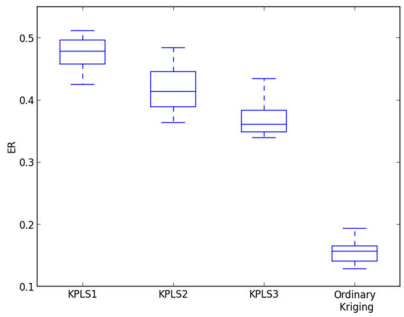
Fig. 7 RE for Griewank function in 20D over interval $[-5, 5]$. Experiments are based on the 10 latin hypercube design



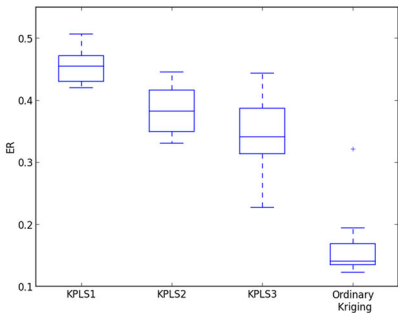
(a) $RE(\%)$ for 20 input variables and 50 sampling points.



(b) $RE(\%)$ for 20 input variables and 100 sampling points.

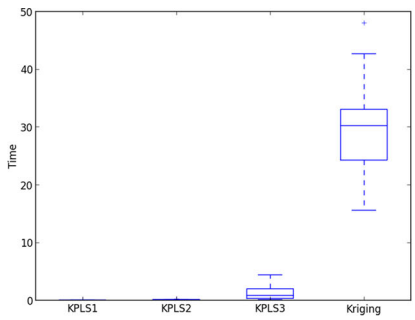


(c) $RE(\%)$ for 20 input variables and 200 sampling points.

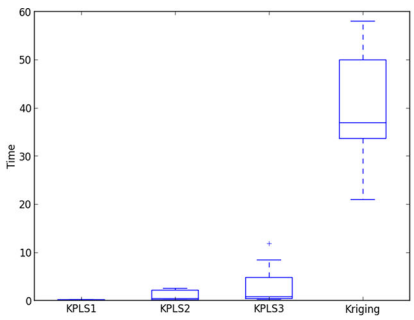


(d) $RE(\%)$ for 20 input variables and 300 sampling points.

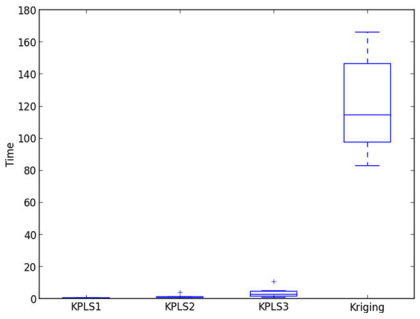
Fig. 8 CPU time for Griewank function in 20D over interval $[-5, 5]$. Experiments are based on the 10 latin hypercube design



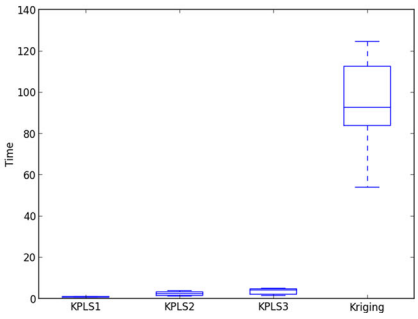
(a) CPU time for 20 input variables and 50 sampling points.



(b) CPU time for 20 input variables and 100 sampling points.

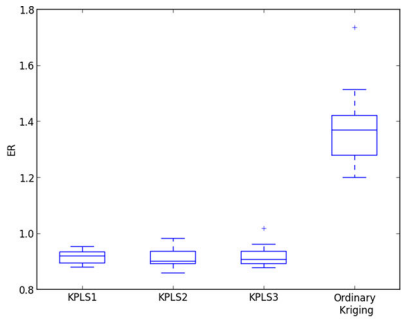


(c) CPU time for 20 input variables and 200 sampling points.

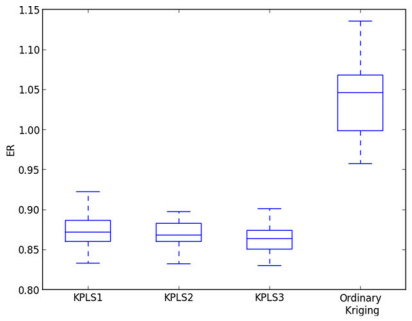


(d) CPU time for 20 input variables and 300 sampling points.

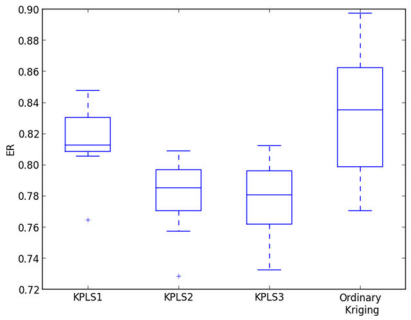
Fig. 9 RE for Griewank function in 60D over interval $[-5, 5]$. Experiments are based on the 10 latin hypercube design



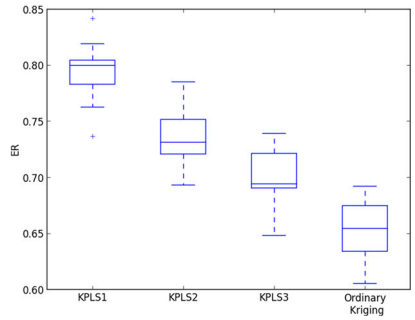
(a) $RE(\%)$ for 60 input variables and 50 sampling points.



(b) $RE(\%)$ for 60 input variables and 100 sampling points.

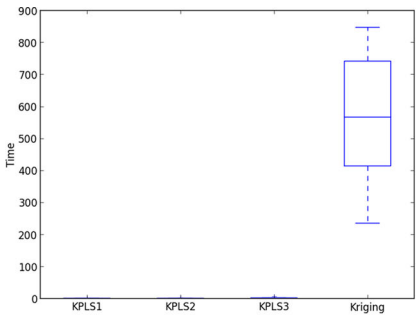


(c) $RE(\%)$ for 60 input variables and 200 sampling points.

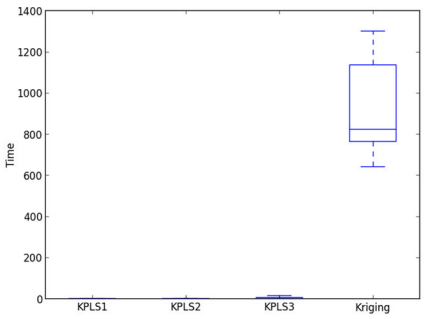


(d) $RE(\%)$ for 60 input variables and 300 sampling points.

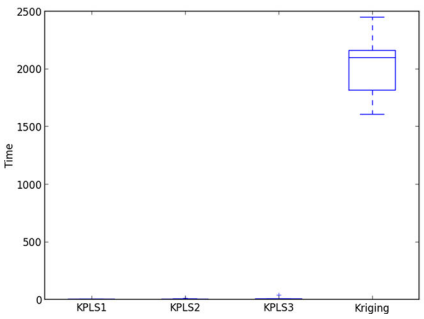
Fig. 10 CPU time for Griewank function in 60D over interval $[-5, 5]$. Experiments are based on the 10 latin hypercube design



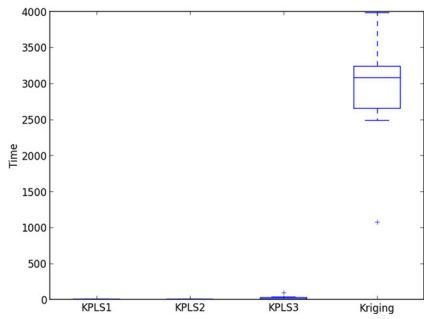
(a) CPU time for 60 input variables and 50 sampling points.



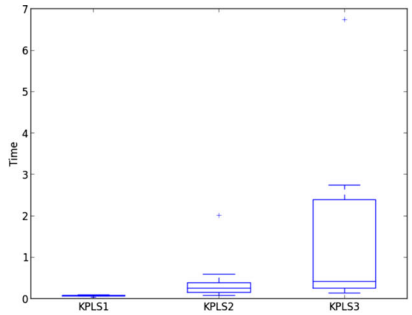
(b) CPU time for 60 input variables and 100 sampling points.



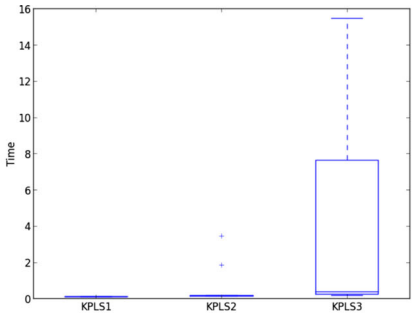
(c) CPU time for 60 input variables and 200 sampling points.



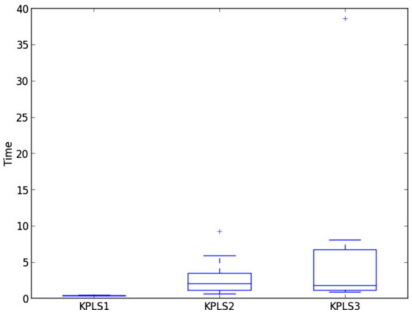
(d) CPU time for 60 input variables and 300 sampling points.



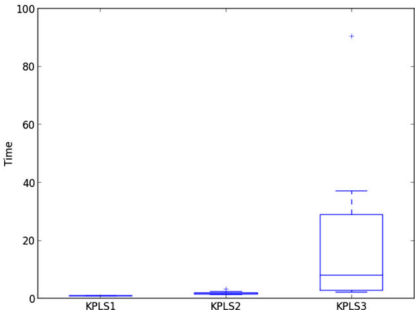
(a) CPU time for 60 input variables and 50 sampling points.



(b) CPU time for 60 input variables and 100 sampling points.



(c) CPU time for 60 input variables and 200 sampling points.



(d) CPU time for 60 input variables and 300 sampling points.

Fig. 11 CPU time for Griewank function in 60D for only KPLS models over interval $[-5, 5]$. Experiments are based on the 10 latin hypercube design

References

Alberto P, González F (2012) Partial Least Squares regression on symmetric positive-definite matrices. Rev Col Estad 36(1):177–192

Bachoc F (2013) Cross Validation and Maximum Likelihood estimation of hyper-parameters of Gaussian processes with model misspecification. Comput Stat Data Anal 66:55–69

Bishop CM (2007) Pattern recognition and machine learning (information science and statistics). Springer

Braham H, Ben Jemaa S, Sayrac B, Fort G, Moulines E (2014) Low complexity spatial interpolation for cellular coverage analysis. In: 2014 12th international symposium on modeling and optimization in mobile, ad hoc, and wireless networks (WiOpt). IEEE, pp 188–195

Buhmann MD (2003) Radial basis functions: theory and implementations, vol 12. Cambridge University Press, Cambridge

Cressie N (1988) Spatial prediction and ordinary kriging. Math Geol 20(4):405–421

Damianou A, Lawrence ND (2013) Deep gaussian processes. In: Proceedings of the sixteenth international conference on artificial intelligence and statistics, AISTATS 2013, Scottsdale, pp 207–215

Durrande N (2011) Covariance kernels for simplified and interpretable modeling. A functional and probabilistic approach. Theses, Ecole Nationale Supérieure des Mines de saint-Etienne

Durrande N, Ginsbourger D, Roustant O (2012) Additive covariance kernels for high-dimensional gaussian process modeling. Ann Fac Sci Toulouse Math 21(3):481–499

Forrester A, Sobester A, Keane A (2008) Engineering design via surrogate modelling: a practical guide. Wiley, New York

Frank IE, Friedman JH (1993) A statistical view of some chemometrics regression tools. Technometrics 35:109–148

Goovaerts P (1997) Geostatistics for natural resources evaluation (applied geostatistics). Oxford University Press, New York

Haykin S (1998) Neural networks: a comprehensive foundation, 2nd edn. Prentice Hall PTR, Upper Saddle River

Helland I (1988) On structure of partial least squares regression. Commun Stat - Simul Comput 17:581–607

Hensman J, Fusi N, Lawrence ND (2013) Gaussian processes for big data. In: Proceedings of the twenty-ninth conference on uncertainty in artificial intelligence, Bellevue, p 2013

Jones DR, Schonlau M, Welch WJ (1998) Efficient global optimization of expensive black-box functions. J Glob Optim 13(4):455–492

Lanczos C (1950) An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. J Res Natl Bur Stand 45(4):255–282

Liem RP, Martins JRRA (2014) Surrogate models and mixtures of experts in aerodynamic performance prediction for mission analysis. In: 15th AIAA/ISSMO multidisciplinary analysis and optimization conference, Atlanta, GA, AIAA-2014-2301

Manne R (1987) Analysis of two Partial-Least-Squares algorithms for multivariate calibration. Chemom Intell Lab Syst 2(1–3):187–197

Mera NS (2007) Efficient optimization processes using kriging approximation models in electrical impedance tomography. Int J Numer Methods Eng 69(1):202–220

Michalewicz Z, Schoenauer M (1996) Evolutionary algorithms for constrained parameter optimization problems. Evol Comput 4:1–32

Noesis Solutions (2009) OPTIMUS. http://www.noesissolutions.com/Noesis/optimus-details/optimus-design-optimization

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V et al (2011) Scikit-learn: machine learning in python. J Mach Learn Res 12:2825–2830

Picheny V, Ginsbourger D, Roustant O, Haftka RT, Kim NH (2010) Adaptive designs of experiments for accurate approximation of a target region. J Mech Des 132(7):071008

Powell MJ (1994) A direct search optimization method that models the objective and constraint functions by linear interpolation. In: Advances in optimization and numerical analysis. Springer, pp 51–67

Rasmussen C, Williams C (2006) Gaussian processes for machine learning. Adaptive computation and machine learning. MIT Press, Cambridge

Regis R, Shoemaker C (2013) Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization. Eng Optim 45(5):529–555

Roustant O, Ginsbourger D, Deville Y (2012) DiceKriging, DiceOptim: two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization. J Stat Softw 51(1):1–55

Sakata S, Ashida F, Zako M (2004) An efficient algorithm for Kriging approximation and optimization with large-scale sampling data. Comput Methods Appl Mech Eng 193(3):385–404

Sasena M (2002) Flexibility and efficiency enhancements for constrained global design optimization with Kriging approximations. PhD thesis, University of Michigan

Schonlau M (1998) Computer experiments and global optimization. PhD thesis, University of Waterloo

Wahba G (1990) Spline models for observational data, CBMS-NSF regional conference series in applied mathematics, vol 59. Society for Industrial and Applied Mathematics (SIAM), Philadelphia

Wahba G, Craven P (1978) Smoothing noisy data with spline functions. Estimating the correct degree of smoothing by the method of generalized cross-validation. Numer Math 31:377–404

Zimmerman DL, Homer KE (1991) A network design criterion for estimating selected attributes of the semivariogram. Environmetrics 2(4):425–441