# The Gaussian Process Modeling Module in UQLab

## C. Lataniotis[1*] iD, S. Marelli[2], B. Sudret[2]

1. Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Stefano-Franscini-Platz 5, 8093 Zurich, Switzerland
2. ETH Zurich, Stefano-Franscini-Platz 5, 8093 Zurich, Switzerland
Corresponding author: *latanioc@ethz.ch*

doi https://doi.org/10.22115/SCCE.2018.129323.1062

**ARTICLE INFO**

**ABSTRACT**

We introduce the Gaussian process (GP) modeling module developed within the UQLab software framework. The novel design of the GP-module aims at providing seamless integration of GP modeling into any uncertainty quantification workflow, as well as a standalone surrogate modeling tool. We first briefly present the key mathematical tools on the basis of GP modeling (a.k.a. Kriging), as well as the associated theoretical and computational framework. We then provide an extensive overview of the available features of the software and demonstrate its flexibility and user-friendliness. Finally, we showcase the usage and the performance of the software on several applications borrowed from different fields of engineering. These include a basic surrogate of a well-known analytical benchmark function; a hierarchical Kriging example applied to wind turbine aero-servo-elastic simulations and a more complex geotechnical example that requires a non-stationary, user-defined correlation function. The GP-module, like the rest of the scientific code that is shipped with UQLab, is open source (BSD license).

## 1. Introduction

Uncertainty quantification (UQ) through computer simulation is an interdisciplinary field that has seen rapid growth in the last decades. It aims at i) identifying and quantifying the uncertainty

in the input parameters of numerical models of physical systems, and ii) quantitatively assessing its effect on the model responses. Such a general formulation comprises some applications, including structural reliability [1], sensitivity analysis [2], reliability-based design optimization [3] and Bayesian techniques for calibration and validation of computer models [4].

Due to the high cost of repeatedly evaluating complex computational models, analyses with classical sampling techniques such as Monte Carlo simulation are often intractable. In this context, meta-modeling techniques (also known as surrogate modeling) allow one to develop fast-to-evaluate surrogate models from a limited collection of runs of the original computational model, referred to as the experimental design [5–7]. Popular surrogate modeling techniques include Kriging [8], polynomial chaos expansions [9,10] and support vector regression [11].

Kriging is a surrogate modeling technique first conceived by Krige [12] in the field of geostatistics and later introduced for the design and analysis of computer experiments by Sacks et al. [8] and Welch et al. [13]. The potential applications of Kriging in the context of civil and mechanical engineering, range from basic uncertainty propagation to reliability and sensitivity analysis [14–18]. Beyond approximating the output of a computational model, Kriging surrogates also provide local estimates of their accuracy (via the variance of the Kriging predictor). This enables adaptive schemes, *e.g.* in the context of reliability analysis [19,20] or surrogate model-based design optimization [21,22]. The local error estimates of a Kriging surrogate have also led to improved Bayesian calibration of computer models (see, *e.g.* Bachoc et al. [23]).

Although in its standard form Kriging is a stochastic interpolation method, certain extensions have been proposed for dealing with noisy observations. Such extensions have been of particular interest to the machine learning community, and they are commonly referred to as *Gaussian process regression* [24].

Some dedicated toolboxes are readily available for calculating Kriging surrogate models. Of interest to this review is general purpose software not targeted to specific Kriging applications, because they are typically limited to two or three-dimensional problems (see, *e.g.* gslib [25]). Within the R community, one of the most comprehensive and well-established Kriging packages is arguably DiceKriging, developed by the DICE consortium [26]. This set of packages provides Kriging meta-modeling as part of a framework for adaptive experimental designs and Kriging-based optimization based on the packages DiceDesign and DiceOptim [27,28]. scikit-learn provides a Python-based, machine-learning-oriented implementation of Gaussian processes for regression and classification [29]. Alternatively, PyKriging [30] offers a Kriging toolbox in python that offers basic functionality with a focus on user-friendliness. Gpy [31] offers a Gaussian process framework with a focus on regression and classification problems. Within the Matlab programming language, the first Kriging toolbox with widespread use was DACE [32]. DACE was later extended to ooDACE [33], an object-oriented Kriging implementation with a richer feature set. Small Toolbox for Kriging [34] offers an alternative Kriging implementation that is mainly focused on providing a set of functions for Kriging surrogate modeling and design of experiments. GPML [35] offers a library of functions that are directed towards Gaussian processes for regression and classification in a machine learning context. Finally, recent versions

of Matlab (starting from R2015b) provide a rapidly growing Gaussian process library for regression and classification.

Due to the variety of potential applications of Kriging, different toolboxes tend to be focused on a specific user niche. There is limited availability of general purpose Kriging toolboxes that allow for seamless integration within various UQ workflows ranging from, *e.g.* basic uncertainty propagation to reliability analysis and surrogate-model-based optimization. To this end, the Kriging toolbox presented here was developed as a module of the general purpose UQ framework, UQLab ([36], www.uqlab.com). Also, although most of the toolboxes above offer a significant set of configuration options, the support for fully customizable Kriging is often limited or not easily accessible, which can be a drawback in a research environment. Finally, the user experience may vary from user-friendly to complex (especially to access the most advanced features), often requiring a significant degree of programming knowledge. This might be rather inconvenient for applied scientists and practitioners with limited programming knowledge. Following these premises, this paper introduces the UQLab Gaussian process modeling tool (GP-module) focusing on its unique embedding into a complex uncertainty quantification environment, its user-friendliness and customisability.

The paper is structured as follows: in Section 2 a theoretical introduction to Kriging is given to highlight its main building blocks. In Section 3 the key-features of the GP-module are presented. Finally, a set of application examples is used to showcase in detail the usage of the software in Section 4, followed by a summary and a roadmap of the upcoming developments in Section 5.

## 2. Kriging theory

### 2.1. Kriging basics

Any metamodeling approach, such as Kriging, aims at approximating the response of a computational model given a finite set of observations. In this context, consider a system whose behavior is represented by a computational model $\mathcal{M}$ which maps the *M*-dimensional input parameter space $\mathcal{D}_{\mathbf{x}}$ to the 1-dimensional output space, *i.e.,* $\mathcal{M}:\mathbf{x}\in\mathcal{D}_{\mathbf{x}}\subset\mathbb{R}^{M}\mapsto y\in\mathbb{R}$ where $\mathbf{x}=\left\{x_{1},\ldots,x_{M}\right\}^{\top}$.

Kriging is a meta-modeling technique which assumes that the true model response is a realization of a Gaussian process described by the following equation [5]:

$$\mathcal{M}^{K}(\mathbf{x})=\boldsymbol{\beta}^{\top}\mathbf{f}(\mathbf{x})+\sigma^{2}Z(\mathbf{x},\omega) \tag{1}$$

Where $\boldsymbol{\beta}^{\top}\mathbf{f}(\mathbf{x})$ is the mean value of the Gaussian process, also called a *trend*, $\sigma^{2}$ is the Gaussian process variance and $Z(\mathbf{x},\omega)$ is a zero-mean, unit-variance Gaussian process. This process is fully characterized by the auto-correlation function between two sample points $R(\mathbf{x},\mathbf{x}';\boldsymbol{\theta})$. The hyperparameters $\boldsymbol{\theta}$ associated with the correlation function $R(\cdot;\boldsymbol{\theta})$ are typically unknown and need to be estimated from the available observations.

Having specified the trend and the correlation function parameters it is possible to obtain an arbitrary number of realizations of the so-called *prior* Gaussian process (see Figure 1 left). In the context of metamodelling, the goal is to calculate a prediction $\mathcal{M}^K(\mathbf{x})$ for a new point $\mathbf{x}$, given an experimental design $\mathcal{X} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$ of size $N$ and the corresponding (noise-free) model responses $\mathbf{y} = \{y^{(1)} = \mathcal{M}(\mathbf{x}^{(1)}), \ldots, y^{(N)} = \mathcal{M}(\mathbf{x}^{(N)})\}^{\top}$. A Kriging metamodel (a.k.a. Kriging predictor) provides such predictions based on the properties of the so-called *posterior* Gaussian process conditioned on the available data (see Figure 1 right). The Kriging prediction $\mathbf{x}$ corresponds to a random variate $\hat{Y}(\mathbf{x}) \sim \mathcal{N}\left(\mu_{\hat{Y}}(\mathbf{x}), \sigma_{\hat{Y}}(\mathbf{x})\right)$. Therefore the approximation of the computational model that is obtained is essentially an infinite family of such models. Each of these models is a realization (or sample) of the posterior Gaussian process. In practice, the mean response is used (see Eq. (6)) as the Kriging surrogate, while its variance (see Eq. (7)) is often interpreted as a measure of the local error of the prediction. The equations for calculating the mean and variance of a universal Kriging predictor are given next.
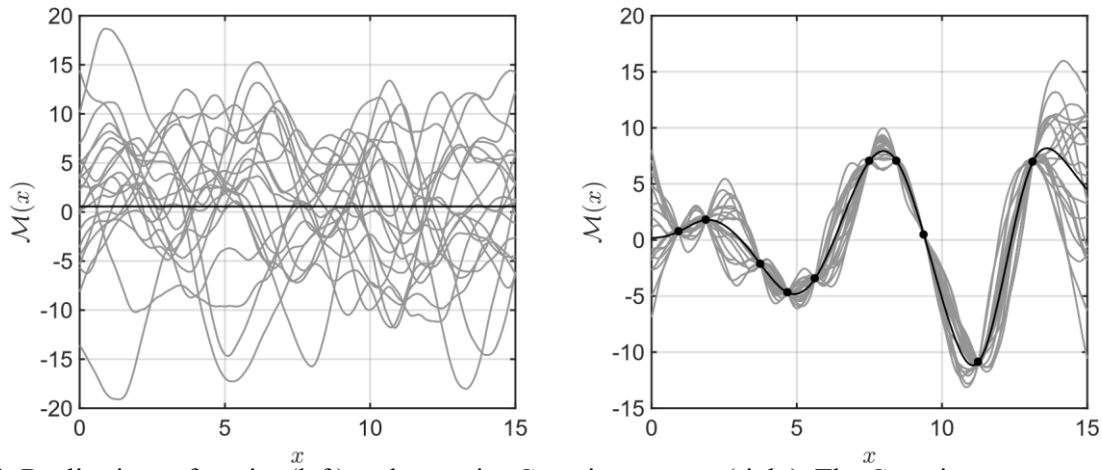


**Fig. 1.** Realisations of a prior (left) and posterior Gaussian process (right). The Gaussian process mean in each case is denoted by a black line.

The Gaussian assumption states that the vector formed by the true model responses, $\mathbf{y}$ and the prediction, $\hat{Y}(\mathbf{x})$, has a joint Gaussian distribution defined by:

$$\left\{ \begin{array}{c} \hat{Y}(\mathbf{x}) \\ \mathbf{y} \end{array} \right\} \sim \mathcal{N}_{N+1}\left( \left\{ \begin{array}{c} \mathbf{f}(\mathbf{x})^{\top}\boldsymbol{\beta} \\ \mathbf{F}\boldsymbol{\beta} \end{array} \right\}, \sigma^2 \left\{ \begin{array}{cc} 1 & \mathbf{r}(\mathbf{x})^{\top} \\ \mathbf{r}(\mathbf{x}) & \mathbf{R} \end{array} \right\} \right) \tag{2}$$

where $\mathbf{F}$ is the information matrix of generic terms:

$$F_{ij} = f_j(\mathbf{x}^{(i)}), \; i = 1, \ldots, N, \; j = 1, \ldots, P, \tag{3}$$

$\mathbf{r}(\mathbf{x})$ is the vector of cross-correlations between the prediction point $\mathbf{x}$ and each one of the observations whose terms read:

$$r_i(\mathbf{x}) = R(\mathbf{x}, \mathbf{x}^{(i)}; \boldsymbol{\theta}), \; i = 1, \ldots, N. \tag{4}$$

**R** is the correlation matrix given by:

$$R_{ij} = R(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}; \boldsymbol{\theta}), \ i,j=1,\ldots,N. \tag{5}$$

The mean and variance of the Gaussian random variate $\hat{Y}(\mathbf{x})$ (a.k.a. mean and variance of the Kriging predictor) can be calculated based on the best linear unbiased predictor properties [5]:

$$\mu_{\hat{Y}}(\mathbf{x}) = \mathbf{f}(\mathbf{x})^{\top} \boldsymbol{\beta} + \mathbf{r}(\mathbf{x})^{\top} \mathbf{R}^{-1}(\mathbf{y} - \mathbf{F}\boldsymbol{\beta}), \tag{6}$$

$$\sigma_{\hat{Y}}^{2}(\mathbf{x}) = \sigma^{2}\left(1 - \mathbf{r}(\mathbf{x})^{\top} \mathbf{R}^{-1}\mathbf{r}(\mathbf{x}) + \mathbf{u}(\mathbf{x})^{\top}\left(\mathbf{F}^{\top}\mathbf{R}^{-1}\mathbf{F}\right)^{-1}\mathbf{u}(\mathbf{x})\right) \tag{7}$$

where:

$$\boldsymbol{\beta} = \left(\mathbf{F}^{\top}\mathbf{R}^{-1}\mathbf{F}\right)^{-1}\mathbf{F}^{\top}\mathbf{R}^{-1}\mathbf{y} \tag{8}$$

is the generalized least-squares estimate of the underlying regression problem and

$$\mathbf{u}(\mathbf{x}) = \mathbf{F}^{\top}\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}) - \mathbf{f}(\mathbf{x}). \tag{9}$$

Once $\mu_{\hat{Y}}(\mathbf{x})$ and $\sigma_{\hat{Y}}^{2}(\mathbf{x})$ are available, confidence bounds on predictions can be derived by observing that:

$$\mathcal{P}\left[\hat{Y}(\mathbf{x}) \le t\right] = \Phi\left(\frac{t - \mu_{\hat{Y}}(\mathbf{x})}{\sigma_{\hat{Y}}(\mathbf{x})}\right), \tag{10}$$

**Table 1**
Formulas of the most commonly used Kriging trends.

| Trend | Formula |
|---|---|
| constant (ordinary Kriging) | $\beta_0$ |
| Linear | $\beta_0 + \sum_{i=1}^{M}\beta_i x_i$ |
| quadratic | $\beta_0 + \sum_{i=1}^{M}\beta_i x_i + \sum_{i=1}^{M}\sum_{j=1}^{M}\beta_{ij} x_i x_j$ |

Where $\Phi(\cdot)$ denotes the Gaussian cumulative distribution function. Based on Eq. (10) the confidence intervals on the predictor can be calculated by:

$$\hat{Y}(\mathbf{x}) \in \left[\mu_{\hat{Y}}(\mathbf{x}) - \Phi^{-1}\left(1 - \frac{a}{2}\right)\sigma_{\hat{Y}}(\mathbf{x}), \ \mu_{\hat{Y}}(\mathbf{x}) + \Phi^{-1}\left(1 - \frac{a}{2}\right)\sigma_{\hat{Y}}(\mathbf{x})\right] \tag{11}$$

moreover, can be interpreted as the interval within which the Kriging prediction falls with probability $1-a$.

The equations that were derived for the best linear unbiased Kriging predictor assumed that the covariance function $\sigma^2 R(\cdot; \boldsymbol{\theta})$ is known. In practice, however, the family and other properties of the correlation function need to be selected *a priori*. The hyperparameters $\theta$, the regression coefficients, and the variance $\sigma^2$ need to be estimated based on the available experimental design. This involves solving an optimization problem that is further discussed in Section 2.4. The resulting best linear unbiased predictors are called *empirical* in Santner et al. [5] because they typically result from the empirical choice of various Kriging parameters that are further discussed in Sections 2.2 - 2.4.

## 2.2. Trend

The trend refers to the mean of the Gaussian process, *i.e.*, the $\boldsymbol{\beta}^\top \mathbf{f}(\mathbf{x})$ term in Eq. (1). Using a non-zero trend is optional, but it is often preferred in practice (see, *e.g.* Rasmussen and Williams [24], Schöbi et al. [37]). Note that the mean of the Kriging predictor in Eq. (6) is not confined to be zero when the trend is zero.

In the literature, it is customary to distinguish between Kriging metamodels depending on the type of trend they use [5,24,38]. The most general and flexible formulation is *universal Kriging*, which assumes that the trend is composed of a sum of $P$ arbitrary functions $f_k(\mathbf{x})$, *i.e.*

$$\boldsymbol{\beta}^\top \mathbf{f}(\mathbf{x}) = \sum_{k=1}^{P} \beta_k f_k(\mathbf{x}). \tag{12}$$

Some of the most commonly used trends for universal Kriging are given for reference in Table 1. *Simple Kriging* assumes that the trend has a known constant value, *i.e.* $P = 1$, $f_1(\mathbf{x})$ and $\beta_1$ is known. In *Ordinary Kriging*, the trend has a constant but unknown value, *i.e.* $P = 1$, $f_1(\mathbf{x}) = 1$ and $\beta_1$ is unknown.

## 2.3. Correlation function

The correlation function (also called *kernel* in the literature, or *covariance* function if it includes the Gaussian process variance $\sigma^2$) is a crucial ingredient for a Kriging metamodel since it contains the assumptions about the function that is being approximated. An arbitrary function of $(\mathbf{x}, \mathbf{x}')$ is in general not a valid correlation function. In order to be admissible, it has to be chosen in the set of positive definite kernels. However, checking for positive definiteness of a kernel can be a challenging task. Therefore it is usually the case in practice to select families of kernels known to be positive definite and to estimate their parameters based on the available experimental design and model responses (see Section 2.4). A usual assumption is to consider kernels that depend only on the quantity $h = \|\mathbf{x} - \mathbf{x}'\|$ which are called *stationary*. A list of stationary kernels commonly used in the literature can be found in Table 2. Different correlation families result in different levels of smoothness for the associated Gaussian processes, as depicted in Figure 2 [24].

In case of multidimensional inputs ($M > 1$), it is common practice to obtain admissible kernels as functions of one-dimensional correlation families like the ones in Table 2. Two standard approaches in the literature are the *separable* correlation type [8]:

$$R(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = \prod_{i=1}^{M} R(x_i, x_i', \theta_i) \tag{13}$$

**Table 2**
List of available correlation families.

| Name | Formula |
|---|---|
| Linear | $R(h; \theta) = \max\left(0, 1 - \dfrac{h}{\theta}\right)$ |
| Exponential | $R(h; \theta) = \exp\left(-\dfrac{|h|}{\theta}\right)$ |
| Matérn 3/2 | $R(h; \theta) = \left(1 + \dfrac{\sqrt{3}\,|h|}{\theta}\right) \exp\left(-\dfrac{\sqrt{3}\,|h|}{\theta}\right)$ |
| Matérn 5/2 | $R(h; \theta) = \left(1 + \dfrac{\sqrt{5}\,|h|}{\theta} + \dfrac{5h^2}{3\theta^2}\right) \exp\left(-\dfrac{\sqrt{5}\,|h|}{\theta}\right)$ |
| Gaussian (squared exponential) | $R(h; \theta) = \exp\left(-\sum_{i=1}^{M}\left(\dfrac{h}{\theta}\right)^2\right)$ |
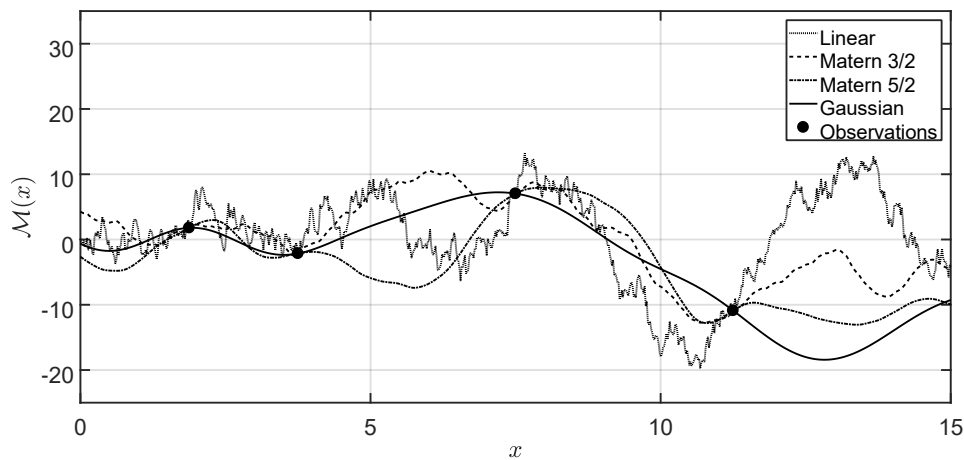


**Fig. 2.** Realisations of Gaussian processes, characterized by various correlation families and the same length-scale ($\theta$) value.

and the *ellipsoidal* type [24]:

$$R(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = R(h) \ , \ h = \sqrt{\sum_{i=1}^{M}\left(\frac{x_i - x_i'}{\theta_i}\right)^2} \ . \tag{14}$$

Although typically $\boldsymbol{\theta} \in \mathbb{R}^M$ this is not necessarily true in the general case, since the number of components of $\boldsymbol{\theta}$ that corresponds to each input dimension may vary. In the current stage, it is assumed however that one element of $\boldsymbol{\theta}$ is used per dimension for notational clarity.

In certain scenarios (*e.g.,* based on prior knowledge), *isotropic* correlation functions can be used for multidimensional inputs. In that case, the same correlation function parameters $\boldsymbol{\theta}$ are used for each input dimension in Eq. (13), and Eq. (14).

## 2.4. Estimating the hyperparameters

In most practical applications of Kriging surrogate modeling, the hyperparameters $\boldsymbol{\theta}$ are estimated given an experimental design $\mathcal{X}$ and model responses $\mathbf{y}$. *Maximum likelihood* and *cross-validation* are the most commonly used methods for doing so and further discussed next.

The *maximum likelihood* approach aims at finding the set of parameters $\boldsymbol{\beta}$, $\boldsymbol{\theta}$, $\sigma^2$ such that the likelihood of the observations $\mathbf{y} = \left\{ \mathcal{M}(\mathbf{x}^{(1)}), \ldots, \mathcal{M}(\mathbf{x}^{(N)}) \right\}^\top$ is maximal. Since $\mathbf{y}$ follows a multivariate Gaussian distribution, the likelihood function reads:

$$L(\mathbf{y} \mid \boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta}) = \frac{\det(\mathbf{R})^{-1/2}}{(2\pi\sigma^2)^{N/2}} \exp\left[ -\frac{1}{2\sigma^2} (\mathbf{y} - \mathbf{F}\boldsymbol{\beta})^\top \mathbf{R}^{-1} (\mathbf{y} - \mathbf{F}\boldsymbol{\beta}) \right]. \tag{15}$$

For any given value of $\boldsymbol{\theta}$, the maximization of the likelihood *w.r.t.* $\boldsymbol{\beta}$, and $\sigma^2$ is a convex quadratic programming problem. Consequently, it admits closed form generalized least-squares estimates of $\boldsymbol{\beta}$ and $\sigma^2$ (for proof and more details see, *e.g.* Santner et al. [5]):

$$\boldsymbol{\beta} = \boldsymbol{\beta}(\boldsymbol{\theta}) = \left( \mathbf{F}^\top \mathbf{R}^{-1} \mathbf{F} \right)^{-1} \mathbf{F}^\top \mathbf{R}^{-1} \mathbf{y}, \tag{16}$$

$$\sigma^2 = \sigma^2(\boldsymbol{\theta}) = \frac{1}{N} (\mathbf{y} - \mathbf{F}\boldsymbol{\beta})^\top \mathbf{R}^{-1} (\mathbf{y} - \mathbf{F}\boldsymbol{\beta}). \tag{17}$$

The value of the hyperparameters $\boldsymbol{\theta}$ is calculated by solving the optimization problem:

$$\boldsymbol{\theta} = \arg\min_{\mathcal{D}_{\boldsymbol{\theta}}} \left( -\log L(\mathbf{y} \mid \boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta}) \right) \tag{18}$$

Based on Eqs (15) - (17) the optimization problem in Eq. (18) can be written as follows:

$$\boldsymbol{\theta} = \arg\min_{\mathcal{D}_{\boldsymbol{\theta}}} \left( \frac{1}{2} \log\left( \det\left( \mathbf{R} \right) \right) + \frac{N}{2} \log(2\pi\sigma^2) + \frac{N}{2} \right) \tag{19}$$

The *cross-validation* method (also known as *K*-fold cross-validation) is based instead on partitioning the whole set of observations $\mathcal{S} \overset{\text{def}}{=} \{\mathcal{X}, \mathbf{y}\}$ into $K$ mutually exclusive and collectively exhaustive subsets $\{\mathcal{S}_k, k = 1, \ldots, K\}$ such that

$$\mathcal{S}_i \cap \mathcal{S}_j = \varnothing, \ \forall (i, j) \in \{1, \dots, K\}^2 \ \text{and} \ \bigcup_{k=1}^{K} \mathcal{S}_k = \mathcal{S} \tag{20}$$

The *k*-th set of cross-validated predictions is obtained by calculating the Kriging predictor using all the subsets but the *k*-th one and evaluating its predictions on that specific *k*-th fold that was left apart. The leave-one-out cross-validation procedure corresponds to the special case that the number of classes is equal to the number of observations ($K = N$).

In the latter case the objective function is [5,39]:

$$\boldsymbol{\theta} = \arg \min_{\mathcal{D}_{\boldsymbol{\theta}}} \sum_{i=1}^{K} \left( \mathcal{M}(\mathbf{x}^{(i)}) - \mu_{\hat{Y}, (-i)}(\mathbf{x}^{(i)}) \right)^2 \tag{21}$$

Where $\mu_{\hat{Y}, (-i)}(\mathbf{x}^{(i)})$ is the mean Kriging predictor that was calculated using $\mathcal{S} \backslash \{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}$ evaluated at a point $\mathbf{x}^{(i)}$. Notice that for the case of leave-one-out cross-validation, *i* is an index, but in the general case, *i* is a vector of indices. Calculating the objective function in Eq. (21) requires the calculation of *K* Kriging surrogates. The computational requirements for performing this operation can be significantly reduced as shown in Dubrule [40].

The estimate $\sigma^2$ is calculated using the following equation [39,41]:

$$\sigma^2 = \sigma^2(\boldsymbol{\theta}) = \frac{1}{K} \sum_{i=1}^{K} \frac{\left( \mathcal{M}(\mathbf{x}^{(i)}) - \mu_{\hat{Y},(-i)}(\mathbf{x}^{(i)}) \right)^2}{\sigma^2_{\hat{Y},(-i)}(\mathbf{x}^{(i)})} \tag{22}$$

where $\sigma^2_{\hat{Y},(-i)}(\mathbf{x}^{(i)})$ is the variance of a Kriging predictor that was calculated using $\mathcal{S} \backslash \{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}$, evaluated at a point $\mathbf{x}^{(i)}$. When *i* is a set of indices, the division and the squared operations in Eq. (22) are performed element-wise.

Numerically solving the optimization problems described in Eq. (19) (maximum likelihood case) alternatively, Eq. (21) (cross-validation case) relies on either local (*e.g.,* gradient-based) or global (*e.g.,* evolutionary) algorithms. On the one hand, local methods tend to converge faster and require fewer objective function evaluations than their global counterparts. On the other hand, the existence of flat regions and multiple local minima, especially for larger input dimension, can lead gradient methods to poor performance when compared to global methods. It is common practice to combine both strategies sequentially to improve global optimization results with a final local search (which is also known as *hybrid* methods).

It can often be the case in engineering applications that different components of the input variable $\mathbf{x}$ take values that differ by orders of magnitude. In such cases, potential numerical instabilities can be avoided by scaling $\mathcal{X} \mapsto \mathcal{U}$, *e.g.,* as follows:

$$u_j^{(i)} = \frac{x_j^{(i)} - \mathbb{E}\left[\mathcal{X}_j\right]}{\sqrt{\mathrm{Var}\left[\mathcal{X}_j\right]}} , \ i = 1,\ldots,N , \ j = 1,\ldots,M \tag{23}$$

where $u_j^{(i)}$ (resp. $x_j^{(i)}$) refer to the *i*-th sample of the *j*-th component $\mathcal{U}$ (resp. of $\mathcal{X}$) and $\mathbb{E}\left[\mathcal{X}_j\right]$ $\mathrm{Var}\left[\mathcal{X}_j\right]$ refer to the empirical mean and variance of the *j*-th component of $\mathcal{X}$.

## 3. The UQLab Gaussian process modeling module

### 3.1. The UQLab project

UQLab is a software framework developed by the Chair of Risk, Safety and Uncertainty Quantification at ETH Zürich [36]. The goal of this project is to provide an uncertainty quantification tool that is accessible also to a non-highly-IT trained scientific audience. Due to the broadness of the UQ scope, a correspondingly general theoretical framework is required. The theoretical backbone of the UQLab software lies in the global uncertainty framework developed by Sudret [42], De Rocquigny et al. [43], sketched in Figure 3a. According to this framework, the solution of any UQ problem can generally be decomposed into the following steps:

Step A   Define the *physical model* and the quantities of interest for the analysis. It is a deterministic representation of an arbitrarily complex physical model, *e.g.,* a finite element model in civil and mechanical engineering. In this category also lie metamodels, such as Kriging, since once they are calculated, they can be used as surrogates of the underlying "true" model.

Step B   Identify and quantify the sources of uncertainty in the parameters of the system that serves as input for Step A. They are represented by a set of random variables and their joint probability density function (PDF).

Step C   Propagate the uncertainties identified in Step B through the computational model in Step A to characterize the uncertainty in the model response. This type of analyses includes moments analysis, full PDF characterization, rare events estimation, sensitivity analysis, etc.

Step C'  Optionally, exploit the by-products of the analysis in Step C to update the sources of uncertainty, *e.g.,* by performing model reduction based on sensitivity analysis.
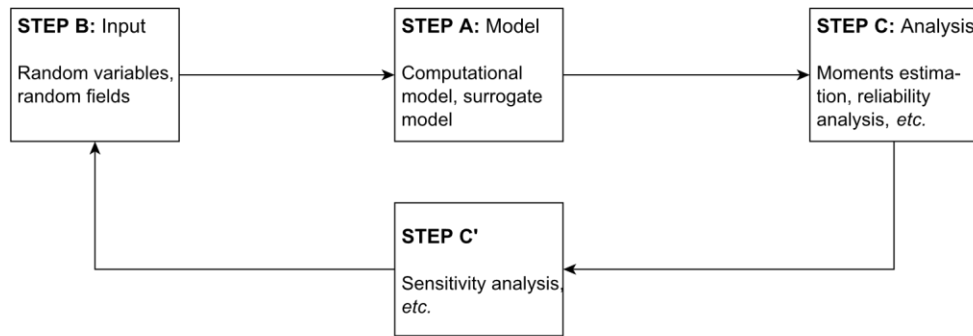
a The theoretical UQ framework based on which any UQ problem can be described.
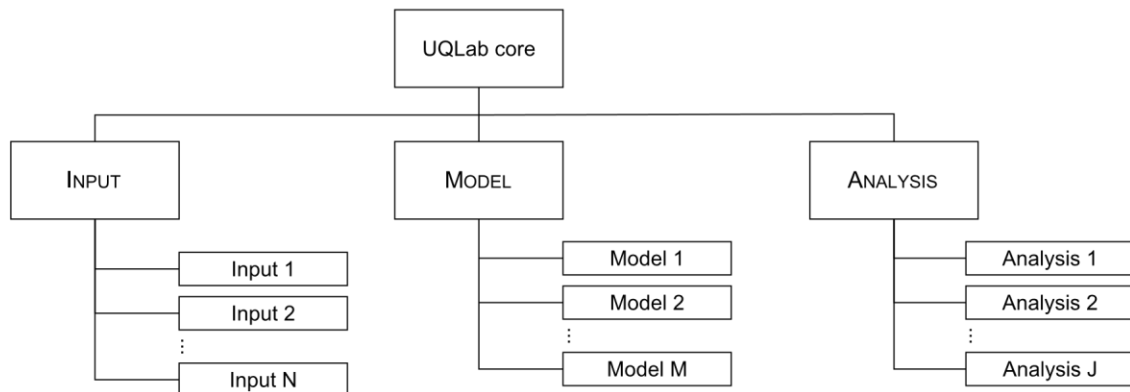


b The modular structure of the UQLab framework. An arbitrary number of objects (Input, Model, Analysis) can be connected at any stage of the UQ problem.

**Fig. 3.** An abstract illustration of the UQLab architecture (b) based on the theoretical UQ framework in (a) by Sudret [42].

These components introduce a clear semantic distinction between the elements involved in any UQ problem: model, input, and analysis. This theoretical framework provides the ideal foundation for the development of the information flow model in a multi-purpose UQ software.

At the core of UQLab lies a modular infrastructure that closely follows the semantics previously described, graphically represented in Figure 3b. The three steps identified in Figure 3a are directly mapped to *core modules* in Figure 3b: model corresponds to Step A (physical modeling, metamodeling), input to Step B (sources of uncertainty) and analysis to Step C (uncertainty analysis). Within the UQLab framework, a *module* refers to some particular functionality, *e.g.,* the GP-module provides Kriging surrogate modeling. Each module extends the functionalities of one of the core modules. It can be either self-contained or capitalize on other modules for extended functionalities.

The real "actors" of a UQ problem are contained in the *objects* connected to each of the core modules. A typical example of such objects would be an input object that generates samples distributed according to arbitrary PDFs, a model object that runs a complex FEM simulation, or an analysis object that performs reliability analysis. The platform allows one to define an arbitrary number of objects and select the desired ones at various stages of the solution of a complex UQ problem.

UQLab first became freely available to the academic community on July 2015 as a beta version. On April 2017 the version 1.0 of UQLab was released. Starting from version 1.0 all the scientific code of the software is open-source (BSD license). By May 2018 around 1300 users have already registered and used it.

## 3.2. The GP-module

Kriging is one of the metamodelling modules available in UQLab [44]. Following the semantics described in the previous section, it is attached to the model core module. Although other modules can use the GP-module itself, *e.g.,* an analysis module performing reliability analysis combining Kriging and Monte Carlo Simulation (AK-MCS) [19,45], the focus of this work is on the capabilities of the GP-module itself.

An overview of the available features of the GP-module is given in Table 3. The GP-module incorporates the four ingredients identified in Section 2.1:

- *Trends*: Universal Kriging trends are fully supported, including simple, ordinary, or polynomial of arbitrary degree. Also, custom basis functions $f(x)$ or a completely custom trend function may be specified
- *Correlation functions*: Standard correlation families from the literature are readily available as well as the possibility of creating user-defined ones. For multi-dimensional inputs, ellipsoidal and separable correlation functions can be used, also allowed for isotropic ones. Fully user-specified correlation functions are also supported
- *Estimation methods*: Maximum likelihood (Eq. (19)) and cross-validation (Eq. (21)) methods can be used for estimating the hyper-parameters
- *Optimisation methods*: Matlab's built-in local and global optimization methods are offered, namely BFGS and genetic algorithm as well as a genetic algorithm with BFGS refinement (hybrid).

In addition, various scaling operations are allowed for avoiding numerical instabilities during the hyperparameters estimation. Such operations may vary from simple zero-mean scaling to more advanced ones such as iso-probabilistic transformations by interfacing with other UQLab modules.

Following the general design principle of UQLab concerning user-friendliness, all the possible configuration options have default values pre-assigned to allow basic usage of the module with very few lines of code (see Section 4.1). A Matlab structure variable is used to specify a Kriging configuration, called `KOptions` in the following sections.

To showcase the minimal working code for obtaining a Kriging surrogate, a simple application is considered. The experimental design consists of 8 random samples in the $[0,15]$ interval, and it is contained in the variable `XED`. The "true" model is $\mathcal{M}(x) = x\sin(x)$, and the corresponding model responses are stored in the variable `YED`. The minimal code required for obtaining a Kriging surrogate, given `XED` and `YED` is the following:

```
KOptions.Type = 'Metamodel';
KOptions.MetaType = 'Kriging';
```

```
KOptions.ExpDesign.X = XED;
KOptions.ExpDesign.Y = YED;
myKriging = uq_createModel(KOptions);
```

The first line clarifies the type of UQLab object that is being requested. Following the general UQ Framework in Figure 3a a model object of type 'Metamodel' is created. The next line specifies the type of metamodel, followed by the manual specification of the experimental design. Finally, the UQLab command `uq_createModel` is used in order to create a model object using the configuration options in `KOptions`.

The resulting Kriging metamodel object `myKriging` contains all the required information to compute the mean and variance of the Kriging predictor on new test points (X). This can be done using the following command:

```
[meanY, varY] = uq_evalModel(myKriging, X);
```
where `meanY` corresponds to the mean and `varY` to the variance of the Kriging predictor on the test points (see Eqs. (6), (7)).

Once the metamodel is created, a report of the main properties of the Kriging surrogate model can be printed on screen by:

```
uq_print(myKriging);

%-------------- Kriging metamodel --------------%
Object Name: Model 1 Input Dimension:      1
Experimental Design
    Sampling:     User
    X size:       [8x1]
    Y size:       [8x1]
    Trend

    Type:         ordinary
    Degree:       0
Gaussian Process
```
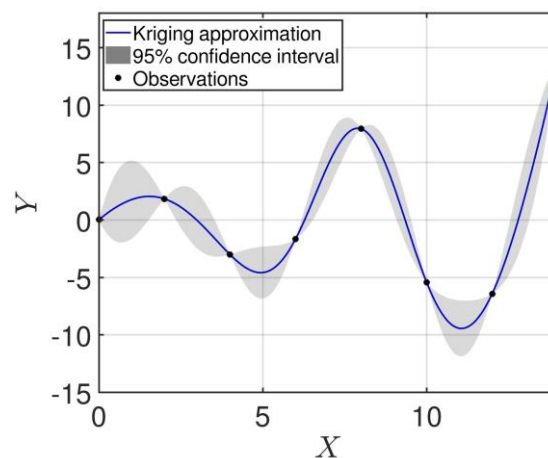


**Fig. 4.** The output of `uq_display` of a Kriging model object having a one-dimensional input.

**Table 3**
List of features of the UQLab GP-module. The default values for each property is in bold.

| Feature | Specification | Value | Description |
|---|---|---|---|
| Trend | | Simple | A constant term specified by the user (simple Kriging) |
| | | **Ordinary** | A constant term estimated using Eq. (8) (ordinary Kriging) |
| | | Polynomial basis | The trend in Eq. (12) consists of polynomial basis function $f_k$ of arbitrary degree |
| | | Custom basis | The trend in Eq. (12) consists of arbitrary functions $f_k$ |
| | | Custom trend | Custom trend function that computes **F** directly |
| Correlation | Types | Separable | As described in Eq. (13). Both isotropic and anisotropic variants are supported. |
| | | **Ellipsoidal** | As described in Eq. (14). Both isotropic and anisotropic variants are supported. |
| | | Custom | Custom correlation function that computes **R** directly |
| | Families | Commonly used | All the correlation families reported in Table 2 are available |
| | | Custom | A custom correlation family can be specified |
| Estimation | | ML | Maximum-likelihood estimation (see Eq. (19)) |
| | | **CV** | $K$-fold Cross-Validation method (see Eq. (21)). Any $K$ value is supported |
| Optimisation | | BFGS | Gradient-based optimization method (BroydenFletcher-Goldfarb-Shanno algorithm). Matlab built-in |
| | | GA | Global optimization method (genetic algorithm). Matlab built-in |
| | | **HGA** | Genetic algorithm optimization with BFGS refinement |

```
    Corr. Type:             ellipsoidal(anisotropic)
    Corr. family:           matern-5_2
    sigma^2:                    4.787983e+01
    Estimation method: Cross-Validation

    Hyperparameters
     theta:                     [ 0.00100 ]
     Optim. method:         Hybrid Genetic Algorithm

    Leave-one-out error:    4.3698313e-01
    %--------------------------------------------%
```

It can be observed that the default values for the trend, correlation function, estimation, and optimization method have been assigned (see Table 3). A visual representation of the metamodel can be obtained by:

```
uq_display(myKriging);
```

Note that the `uq_display` command can only be used for quickly visualizing Kriging surrogates when the inputs are one- or two-dimensional. The figure produced by `uq_display` is shown in Figure 4.

## 4. Application examples

### 4.1. Basic example

The goal of this introductory example is to calculate a Kriging surrogate of a well-known surrogate modeling benchmark, the Branin-Hoo function. This function has been traditionally used as a benchmark for global optimization methods (see, *e.g.* Jones et al. [46]). A slightly modified version is considered this work, that was first proposed as a surrogate modeling benchmark by Forrester et al. [7] due to its representative shape concerning engineering applications. It is an analytical function given by:

$$\mathcal{M}(\mathbf{x}) = a\left(x_2 - bx_1^2 + cx_1 - r^2\right)^2 + s\left(1 - t\right)\cos\left(x_1\right) + s, \; x \in \mathbb{R}^2. \tag{24}$$

Some standard values of the parameters are used, namely $a = 1$, $b = 5.1$ $(4\pi^2)$, $c = 5/\pi$, $r = 6$, $s = 10$ and $t = 1/(8\pi)$. The function is evaluated on the square $x_1 \in [-5,10]$, $x_2 \in [0,15]$.

By taking advantage of the input and model modules of UQLab, the experimental design and model responses that will be used for calculating the surrogate can be generated with minimal effort. First, the probabilistic input model and the true model are defined as follows:

```
% Start the UQLab framework
uqlab;
% Specify the probabilistic input model IOptions.Marginals(1).Type =
'Uniform';
IOptions.Marginals(1).Parameters = [-5, 10];
IOptions.Marginals(2).Type = 'Uniform'; IOptions.Marginals(2).Parameters
= [0, 15];
myInput = uq_createInput(IOptions);
% Specify the computational model
MOptions.mString = ['(X(:,2) - .1/(2*pi)^2*X(:,1).^2 + 5/pi*X(:,1) …
- 6).^2' + 10*(1-1/(8*pi))*cos(X(:,1)) + 10'];
myModel = uq_createModel(MOptions);
```
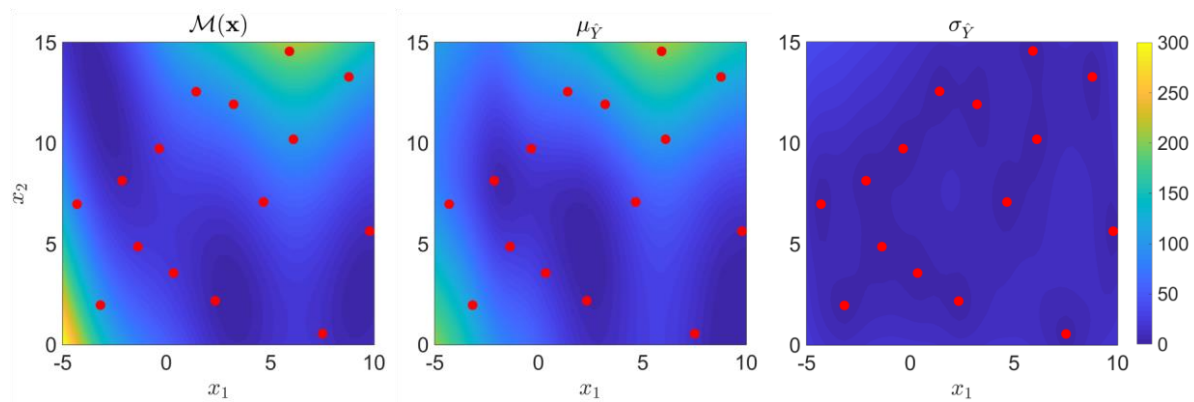
**Fig. 5.** From left to right: the Branin-Hoo function (true model) followed by the mean and standard deviation of the Kriging predictor. Red dots illustrate the experimental design.

Note that the model object of the Branin-Hoo function can be equally coded in a Matlab m-file or written as a string (which is a useful feature for simple demo functions only).

Next, the experimental design `XED` is generated along with the corresponding true model responses `YED`. The Latin Hypercube Sampling (LHS) method is used to obtain a space-filling experimental design of 15 samples [47]:

```
% Draw 15 samples using Latin Hypercube Sampling
XED = uq_getSample(15, 'LHS');
% Calculate the corresponding model responses YED = uq_evalModel(myModel,
XED);
```

A Kriging surrogate model using the `XED`, `YED` variables can be created as follows:

```
KOptions.Type = 'Metamodel';
KOptions.MetaType = 'Kriging';
KOptions.ExpDesign.Sampling = 'user';
KOptions.ExpDesign.X = XED;
KOptions.ExpDesign.Y = YED;
myKriging = uq_createModel(KOptions);
```

All the required ingredients for obtaining a Kriging surrogate are assigned default values unless specified by the user (see Section 3.2). The surrogate that is obtained can be visually inspected by issuing the command:

```
uq_display(myKriging);
```

The result of the `uq_display` command is shown in Figure 5. The Kriging surrogate `myKriging` can be used like any other model (*e.g.,* `myModel`) to calculate its response given a new sample of the input `X` using the `uq_evalModel` function. For example, the mean predictor, `meanY`, of 100 samples generated by Monte Carlo sampling can be computed as follows:

```
X = uq_getSample(100);
meanY = uq_evalModel(myKriging, X);
```

More information can be extracted from the Kriging predictor using a slightly different syntax. The following code:

```
[meanY, varY, covY] = uq_evalModel(myKriging, X);
```

allows to retrieve the 100 × 1 Kriging mean `meanY`, the 100 × 1 Kriging variance `varY` and the 100 × 100 full covariance matrix of the surrogate model responses `covY`.
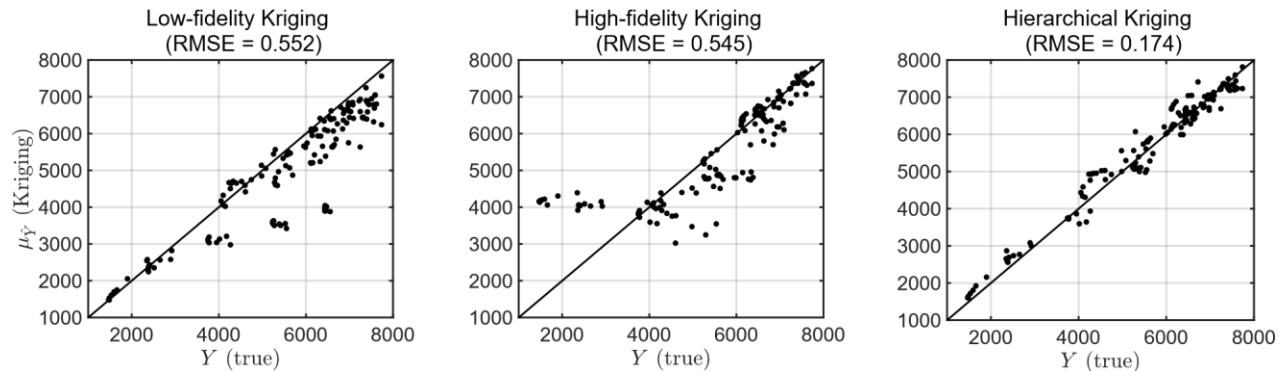


**Fig. 6.** Comparison of true model output (from high fidelity simulations) versus various Kriging surrogates on a validation set of size 150.

## 4.2. Hierarchical kriging

To further illustrate the flexibility that can be achieved with the use of arbitrary trend functions, a hierarchical Kriging application is showcased. Hierarchical Kriging [48] is one Kriging extension aiming to fuse information from experimental designs related to different physical models of different fidelity. This is achieved by first calculating a Kriging surrogate using the low-fidelity observations and then using it as the trend of the high-fidelity surrogate. This approach can be extended to more fidelity levels in a similar fashion. A set of observations and model responses is used that originates from aero-servo-elastic simulations of a wind-turbine as presented in Abdallah et al. [49]. Given a set of input parameters related to the wind flow, the output of interest is the maximal bending moment at the blade root of a wind turbine.

Two types of simulators are available for estimating the maximal bending moment given the wind conditions. A low-fidelity simulator can generate estimates of the output with minimal computation time at the cost of lower accuracy. On the other hand, a high-fidelity simulator can more accurately predict the maximal bending moment at a significantly higher computational cost. In this example a total of 300 low-fidelity and 15 high-fidelity simulations are available. First, a Kriging surrogate is computed on the low-fidelity dataset that is contained in variables `XED_LF`, `YED_LF` as follows:

```
% Create the low-fidelity surrogate
KOptions_LF.Type = 'Metamodel';
KOptions_LF.MetaType = 'Kriging';
KOptions_LF.ExpDesign.X = XED_LF;
KOptions_LF.ExpDesign.Y = YED_LF;
KOptions_LF.Corr.Family = 'Matern-3_2';
myKriging_LF = uq_createModel(KOptions_LF);
```

Using the same configuration options, another Kriging surrogate is computed using the high-fidelity dataset (`XED_HF` and `YED_HF`):

```
% Create the high-fidelity surrogate
KOptions_HF.Type = 'Metamodel';
```

```
KOptions_HF.MetaType = 'Kriging';
KOptions_HF.ExpDesign.X = XED_HF;
KOptions_HF.ExpDesign.Y = YED_HF;
KOptions_HF.Corr.Family = 'Matern-3_2';
myKriging_HF = uq_createModel(KOptions_HF);
```

Now a hierarchical Kriging surrogate is computed which is trained on the high-fidelity dataset but uses the low-fidelity Kriging surrogate (*i.e.,* its mean predictor) as a trend:

```
% Create the hierarchical Kriging surrogate
KOptions_Hier.Type = 'Metamodel';
KOptions_Hier.MetaType = 'Kriging';
KOptions_Hier.ExpDesign.X = XED_HF;
KOptions_Hier.ExpDesign.Y = YED_HF;
KOptions_Hier.Corr.Family = 'Matern-3_2';
KOptions_Hier.Trend.Type = 'custom';
KOptions_Hier.Trend.CustomF = @(x) uq_evalModel(myKriging_LF, x);
KOptions_Hier.Scaling = false;
myKriging_Hier = uq_createModel(KOptions_Hier);
```

The option `KOptions_Hier.Scaling` refers to the scaling of the input space before computing the surrogate model. In case of hierarchical Kriging scaling should be disabled because the low-fidelity surrogate is calculated on the original data and needs to be used "as is".

The performance of the different surrogate models is tested on a separate validation set of 150 high-fidelity simulations that is contained in the variables `XVAL_HF` and `YVAL_HF`. The output mean Kriging predictor on the validation set is calculated as follows:

```
meanY_LF = uq_evalModel(myKriging_LF, XVAL_HF);
meanY_HF = uq_evalModel(myKriging_HF, XVAL_HF);
meanY_Hier = uq_evalModel(myKriging_Hier, XVAL_HF);
```

where `meanY_LF`, `meanY_HF` and `meanY_Hier` correspond to the low-fidelity, high-fidelity and hierarchical Kriging predictors respectively.

In Figure 6 a comparison of the true model output `YVAL_HF` versus the mean Kriging predictors is made. In each case the Root Mean Square Error (RMSE) is reported for quantifying the predictive performance of the surrogate:

$$E_{RMSE} = \frac{1}{N\mathrm{Var}[Y]}\sum_{i=1}^{N}\left(Y^{(i)} - \mu_{\hat{Y}}^{(i)}\right)^2 \tag{25}$$

where $Y$ denotes the true model outputs (in this case `YVAL_HF`), $\mu_{\hat{Y}}$ the Kriging predictor mean (in this case variables `meanY_LF`, `meanY_HF` and `meanY_Hier` for each surrogate, respectively) and $N$ the number of samples in the validation set.

In this example, by taking advantage of the low-cost, low-fidelity observations, the hierarchical Kriging predictor achieves a 68% decrease of the RMSE on the validation set compared to the Kriging model that was solely based on the high-fidelity measurements. Moreover, by inspecting

the mean responses of each Kriging predictor in Figure 6, it is clear that the hierarchical Kriging surrogate significantly reduces the prediction bias compared to the low- and high-fidelity ones taken as standalone. As demonstrated by this application, building a hierarchical Kriging surrogate model requires minimal effort thanks to the customisability of the GP-module.

## 4.3. Kriging with custom correlation function

This example illustrates how the correlation function customization capabilities of the GP-module can be used to apply Kriging in a non-standard setting.

Consider the discontinuous subsurface model given in Figure 7, which may represent the distribution of some soil property (e.g., porosity) in the presence of a fault. The true model consists of two realizations of two distinct random processes on the two regions $A_1$ and $A_2$ at the left and right of the fault, respectively:

$$\mathcal{M}(\mathbf{x}) = \begin{cases} Z_1\left(\mathbf{x}, R(\boldsymbol{\theta}_1)\right), & \mathbf{x} \in A_1 \\ Z_2\left(\mathbf{x}, R(\boldsymbol{\theta}_2)\right), & \mathbf{x} \in A_2 \end{cases} \tag{26}$$

where $\mathbf{x} = \{x_1, x_2\}$ represents the spatial coordinates in the 2D domain, $Z_1$ (resp. $Z_2$) are realizations of a Gaussian process characterized by a correlation function with length scales $\boldsymbol{\theta}_1 = \{\theta_{11}, \theta_{12}\}$ (resp. $\boldsymbol{\theta}_2 = \{\theta_{21}, \theta_{22}\}$).

A Kriging surrogate model will be calculated using the following correlation function:

$$R\left(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}\right) = \begin{cases} R\left(\mathbf{x}, \mathbf{x}'; \hat{\boldsymbol{\theta}}_1\right), & \left(\mathbf{x}, \mathbf{x}'\right) \in A_1 \times A_1 \\ R\left(\mathbf{x}, \mathbf{x}'; \hat{\boldsymbol{\theta}}_2\right), & \left(\mathbf{x}, \mathbf{x}'\right) \in A_2 \times A_2 \\ 0 & , \quad \text{otherwise} \end{cases} \tag{27}$$
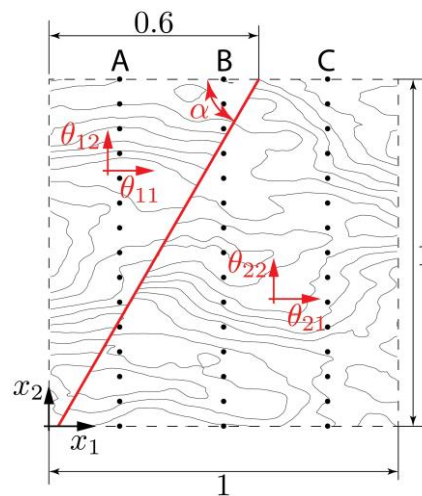


**Fig. 7.** Graphical visualization of the subsurface model. The unknowns (length scales of each random field and the fault angle) are denoted by red color.

Where $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, a\}$. There is a smooth dependence on $x_1, x_2$ within each region, but no correlation between points that belong to different regions. The boundary between the two regions is fully defined by the crack angle, $\alpha$, which is unknown and the fault location that is assumed to be known ($\{x_1, x_2\} = \{0.6, 1\}$). The goal here is to use Kriging to interpolate the measurements taken at borehole locations A, B, and C and estimate the five unknown parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, a\}$. The correlation function of each region is the same, both in the true model and the Kriging surrogate, *i.e.,* it is assumed to be known. In particular, the correlation function is separable Matérn 3/2 (see Eq. (13) and Table 2). The maximum-likelihood method is selected for estimating $\boldsymbol{\theta}$. Due to the complexity of the underlying optimization problem a hybrid genetic algorithm with relatively large population size and a maximum number of generations is selected.

A Matlab implementation of the correlation function in Eq. (27) is given in Appendix A. This Matlab function is called `my_eval_R` in the following code snippet.
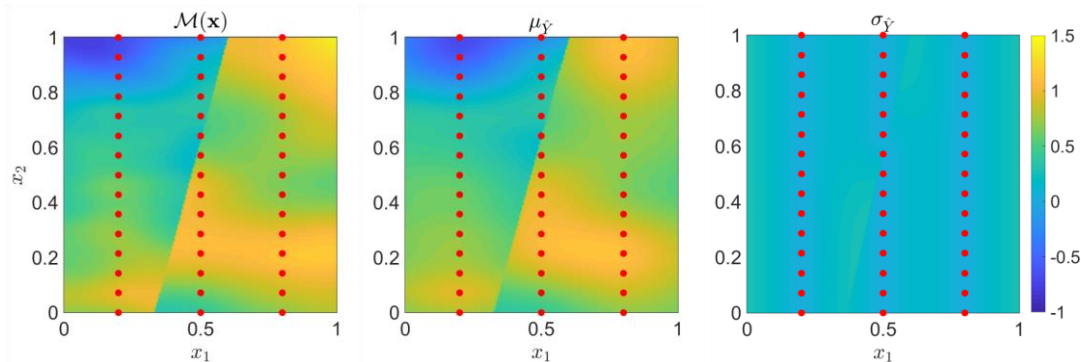


Figure 8: From left to right: The true permeability of the soil, followed by the mean and standard deviation of the Kriging predictor. Red dots illustrate the experimental design.

The Kriging surrogate is created next, based on a limited set of observations contained in the variables `BoreholeLocations` and `BoreValues`, which contain the locations of the measurements along the boreholes and the value of the desired property, respectively.

```
KOptions.Type = 'Metamodel';
KOptions.MetaType = 'Kriging';
KOptions.ExpDesign.X = BoreholeLocations;
KOptions.ExpDesign.Y = BoreValues;
KOptions.Corr.Handle = @my_eval_R;
% Add upper and lower bounds on the optimization variables
BoundsL = [0.3 0.1 0.3 0.1 pi/6] ;
BoundsU = [0.9 0.5 0.9 0.5 5*pi/6] ;

KOptions.Optim.Bounds =[BoundsL ;BoundsU];
KOptions.Optim.Method = 'HGA';
KOptions.Optim.HGA.nPop = 60;
KOptions.Optim.MaxIter = 50;
KOptions.EstimMethod = 'ML';
KOptions.Scaling = False;
myKriging = uq_createModel(KOptions);
```

Once the Kriging metamodel has been computed, the mean and standard deviation of the Kriging predictor can be quickly visualized for 1D and 2D models using the `uq_display` command, which produces a plot similar to Figure 8, except in a smaller domain determined by the range of the points in the experimental design. A comparison between the true and the estimated values, $\theta$ is given in Table 4. As expected, the accuracy of the hyperparameters estimation is low due to the limited dispersion of the experimental design. The error of the length scale estimates along the $x_1$ direction is consistently larger due to the lack of samples along that direction. From a coding perspective, although the correlation function that is used is relatively complex, it is straightforward to use in a Kriging surrogate once coded as a Matlab function (by setting the `KOptions.Corr.Handle` value appropriately). Moreover, custom correlation functions are allowed to have an arbitrary number of hyperparameters. The only requirement is that the optimisation bounds (or initial value, depending on the optimisation method that is used) must have the same length as the number of the hyperparameters.

**Table 4**
Listing of the true and estimated correlation function parameters, $\theta$, for the Kriging surrogate of the subsurface model.

| Parameter | $\theta_{11}$ | $\theta_{12}$ | $\theta_{21}$ | $\theta_{22}$ | $\alpha$ |
|---|---|---|---|---|---|
| True value | 0.600 | 0.250 | 0.900 | 0.350 | 1.309 |
| Estimated value | 0.310 | 0.271 | 0.310 | 0.374 | 1.342 |
| Relative error (%) | 48.3 | 8.2 | 65.6 | 6.9 | 2.5 |

## 5. Summary and outlook

In this paper, the GP-module of the UQLab software framework was presented. This UQLab module enables practitioners from various disciplines to get started with Kriging metamodelling with minimal effort as was illustrated in the introductory application in Section 4.1. However, it is also possible to access more advanced customization, *e.g.,* for research purposes. This was showcased in Section 4.2 where a hierarchical Kriging metamodel was developed and in Section 4.3 where a relatively complex, non-stationary correlation function was used to solve a geostatistical inverse problem. The GP-module is freely available to the academic community since the first beta release of UQLab in July 2015.

The current version of the GP-module only allows for computing Kriging models on noisy data by explicitly providing the noise level via the nugget effect. The general case where the noise level is unknown and needs to be estimated (*a.k.a.* Gaussian process regression) is currently under development and will be addressed in an upcoming release. In addition, the current version of the GP-module relies on additional Matlab toolboxes for performing the hyperparameter optimization. This may be a limiting factor for some users.

In addition to the modules currently exploiting its functionality (Polynomial Chaos-Kriging and Reliability analysis [45,50]), new UQLab modules that interface with the GP-module are

currently under active development. The upcoming random fields module will offer several random field types (conditional and unconditional) together with advanced sampling methodologies and will be interfaced with the GP-module to offer trajectory resampling capabilities. Similarly, the upcoming Reliability-Based Design Optimisation (RBDO) module uses the surrogate modeling capabilities of the GP-module for solving RBDO problems as described in Moustapha et al. [22].

## Appendix A. Kriging with custom correlation function: implementation details

This section aims to provide some additional implementation details on the application example in Section 4.3, in terms of the Matlab code involved. The correlation function described in Eq. (27) can be translated to the following Matlab function:

```matlab
function R = my_eval_R( x1,x2,theta,parameters )
xc = 0.6; % the x-location of the crack on the surface
yc = 1 ;  % the y-location of the crack on the surface
length_scales_1 = theta(1:2);
length_scales_2 = theta(3:4);
crack_angle = theta(5) ;
% find the angles of each sample of x1
angles_x1 = acos( (xc - x1(:,1))./sqrt((x1(:,1) - xc).^2 + ...
(x1(:,2) - yc).^2 ) );
% find the indices of x1 that belong to first region
idx_x1_1 = angles_x1 <= crack_angle;
% find the indices of x1 that belong to second region
idx_x1_2 = ~idx_x1_1;
% find the angles of each sample of x2
angles_x2 = acos( (xc - x2(:,1))./sqrt((x2(:,1) - xc).^2 + ...
(x2(:,2) - yc).^2 ) );
% find the indices of x2 that belong to first region
idx_x2_1 = angles_x2 <= crack_angle;
% find the indices of x2 that belong to second region
idx_x2_2 = ~idx_x2_1;
% set-up various correlation function options so that we can re-use the
% build-in UQLab function for evaluating R in each region
CorrOptions.Type = 'separable';
CorrOptions.Family = 'Matern-3_2';
CorrOptions.Isotropic = false;
CorrOptions.Nugget = 1e-2;
% initialize R matrix
R = zeros(size(x1,1), size(x2,1));
% Compute the R values in region 1
R(idx_x1_1,idx_x2_1) = uq_Kriging_eval_R( x1(idx_x1_1,:),
x2(idx_x2_1,:),...
length_scales_1, CorrOptions);
% Compute the R values in region 2
R(idx_x1_2,idx_x2_2) = uq_Kriging_eval_R( x1(idx_x1_2,:),
x2(idx_x2_2,:),...
length_scales_2, CorrOptions);
end
```

The provided code, although vectorized, is optimized for readability and not performance. To that end, the internal function of the GP-module `uq_Kriging_eval_R` is used for calculating the correlation function value in each of the regions.

## Acknowledgments

## References

[1]    Lemaire M, Chateauneuf A, Mitteau J-C. Structural Reliability. London, UK: ISTE; 2009. doi:10.1002/9780470611708.

[2]    Saltelli A, Chan K, Scott EM. Sensitivity Analysis. J. Wiley & Sons; 2000.

[3]    Tsompanakis Y, Lagaros ND, Papadrakakis M. Structural Design Optimization Considering Uncertainties. CRC Press; 2008.

[4]    Dashti M, Stuart AM. The Bayesian Approach to Inverse Problems. Handb. Uncertain. Quantif., Cham: Springer International Publishing; 2017, p. 311–428. doi:10.1007/978-3-319-12385-1_7.

[5]    Santner TJ, Williams BJ, Notz WI. The Design and Analysis of Computer Experiments. New York, NY: Springer New York; 2003. doi:10.1007/978-1-4757-3799-8.

[6]    Fang K-T, Li R, Sudjianto A. Design and Modeling for Computer Experiments. Chapman and Hall/CRC; 2005. doi:10.1201/9781420034899.

[7]    Forrester AIJ, Sbester A, Keane AJ. Engineering Design via Surrogate Modelling. Chichester, UK: John Wiley & Sons, Ltd; 2008. doi:10.1002/9780470770801.

[8]    Sacks J, Welch WJ, Mitchell TJ, Wynn HP. Design and analysis of computer experiments. Stat Sci 1989;4:409–23.

[9]    Ghanem RG, Spanos PD. Stochastic Finite Elements: A Spectral Approach. New York, NY: Springer New York; 1991. doi:10.1007/978-1-4612-3094-6.

[10]   Xiu D, Karniadakis GE. The Wiener--Askey Polynomial Chaos for Stochastic Differential Equations. SIAM J Sci Comput 2002;24:619–44. doi:10.1137/S1064827501387826.

[11]   Vapnik VN. The Nature of Statistical Learning Theory. New York, NY: Springer New York; 2000. doi:10.1007/978-1-4757-3264-1.

[12]   Krige DG. A statistical approach to some basic mine valuation problems on the Witwatersrand. J South African Inst Min Metall 1951;52:119–39.

[13]   Welch WJ, Buck RJ, Sacks J, Wynn HP, Mitchell TJ, Morris MD. Screening, Predicting, and Computer Experiments. Technometrics 1992;34:15–25. doi:10.1080/00401706.1992.10485229.

[14]    Marrel A, Iooss B, Van Dorpe F, Volkova E. An efficient methodology for modeling complex computer codes with Gaussian processes. Comput Stat Data Anal 2008;52:4731–44. doi:10.1016/j.csda.2008.03.026.

[15]    Gaspar B, Teixeira AP, Soares CG. Assessment of the efficiency of Kriging surrogate models for structural reliability analysis. Probabilistic Eng Mech 2014;37:24–34. doi:10.1016/j.probengmech.2014.03.011.

[16]    Iooss B, Lemaître P. A Review on Global Sensitivity Analysis Methods, 2015, p. 101–22. doi:10.1007/978-1-4899-7547-8_5.

[17]    Gratiet L Le, Marelli S, Sudret B. Metamodel-Based Sensitivity Analysis: Polynomial Chaos Expansions and Gaussian Processes. Handb. Uncertain. Quantif., Cham: Springer International Publishing; 2015, p. 1–37. doi:10.1007/978-3-319-11259-6_38-1.

[18]    Moustapha M, Bourinet J-M, Guillaume B, Sudret B. Comparative Study of Kriging and Support Vector Regression for Structural Engineering Applications. ASCE-ASME J Risk Uncertain Eng Syst Part A Civ Eng 2018;4:04018005. doi:10.1061/AJRUA6.0000950.

[19]    Echard B, Gayton N, Lemaire M. AK-MCS: An active learning reliability method combining Kriging and Monte Carlo Simulation. Struct Saf 2011;33:145–54. doi:10.1016/j.strusafe.2011.01.002.

[20]    Dubourg V, Sudret B. Meta-model-based importance sampling for reliability sensitivity analysis. Struct Saf 2014;49:27–36. doi:10.1016/j.strusafe.2013.08.010.

[21]    Simpson TW, Mauery TM, Korte JJ, Mistree F. Kriging Models for Global Approximation in Simulation-Based Multidisciplinary Design Optimization. AIAA J 2001;39:2233–41. doi:10.2514/2.1234.

[22]    Moustapha M, Sudret B, Bourinet J-M, Guillaume B. Quantile-based optimization under uncertainties using adaptive Kriging surrogate models. Struct Multidiscip Optim 2016;54:1403–21. doi:10.1007/s00158-016-1504-4.

[23]    Bachoc F, Bois G, Garnier J, Martinez J-M. Calibration and Improved Prediction of Computer Models by Universal Kriging. Nucl Sci Eng 2014;176:81–97. doi:10.13182/NSE12-55.

[24]    Rasmussen CE, Williams CK. Gaussian processes for machine learning (adaptive computation and machine learning) Cambridge, Cambridge, MA, USA: MIT Press 2005.

[25]    Deutsch C V, Journel AG. GSLIB: Geostatistical SoftwareLibrary and User's Guide. New York, US: Oxford University Press; 1992.

[26]    Roustant O, Ginsbourger D, Deville Y. Dicekriging, Diceoptim: Two R packages for the analysis of computer experiments by kriging-based metamodelling and optimization. J Stat Softw 2012;51:1–55.

[27]    Dupuy D, Helbert C, Franco J. DiceDesign and DiceEval: Two R packages for design and analysis of computer experiments. J Stat Softw 2015;65:1–38.

[28]    Picheny V, Ginsbourger D, Roustant O. DiceOptim: Kriging-Based Optimization for Computer Experiments. R package version 0.8-1 2016.

[29] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in Python. J Mach Learn Res 2011;12:2825–30.

[30] Paulson C, Ragkousis G. pyKriging: A Python Kriging Toolkit [Data set]. 2015.

[31] GPy: A gaussian process framework in python 2012.

[32] Lophaven SN, Nielsen HB, Sondergaard J. Aspects of the matlab toolbox DACE. Technical report, Informatics and Mathematical Modelling. 2002.

[33] Couckuyt I, Dhaene T, Demeester P. ooDACE toolbox: a flexible object-oriented Kriging implementation. J Mach Learn Res 2014;15:3183–6.

[34] Bect J, Vazquez E. STK: a Small (Matlab/Octave) Toolbox for Kriging 2014.

[35] Rasmussen CE, Nickisch H. Gaussian processes for machine learning (GPML) toolbox. J Mach Learn Res 2010;11:3011–5.

[36] Marelli S, Sudret B. UQLab: A Framework for Uncertainty Quantification in Matlab. Vulnerability, Uncertainty, Risk (Proc. 2nd Int. Conf. Vulnerability, Risk Anal. Manag. (ICVRAM2014), Liverpool, United Kingdom), Reston, VA: American Society of Civil Engineers; 2014, p. 2554–63. doi:10.1061/9780784413609.257.

[37] Schöbi R, Sudret B, Wiart J. Polynomial-chaos-based Kriging. Int J Uncertain Quantif 2015;5:171–93.

[38] Stein ML. Interpolation of Spatial Data. New York, NY: Springer New York; 1999. doi:10.1007/978-1-4612-1494-6.

[39] Bachoc F. Cross Validation and Maximum Likelihood estimations of hyper-parameters of Gaussian processes with model misspecification. Comput Stat Data Anal 2013;66:55–69. doi:10.1016/j.csda.2013.03.016.

[40] Dubrule O. Cross validation of kriging in a unique neighborhood. J Int Assoc Math Geol 1983;15:687–99. doi:10.1007/BF01033232.

[41] Cressie NAC. Statistics for Spatial Data. Hoboken, NJ, USA: John Wiley & Sons, Inc.; 1993. doi:10.1002/9781119115151.

[42] Sudret B. Uncertainty propagation and sensitivity analysis in mechanical models–Contributions to structural reliability and stochastic spectral methods. 2007.

[43] De Rocquigny E, Devictor N, Tarantola S. Uncertainty in industrial practice: a guide to quantitative uncertainty management. John Wiley & Sons; 2008.

[44] Lataniotis C, Marelli S, Sudret B. UQLab user manual–Kriging (Gaussian process modelling). 2017.

[45] Marelli S, Schöbi R, Sudret B. Uqlab user manual - structural reliability. Technical report, Chair of Risk, Safety and Uncertainty Quantification, ETHZurich. Report UQLab-V0.92-107; 2017.

[46] Jones DR, Schonlau M, Welch WJ. Efficient Global Optimization of Expensive Black-Box Functions. J Glob Optim 1998;13:455–92. doi:10.1023/A:1008306431147.

[47]   McKay MD, Beckman RJ, Conover WJ. Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. Technometrics 1979;21:239–45. doi:10.1080/00401706.1979.10489755.

[48]   Han Z, Zimmerman R, Görtz S. Alternative Cokriging Method for Variable-Fidelity Surrogate Modeling. AIAA J 2012;50:1205–10. doi:10.2514/1.J051243.

[49]   Abdallah I, Sudret B, Lataniotis C, Sørensen JD, Natarajan A. Fusing simulation results from multifidelity aero-servo-elastic simulators-Application to extreme loads on wind turbine. Proc. 12th Int. Conf. Appl. Stat. Probab. Civ. Eng. (ICASP12), Vancouver, Canada, July 12-15, University of British Columbia; 2015.

[50]   Schöbi R, Marelli S, Sudret B. Uqlab user manual–pc-kriging, Technical report, Chair of Risk, Safety & Uncertainty Quantification, ETH Zurich. Report UQLab-V1. 0109. Technical report, Chair of Risk, Safety & Uncertainty Quantification, ETH Zurich; 2017.