ETH zürich

UQLab user manual
Kriging (Gaussian process modelling)

C. Lataniotis, S. Marelli, B. Sudret

Chair of Risk, Safety and Uncertainty Quantification
Stefano-Franscini-Platz 5
CH-8093 Zürich

Risk, Safety & Uncertainty Quantification

How to cite UQLab

S. Marelli, and B. Sudret, UQLab: A framework for uncertainty quantification in Matlab, Proc. 2nd Int. Conf. on Vulnerability, Risk Analysis and Management (ICVRAM2014), Liverpool, United Kingdom, 2014, 2554-2563.

How to cite this manual

C. Lataniotis, S. Marelli and B. Sudret, UQLab user manual – Kriging (Gaussian process modelling), Report UQLab-V1.2-105, Chair of Risk, Safety & Uncertainty Quantification, ETH Zurich, 2019.

BibTeX entry

```
@TECHREPORT{UQdoc_12_105,
author = {Lataniotis, C. and Marelli, S. and Sudret, B.},
title = {{UQLab user manual -- Kriging (Gaussian process modelling)}},
institution = {Chair of Risk, Safety \& Uncertainty Quantification, ETH Zurich},
year = {2019},
note = {Report \# UQLab-V1.2-105}
}
```

Document Data Sheet

Document Ref.	UQLab-V1.2-105
Title:	UQLab user manual – Kriging (Gaussian process modelling)
Authors:	C. Lataniotis, S. Marelli, B. Sudret Chair of Risk. Safety and Uncertainty Quantification, ETH Zurich, Switzerland
Date:	22/02/2019

Doc. Version	Date	Comments
V1.2	22/02/2019	UQLab V1.2 release • added automatic calculation of validation error if a validation set is provided
V1.1	05/07/2018	UQLab V1.1 release • Minor updates in the reference list
V1.0	01/05/2017	UQLab V1.0 release • Minor consistency fixes in the reference list • New usage sections regarding scaling and sampling from a Gaussian process posterior
V0.9	01/07/2015	Initial release

Abstract

Kriging is a stochastic interpolation algorithm that has applications in various engineering and applied mathematics fields. UQLab metamodelling tools provide an efficient, modular and easy to use Kriging module that allows one to obtain efficient Kriging predictors with minimal effort. For the experienced users there are numerous configuration options that can be easily set. This guide includes an extensive manual of the Kriging metamodelling module that is divided into three parts:

- A short introduction to the main concepts and techniques behind Kriging, with a selection of references to the relevant literature
- A detailed example-based guide, with explanation of most of the available options and methods
- A comprehensive reference list of all the available functionalities of the module in UQLab.

Keywords: UQLab, Kriging, Gaussian process metamodelling

Contents

Chapter 1

Theory

1.1 Kriging basics

Kriging (a.k.a. Gaussian process modelling) is a stochastic interpolation algorithm which assumes that the model output $\mathcal{M}(\boldsymbol{x})$ is a realization of a Gaussian process indexed by $\boldsymbol{x} \in \mathcal{D}_X \subset \mathbb{R}^M$. A Kriging metamodel is described by the following equation (Santner et al., 2003):

$$\mathcal{M}^K(\boldsymbol{x}) = \boldsymbol{\beta}^T \boldsymbol{f}(\boldsymbol{x}) + \sigma^2 Z(\boldsymbol{x}, \omega) \tag{1.1}$$

The first term in Eq. (1.1), $\boldsymbol{\beta}^T \boldsymbol{f}(\boldsymbol{x})$, is the mean value of the Gaussian process (a.k.a. *trend*) and it consists of the regression coefficients $\{\beta_j, j = 1, \ldots, P\}$ and the basis functions $\{f_j, j = 1, \ldots, P\}$. The second term in Eq. (1.1) consists of σ^2, the (constant) variance of the Gaussian process and $Z(\boldsymbol{x}, \omega)$, a zero mean, unit variance, stationary Gaussian process. The underlying probability space is represented by ω and is defined in terms of a *correlation function* R (a.k.a. correlation family) and its hyperparameters $\boldsymbol{\theta}$. The correlation function $R = R(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta})$ describes the correlation between two samples of the input space, *e.g.*$\boldsymbol{x}$ and $\boldsymbol{x}'$ and depends on the hyperparameters $\boldsymbol{\theta}$.

In the context of metamodelling, it is of interest to calculate a prediction $\mathcal{M}^K(x)$ for a new point $\boldsymbol{x}$, given $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, the experimental design and $\boldsymbol{y} = \{y_1 = \mathcal{M}(\boldsymbol{x}_1), \ldots, y_n = \mathcal{M}(\boldsymbol{x}_n)\}^T$, the corresponding (noise-free) model responses. A Kriging metamodel (a.k.a. Kriging predictor) provides such predictions based on the Gaussian properties of the process.

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

$$\begin{Bmatrix} \hat{Y}(\boldsymbol{x}) \\ \boldsymbol{y} \end{Bmatrix} \sim \mathcal{N}_{N+1} \left(\begin{Bmatrix} \boldsymbol{f}^T(\boldsymbol{x})\boldsymbol{\beta} \\ \boldsymbol{F}\boldsymbol{\beta} \end{Bmatrix}, \sigma^2 \begin{Bmatrix} 1 & \boldsymbol{r}^T(\boldsymbol{x}) \\ \boldsymbol{r}(\boldsymbol{x}) & \boldsymbol{R} \end{Bmatrix} \right) \tag{1.2}$$

where:

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

$$F_{ij} = f_j(\boldsymbol{x}_i)\ ,\ i = 1, \ldots, N,\ j = 1, \ldots, P$$

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

$$\mathbf{r}_i = R(\boldsymbol{x}, \boldsymbol{x}_i; \boldsymbol{\theta}),\ i = 1, \ldots, N \tag{1.3}$$

$\boldsymbol{R}$ is the correlation matrix whose terms read:

$$R_{ij} = R(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\theta}),\ i, j = 1, \ldots, N \tag{1.4}$$

Consequently, the mean and variance of the Gaussian random variate $\hat{Y}(\boldsymbol{x})$ (a.k.a. mean and variance of the Kriging predictor) can be calculated (Santner et al., 2003; Dubourg, 2011):

$$\mu_{\hat{Y}}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x})^T\boldsymbol{\beta} + \boldsymbol{r}(\boldsymbol{x})^T\boldsymbol{R}^{-1}\left(\boldsymbol{y} - \boldsymbol{F}\boldsymbol{\beta}\right) \tag{1.5}$$

$$\sigma^2_{\hat{Y}}(\boldsymbol{x}) = \sigma^2\left(1 - \boldsymbol{r}^T(\boldsymbol{x})\boldsymbol{R}^{-1}\boldsymbol{r}(\boldsymbol{x}) + \boldsymbol{u}^T(\boldsymbol{x})(\boldsymbol{F}^T\boldsymbol{R}^{-1}\boldsymbol{F})^{-1}\boldsymbol{u}(\boldsymbol{x})\right) \tag{1.6}$$

where:

$$\boldsymbol{\beta} = \left(\boldsymbol{F}^T\boldsymbol{R}^{-1}\boldsymbol{F}\right)^{-1}\boldsymbol{F}^T\boldsymbol{R}^{-1}\boldsymbol{y} \tag{1.7}$$

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

$$\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{F}^T\boldsymbol{R}^{-1}\boldsymbol{r}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x}) \tag{1.8}$$

Another useful corollary of the Gaussian assumption is that:

$$\hat{Y}(\boldsymbol{x}) \sim \mathcal{N}\left(\mu_{\hat{Y}}(\boldsymbol{x}), \sigma^2_{\hat{Y}}(\boldsymbol{x})\right) \tag{1.9}$$

Thus:

$$\mathcal{P}\left[\hat{Y}(\boldsymbol{x}) \le t\right] = \Phi\left(\frac{t - \mu_{\hat{Y}}(\boldsymbol{x})}{\sigma_{\hat{Y}}(\boldsymbol{x})}\right) \tag{1.10}$$

where $\Phi(\cdot)$ denotes the Gaussian cumulative density function. Note that this is the probability that $\hat{Y}(\boldsymbol{x})$ is smaller than t *at a given* $\boldsymbol{x}$. This shall not be confused with a probability of failure computed in the framework of rare event simulation (see UQLab User Manual – Structural reliability (Rare event estimation)). Based on Eq. (1.10) the confidence intervals on the predictor can be calculated by:

$$\hat{Y}(\boldsymbol{x}) \in \left[\mu_{\hat{Y}}(\boldsymbol{x}) - \Phi^{-1}\left(1 - \frac{\alpha}{2}\right)\sigma_{\hat{Y}}(\boldsymbol{x})\ ,\ \mu_{\hat{Y}}(\boldsymbol{x}) + \Phi^{-1}\left(1 - \frac{\alpha}{2}\right)\sigma_{\hat{Y}}(\boldsymbol{x})\right] \tag{1.11}$$

with probability $1 - \alpha$.

In practice, in order to obtain a Kriging metamodel, the following steps are needed:

- Select the functional basis of the Kriging trend. This is further discussed in Section 1.2.
- Select the appropriate correlation function $R(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta})$. This is further discussed in Section 1.3.
- If the hyperparameters $\boldsymbol{\theta}$ and the Gaussian process variance σ^2 are unknown (which is usually the case) get an estimate of those. This involves setting up an optimization

problem (see Section 1.4) and solving it (see Section 1.5).

- Using the optimal value of $\boldsymbol{\theta}$, the rest of the unknown Kriging parameters $(\sigma^2, \boldsymbol{\beta})$ can be calculated.

Then predictions for new points can be made in terms of the mean and variance of $\hat{Y}(\boldsymbol{x})$, using Eqs. (1.5), (1.6).
A one-dimensional Kriging example is shown in Figure 1. Given a set of observations of some model, a Kriging predictor returns the mean and the variance of a Gaussian process that interpolates them. Using Eq. (1.11), the 95% confidence interval of the predictor is also calculated.

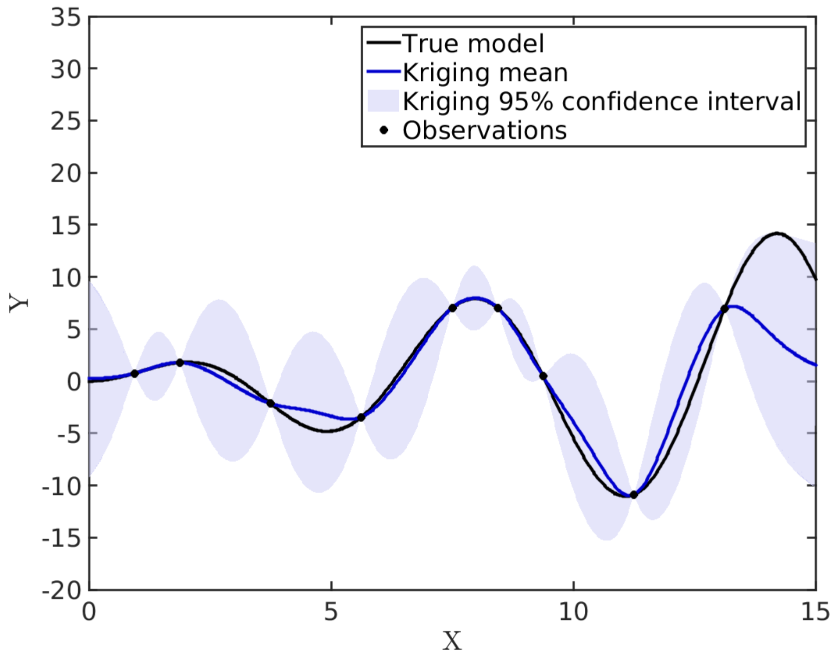


Figure 1: An example of a Kriging metamodel.

1.2 Trend types

The trend refers to the mean of the Kriging metamodel, that is the term $\boldsymbol{\beta}^T \boldsymbol{f}(\boldsymbol{x})$ in Eq. (1.1). Using a non-zero trend is optional but is however commonly preferred. Note that the mean of the Kriging predictor in Eq. (1.5) is not confined to be zero when the trend is zero.
In the literature, a different naming is given to the Kriging metamodel depending on the type of trend that is used, (Dubourg, 2011), namely:

- *Simple Kriging*: The trend has a *known* constant value:

$$\boldsymbol{\beta}^T \boldsymbol{f}(\boldsymbol{x}) = \beta_0$$

Notice that no regression takes place in this case since the trend is known.

- *Ordinary Kriging*: The trend has a constant yet unknown value:

$$\boldsymbol{\beta}^T \boldsymbol{f}(\boldsymbol{x}) = \beta_1 f_1(\boldsymbol{x}) = \beta_1$$

since $f_1(\boldsymbol{x}) = 1$.

- *Universal Kriging*: This is the most general and flexible formulation. It assumes that the trend is a linear combination of prescribed functions, *e.g.*polynomials:

$$\boldsymbol{\beta}^T \boldsymbol{f}(\boldsymbol{x}) = \sum_{j=1}^{P} \beta_j f_j(\boldsymbol{x})$$

From the above it is clear that simple and ordinary Kriging are special cases of universal Kriging. In the current version of UQLAB, the basis functions can be either constants or polynomials or any other user-defined function.

1.2.1 Commonly used trends

The most commonly used trends are summarized in Table 1.

Table 1: Trend options combinations

Trend	Formula
constant (ordinary Kriging)	β_0
linear	$\beta_0 + \sum\limits_{i=1}^{M} \beta_i x_i$
quadratic	$\beta_0 + \sum\limits_{i=1}^{M} \beta_i x_i + \Sigma_{i=1}^{M} \Sigma_{j=1}^{M} \beta_{ij} x_i x_j$

The multivariate polynomial trends that are given in Table 1 can be written in a more compact and general form as follows:

$$\boldsymbol{F}(\boldsymbol{x}) = \sum_{\alpha \in \boldsymbol{\mathcal{A}}^{\boldsymbol{M},\boldsymbol{P}}} \beta_\alpha f_\alpha(\boldsymbol{x}) \, , \; f_\alpha(\boldsymbol{x}) = \prod_{i=1}^{M} x^{\alpha_i} \tag{1.12}$$

where $\boldsymbol{\alpha} = \{\alpha_1, \dots, \alpha_M\}$ is a vector of indices and $\mathcal{A}^{M,P} = \{\boldsymbol{\alpha} : |\boldsymbol{\alpha}| \leq P\}$ denotes the set of indices that correspond to all polynomials in the M input variables up to degree P.

The Kriging module in UQLAB offers the possibility to use as a trend a multivariate polynomial of arbitrary degree P (as described in Eq. (1.12)) as well as different types of basis functions. For more information see Section 2.4.2 and Section 3.1.2 for a list of all the available options regarding the trend.

1.3 Correlation functions

The correlation function (also called *kernel* or *covariance*[1] function in the literature) is a crucial ingredient for a Kriging predictor, since it contains the assumptions about the function we want to learn. Generally speaking, it describes the ”similarity” between observations and new points, i.e. how similar such points are, depending on their distance. Close input points in that sense are then expected to have similar outputs y.
An arbitrary function of $(\boldsymbol{x}, \boldsymbol{x}')$ is in general not a valid correlation function. The necessary conditions for a function $R(\boldsymbol{x}, \boldsymbol{x}')$ to be a correlation function are:

- The correlation (or Gram) matrix whose elements are computed as $R_{ij} = R(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{X} \times \mathcal{X}$ is positive semi-definite (PSD) for any choice of number of samples N and experimental design $\mathcal{X}$.
- It is *symmetric*, i.e. $R(\boldsymbol{x}, \boldsymbol{x}') = R(\boldsymbol{x}', \boldsymbol{x})$, $\forall \boldsymbol{x}', \boldsymbol{x} \in \mathcal{D}_X$

In general it can be expressed in the form $R(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta})$ where $\boldsymbol{\theta}$ is a vector that contains a set of parameters. Typically $\boldsymbol{\theta} \in \mathbb{R}^M$ yet this is not necessarily true in the general case, since more than one element can correspond to each input dimension. In the current stage, it is assumed however that one element of $\boldsymbol{\theta}$ is used per dimension for notational clarity.

1.3.1 Correlation function types

When the input dimension M is greater than 1 the following different types of correlation functions exist in the current version of UQLab:

- *Ellipsoidal* correlation functions (Rasmussen and Williams, 2006). They are calculated as follows:

$$R(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}) = R(h) \ , \ h = \sqrt{\sum_{i=1}^{M} \left(\frac{x_i - x_i'}{\theta_i} \right)^2} \tag{1.13}$$

- *Separable* correlation functions (Sacks et al., 1989; Dubourg, 2011). They are calculated as

$$R(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}) = \prod_{i=1}^{M} R(x_i, x_i', \theta_i) \tag{1.14}$$

The function $R(\cdot)$ that appears on the right hand side of Eqs. (1.13), (1.14) corresponds to one-dimensional correlation functions also known as *correlation families*. The ones that are currently available in UQLab can be found in Section 1.3.4.

[1]The covariance function corresponds to the correlation function multiplied by the variance of the Gaussian process σ^2.

1.3.2 Isotropic correlation functions

The expressions of the different correlation functions that were given in Section 1.3.1 correspond to the *anisotropic* case. Generally speaking, a correlation function is called *isotropic* when it has the same behavior over all dimensions. In that sense, for each type of correlation function, the isotropy is defined as follows:

- Isotropic ellipsoidal correlation function:

$$R(\boldsymbol{x}, \boldsymbol{x}'; \theta) = R\left(\frac{1}{\theta}\sqrt{\sum_{i=1}^{M}(x_i - x_i')^2}\right) , \ \theta \in \mathbb{R} \tag{1.15}$$

- Isotropic separable correlation function:

$$R(\boldsymbol{x}, \boldsymbol{x}'; \theta) = \prod_{i=1}^{M} R(x_i, x_i'; \theta) , \ \theta \in \mathbb{R} \tag{1.16}$$

1.3.3 Nugget and numerical stability

Regardless on how the correlation matrix $\boldsymbol{R}$ is calculated, it is often the case that it needs to be inverted at various stages of the Kriging process (see, *e.g.* Eq. (1.5) to (1.8)). This is well known to suffer from numerical instabilities, especially when the distances between the observations $\boldsymbol{x}_i$ are small. To offset this limitation, one can introduce a *nugget* $\boldsymbol{\nu}$, which is a set of values that are added to the main diagonal of $\boldsymbol{R}$:

$$R_{ii} = 1 + \nu_i. \tag{1.17}$$

1.3.4 Correlation families

For each of the available correlation families, the function is plotted in Figures 2-6 together with a realization of a zero-mean, unit variance Gaussian process having this correlation function.

- *Linear*.

  It is defined as :

$$R(h; \theta) = \max\left(0, 1 - \frac{|h|}{\theta}\right) \tag{1.18}$$

- *Exponential*.

  It is defined by

$$R(h; \theta) = \exp\left(-\frac{|h|}{\theta}\right) \tag{1.19}$$

  Its sample paths are $\mathcal{C}^0$, i.e. continuous but non-differentiable.

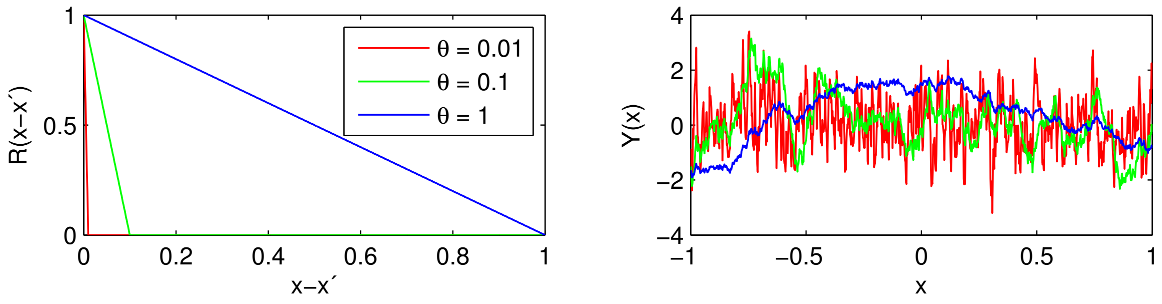


Figure 2: The linear correlation function and some sample paths of the corresponding zero-mean unit-variance Gaussian process for various scale parameters.

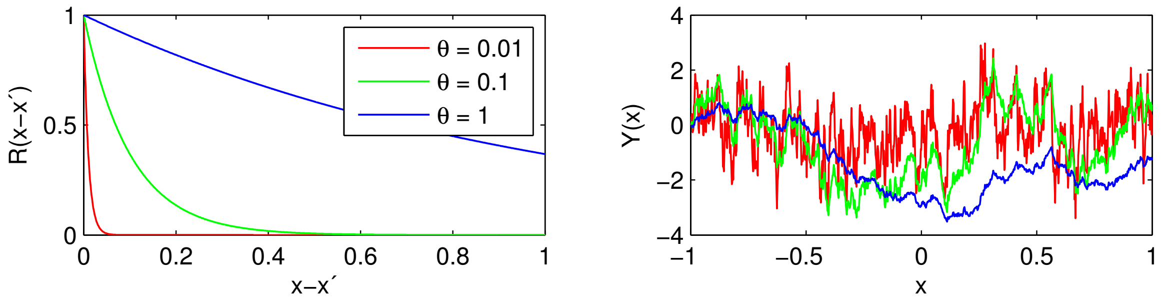


Figure 3: The exponential correlation function and some sample paths of the corresponding zero-mean unit-variance Gaussian process for various scale parameters.

- *Gaussian* (or squared exponential).

  It is defined by

  $$R(h;\theta) = \exp\left(-\frac{1}{2}\sum_{i=1}^{M}\left(\frac{h}{\theta}\right)^2\right) \tag{1.20}$$

  Its sample paths are infinitely differentiable.

- *Matérn*.

  The general form of the Matérn correlation function is given by

  $$R(h;\theta,v) = \frac{1}{2^{v-1}\Gamma(v)}\left(2\sqrt{v}\frac{|h|}{\theta}\right)^{v}\mathcal{K}_v\left(2\sqrt{v}\frac{|h|}{\theta}\right) \tag{1.21}$$

  where θ is the so-called scale parameter, $v \geq 1/2$ is the shape parameter, Γ is the Euler Gamma function and $\mathcal{K}_v$ is the modified Bessel function of the second kind (also known as the Bessel function of the third kind). For more information see Abramovitz and Stegun (1965)).

  An interesting feature of this family of correlation functions is that the sample paths of the corresponding Gaussian process are $\lceil v-1 \rceil$ times differentiable, where $\lceil\cdot\rceil$ denotes the ceiling function. For $v = 1/2$ Matérn kernel coincides with the exponential correlation function which generates $\mathcal{C}^0$ sample paths. For $v \to \infty$ its tends towards the Gaussian correlation function which generates $\mathcal{C}^\infty$ sample paths.

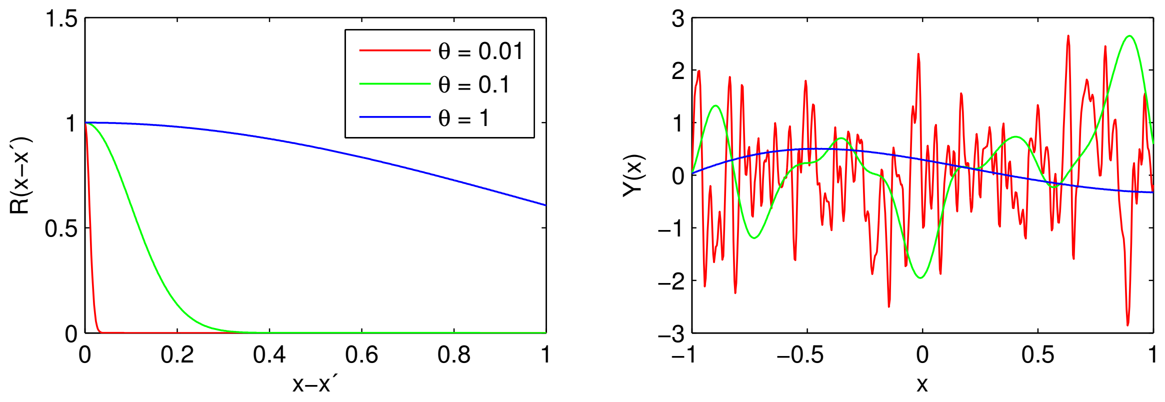


Figure 4: The Gaussian correlation function and some sample paths of the corresponding zero-mean unit-variance Gaussian process for various scale parameters.

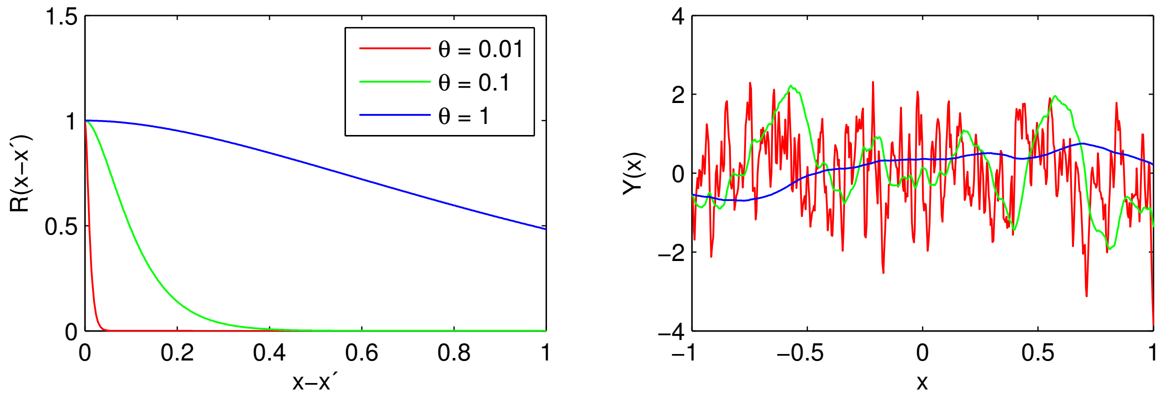


Figure 5: The Matérn $3/2$ correlation function and some sample paths of the corresponding zero-mean unit-variance Gaussian process for various scale parameters.

The Matérn functions can be calculated using simplified formulas when $v = p + 1/2$ where p is a non-negative integer (Rasmussen and Williams, 2006). So far, the Matérn functions with $v = 3/2$ and $v = 5/2$ (abbreviated as Matérn-3/2 and Matérn-5/2 respectively) have been implemented. These types are the most commonly used. Their formulas are given next. For $v = 3/2$:

$$R(h;\theta,v=3/2) = \left(1+\frac{\sqrt{3}|h|}{\theta}\right)\exp\left(-\frac{\sqrt{3}|h|}{\theta}\right) \tag{1.22}$$

and for $v = 5/2$:

$$R(h;\theta,v=5/2) = \left(1+\frac{\sqrt{5}|h|}{\theta}+\frac{5h^2}{3\theta^2}\right)\exp\left(-\frac{\sqrt{5}|h|}{\theta}\right) \tag{1.23}$$

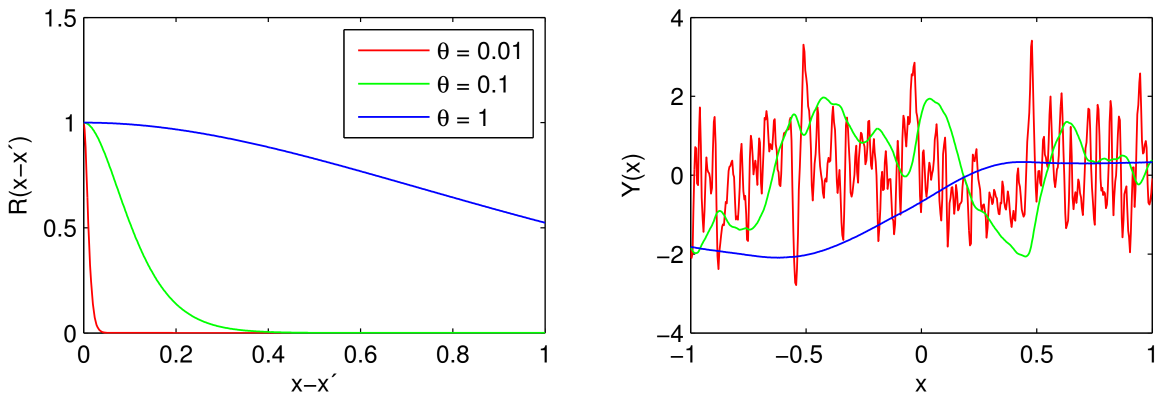


Figure 6: The Matérn $5/2$ correlation function and some sample paths of the corresponding zero-mean unit-variance Gaussian process for various scale parameters.

Note: When using one of the above correlation families to build an anisotropic ellipsoidal correlation function we treat them as $R = R(\overline{h})$ where $\overline{h} = \frac{|h|}{\theta}$.

1.4 Estimation methods

In order to obtain a Kriging metamodel as in Eq. (1.1), usually the hyperparameters $\boldsymbol{\theta}$ are unknown and need to be estimated. This is achieved by solving an optimization problem that differs depending on the estimation method that is used. The estimation methods that are available in UQLab are discussed in the following subsections.

1.4.1 Maximum Likelihood

The idea behind this method is to find the set of parameters $\boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta}$ such that the likelihood of the observations $\boldsymbol{y} = \{\mathcal{M}(\boldsymbol{x}_1), \dots, \mathcal{M}(\boldsymbol{x}_N)\}^T$ is maximal. Since $\boldsymbol{y}$ follows a multivariate Gaussian distribution (recall basic Kriging assumptions), the likelihood function $L(\mathbf{y} \mid \boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta})$ reads as follows:

$$L(\boldsymbol{y} \mid \boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta}) = \frac{\det(\boldsymbol{R})^{-1/2}}{(2\pi\sigma^2)^{N/2}} \exp\left[-\frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{F}\boldsymbol{\beta})^T \boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{F}\boldsymbol{\beta})\right] \tag{1.24}$$

By maximizing the quantity described in Eq. (1.24) one gets the following estimates of $\boldsymbol{\beta}$ and $\boldsymbol{\sigma}^2$ that are known as the generalized least-squares estimates (for proof and more details refer to Dubourg (2011); Santner et al. (2003)):

$$\boldsymbol{\beta} = \boldsymbol{\beta}(\boldsymbol{\theta}) = \left(\boldsymbol{F}^T \boldsymbol{R}^{-1} \boldsymbol{F}\right)^{-1} \boldsymbol{F}^T \boldsymbol{R}^{-1} \boldsymbol{y} \tag{1.25}$$

$$\sigma^2 = \sigma^2(\boldsymbol{\theta}) = \frac{1}{N} (\boldsymbol{y} - \boldsymbol{F}\boldsymbol{\beta})^T \boldsymbol{R}^{-1} (\boldsymbol{y} - \boldsymbol{F}\boldsymbol{\beta}) \tag{1.26}$$

However, they are dependent on the value of the hyperparameters $\boldsymbol{\theta}$ which are calculated by solving the optimization problem:

$$\boldsymbol{\theta} = \arg\min_{\mathcal{D}_{\boldsymbol{\theta}}} \left(-\log L(\boldsymbol{y} \mid \boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta})\right) \tag{1.27}$$

Based on equations (1.24), (1.26) the optimization problem in (1.27) can be written as follows:

$$\boldsymbol{\theta} = \arg\min_{\mathcal{D}_{\boldsymbol{\theta}}} \left(\frac{1}{2}\log(\det(\boldsymbol{R})) + \frac{N}{2}\log(2\pi\sigma^2) + \frac{N}{2}\right) \tag{1.28}$$

1.4.2 Cross-Validation

The general principle of this method (also known as K-fold cross-validation) is to split the whole dataset of observations $\mathcal{D}$ into K mutually exclusive and collectively exhaustive subsets $\{\mathcal{D}_k, k = 1, \ldots, K\}$ such that

$$\mathcal{D}_i \cap \mathcal{D}_j = \emptyset \ , \ \forall (i,j) \in \{1, \ldots, K\}^2 \ \text{ and } \bigcup_{k=1}^{K} \mathcal{D}_k = \mathcal{D} \tag{1.29}$$

The k-th set of cross-validated predictions is obtained by fitting the model using all the subsets but the k-th one and predicting it 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 this case the usual choice of objective function is (Santner et al., 2003; Bachoc, 2013):

$$\boldsymbol{\theta} = \arg\min_{\mathcal{D}_{\boldsymbol{\theta}}} \sum_{i=1}^{K} \left(\mathcal{M}(\boldsymbol{x}_i) - \mu_{\hat{Y},(-i)}(\boldsymbol{x}_i)\right)^2 \tag{1.30}$$

where $\mu_{\hat{Y},(-i)}(\boldsymbol{x}_i)$ is the mean Kriging predictor that was trained using all $\mathcal{X}, \boldsymbol{y}$ but $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ then evaluated at this point $\boldsymbol{x}_i$. 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.
The estimate of σ^2 is calculated using the following equation (Cressie, 1993; Bachoc, 2013):

$$\sigma^2 = \sigma^2(\boldsymbol{\theta}) = \frac{1}{K} \sum_{i=1}^{K} \frac{\left(\mathcal{M}(\boldsymbol{x}_i) - \mu_{\hat{Y},(-i)}(\boldsymbol{x}_i)\right)^2}{\sigma^2_{\hat{Y},(-i)}(\boldsymbol{x}_i)} \tag{1.31}$$

where $\sigma^2_{\hat{Y},(-i)}(\boldsymbol{x}_i)$ denotes the variance of a Kriging predictor that was trained using all $\mathcal{X}, \boldsymbol{y}$ apart from $\boldsymbol{x}_i, \boldsymbol{y}_i$, evaluated at point $\boldsymbol{x}_i$. When i is a set of indices then the division and the squared operations in Eq. (1.31) are performed element-wise.

1.5 Optimization methods

For solving the optimization problems described in (1.28) (maximum likelihood case) or (1.30) (cross-validation case) there are trade-offs for choosing some local (usually gradient-based) or global methods (usually evolutionary algorithms). On the one hand local methods tend to converge faster and require fewer objective function evaluations but on the other hand, the existence of flat regions and multiple local minimas, especially for increasing input dimension, can lead gradient methods to poor performance compared to global methods. There are also approaches that try to use features from both families (so-called hybrid methods).

An example of the objective function landscape for a one-dimensional problem is given, in Figure 7. The original model is $\mathcal{M}(x) = \left(1 + 25x^2\right)^{-1}$. The experimental design consists of 8 points as shown in Figure 7(a). The evaluation of the maximum likelihood and cross-validation criterion (to be minimized) as a function of θ is shown in Figures 7(b) and 7(c) respectively.

The optimization methods that are currently available will be briefly discussed next.

- Interior point with L-BFGS Hessian approximation

  An interior point gradient-based method is used that approximates the Hessian matrix using a Limited memory variant of the quasi-Newton Broyden-Fletcher-Goldfarb-Shanno method (Richard H. Byrd, Mary E. Hribar Y, 1999; Nocedal, 1980).

- Genetic Algorithm

  This is a well-established global optimization method (Goldberg, 1989) that has been applied in many fields for the past decades. A hybrid approach can also be used, where the final solution of the genetic algorithm is used as a starting point of the gradient method that was previously mentioned.

- Covariance matrix adaptation - evolution scheme (CMA-ES)

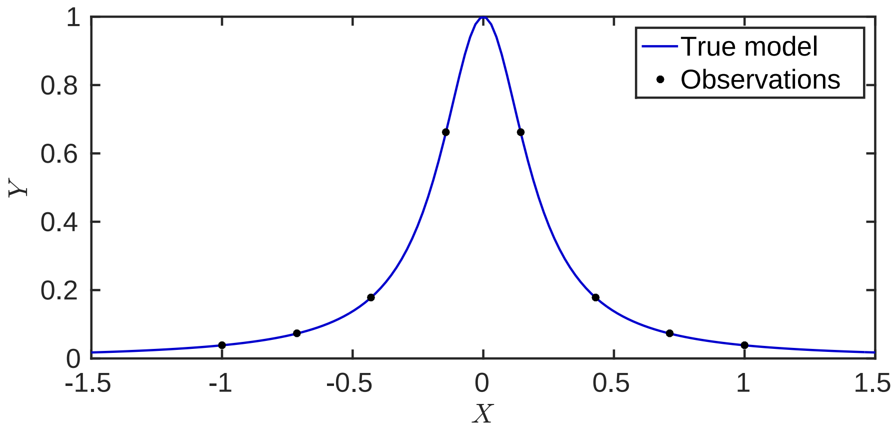
  This is a derandomized stochastic search algorithm introduced by Hansen and Ostermeier (2001). It proceeds by adapting the covariance matrix of a normal distribution such that directions that have improved the objective function in the recent past iterations are more likely to be sampled again. A hybrid approach can also be used, where the final solution of the genetic algorithm is used as a starting point of the gradient method that was previously mentioned.

1.6 *A posteriori* error estimation

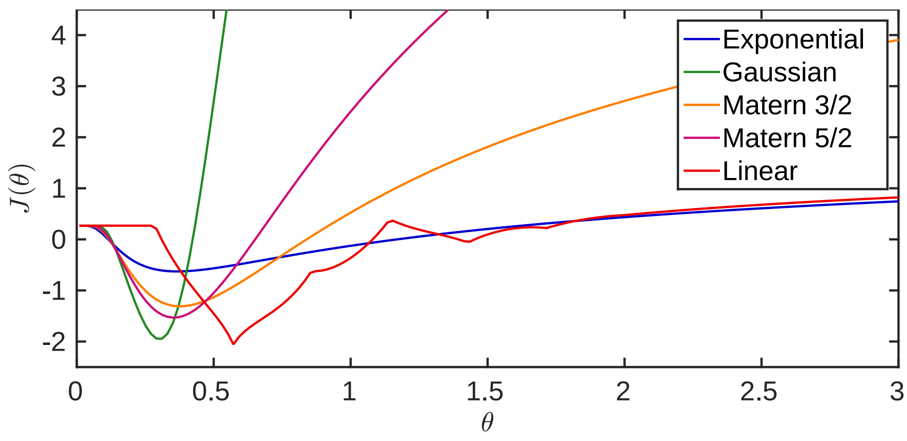
After the Kriging metamodel is set up, is set up, its predictive accuracy on new data can be assessed by using the so-called *validation error*. It is calculated as the relative generalization error on an independent set of inputs and outputs $[\boldsymbol{\mathcal{X}}_{Val},\ \mathcal{Y}_{Val} = \mathcal{M}(\boldsymbol{\mathcal{X}}_{Val})]$:

$$\epsilon_{Val} = \frac{N-1}{N}\left[\frac{\sum\limits_{i=1}^{N}\left(\mathcal{M}(\boldsymbol{x}_{Val}^{(i)}) - \mathcal{M}^{K}(\boldsymbol{x}_{Val}^{(i)})\right)^2}{\sum\limits_{i=1}^{N}\left(\mathcal{M}(\boldsymbol{x}_{Val}^{(i)}) - \hat{\mu}_{Y_{Val}}\right)^2}\right] \tag{1.32}$$

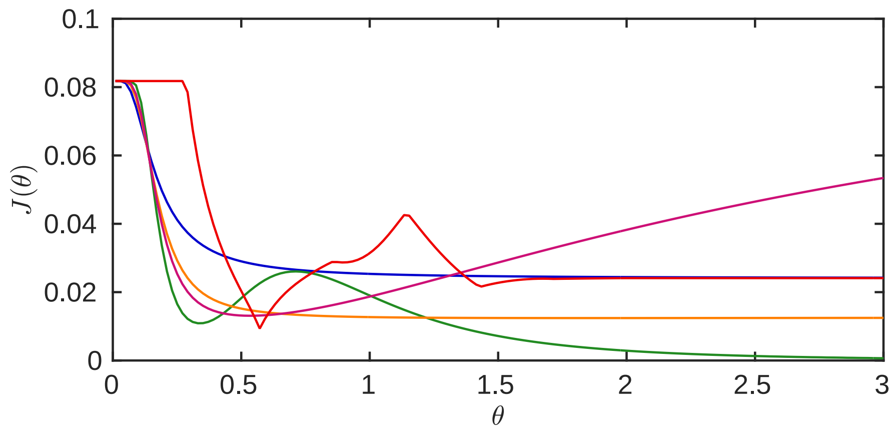
where $\hat{\mu}_{Y_{Val}} = \frac{1}{N}\sum\limits_{i=1}^{N}\mathcal{M}(\boldsymbol{x}_{Val}^{(i)})$ is the sample mean of the validation set response. This error measure is useful to compare the performance of different surrogate models when evaluated on the same validation set.



(a) The Experimental Design, drawn from the Runge function.



(b) Maximum likelihood objective function landscape.



(c) Cross-validation objective function landscape.

Figure 7: A one-dimensional example of objective function landscapes with respect to the scale parameter θ using various correlation families.

Chapter 2

Usage

In this section a reference problem will be set up to showcase how each of the Kriging options described in Section 1.1 can be used in UQLab.

2.1 Reference problem

In the context of this manual Kriging is used for building a surrogate of some model given a set of observations and model responses. To this end, the following one-dimensional function is used as the true model:

$$\mathcal{M}(x) = x \, \sin(x) \; , \; x \sim \mathcal{U}\,(0, 15) \tag{2.1}$$

2.2 Problem set-up

The Kriging module creates a MODEL object that can be used exactly as any other. The basic options common to any Kriging metamodel read:

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

Recalling (and slightly expanding) Eq. (1.1), a Kriging metamodel reads:

$$\mathcal{M}^K(\mathbf{x}) = \sum_{j=1}^{p} \beta_j(\boldsymbol{\theta}) f_j(\boldsymbol{x}) + \sigma^2 Z\left(\boldsymbol{x}\,;\,\boldsymbol{R}(\boldsymbol{\theta})\right) \tag{2.2}$$

where $\boldsymbol{\theta}$ is obtained by solving an optimization problem:

$$\boldsymbol{\theta} = \arg \min_{\boldsymbol{\theta} \in \mathcal{D}_\theta} \; J(\boldsymbol{\theta}) \tag{2.3}$$

In practice the objective function $J(\boldsymbol{\theta})$ is based on the choice of maximum likelihood or cross-validation estimation. Therefore, the main ingredients that need to be set up in order to obtain a Kriging metamodel are the following:

- An experimental design, $\mathcal{X}$, and the corresponding model responses, $\boldsymbol{y}$. If they are not available they can be generated by defining the 'true' model and a probabilistic input vector. Note that this steps belongs to the general context of metamodelling, *i.e.*it is not specific to Kriging (see also UQLAB User Manual – Polynomial chaos expansions).

- A trend specification, *i.e.*selection of the type of basis functions $f_j(\boldsymbol{x})$, $j = 1, \ldots, p$.
- An appropriate correlation function $R(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta})$.
- A hyperparameter estimation method, *i.e.*the type of $J(\boldsymbol{\theta})$ in Eq. (2.3) and the way of estimating the Gaussian process variance $\sigma^2(\boldsymbol{\theta})$.
- A method to solve the optimization problem in Eq. (2.3).

Note that the minimal amount of information that need to be supplied by the user is the experimental design. The rest of the ingredients can either be selected and tuned by the user or just left to their default values.

2.3 Calculation of the Kriging metamodel

The minimal set of options needed in order to obtain a Kriging metamodel were discussed in Section 2.2. Below a minimal configuration example is given.

```
% start the UQLab framework
uqlab;
% Create X,Y
X = transpose(0 : 2 : 14);
Y = X.*sin(X);

% Create the Kriging metamodel
Metaopts.Type = 'Metamodel';
Metaopts.MetaType = 'Kriging';
Metaopts.ExpDesign.Sampling = 'user';
Metaopts.ExpDesign.X = X;
Metaopts.ExpDesign.Y = Y;
myKriging = uq_createModel(Metaopts);
```

Once the metamodel is created, a report of the Kriging results 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
```

```
    Corr. Type:        ellipsoidal(anisotropic)
    Corr. family:      matern-3_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 regarding the trend, correlation function, estimation and optimization method have been assigned. A visual representation of the metamodel can be obtained by:

```
uq_display(myKriging);
```

Note that `uq_display` for Kriging MODEL objects can only be used for one- and two- dimensional inputs. The figure produced by `uq_display` is shown in Figure 8.

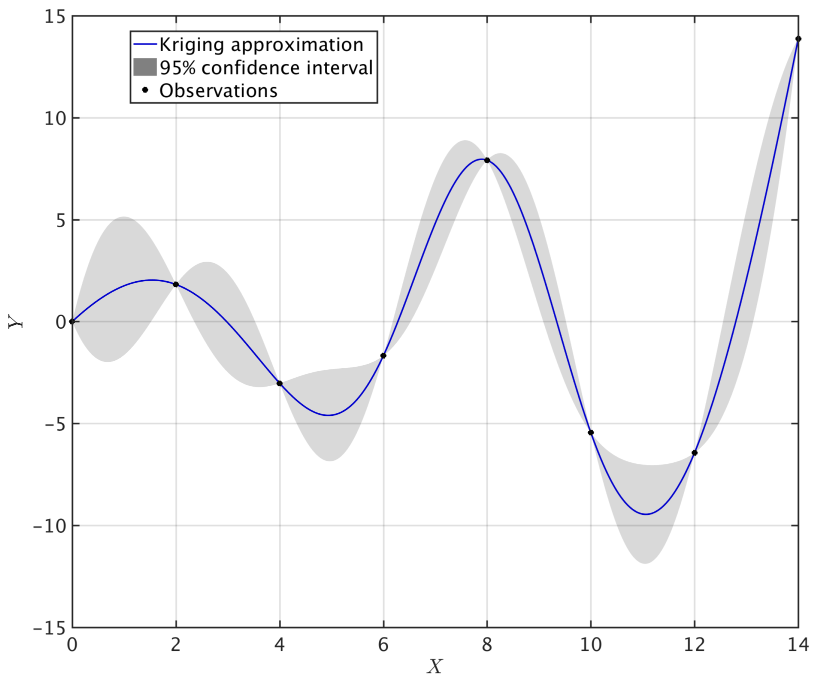


Figure 8: The output of `uq_display` of a Kriging MODEL object having a one-dimensional input.

2.3.1 Accessing the results

The Kriging MODEL object `myKriging` that was created in Section 2.3 uses all the default configuration options (the were listed in the output of `uq_print(myKriging)`). We can directly access this object in order to obtain various results and information of the metamodel as will be discussed next.

Kriging parameters

The values of the basic parameters of the Kriging metamodel, namely $\boldsymbol{\theta}, \sigma^2, \boldsymbol{\beta}$ are contained in the structure `myKriging.Kriging`:

```
myKriging.Kriging
ans =
       beta: 69.8392
       sigmaSQ: 2.5660e+05
       theta: 9.9986
```

Model evaluations

The experimental design and the corresponding model evaluations are stored in the `myKriging.ExpDesign` structure:

```
myKriging.ExpDesign
ans =
  Sampling: 'user'
  X: [8x1 double]
  Y: [8x1 double]
  NSamples: 8
  U: [8x1 double]
  time: 0.0131
```

The `ExpDesign.Sampling` field contains information about the generation of the sample of model evaluations. The `ExpDesign.NSamples` field contains the total number of full-model evaluations that were run during the calculation.

The remaining fields contain, respectively:

- `ExpDesign.X`: the experimental design $\mathcal{X}$.
- `ExpDesign.Y`: the full model evaluations, $\boldsymbol{y} = \mathcal{M}(\mathcal{X})$.
- `ExpDesign.U`: the scaled experimental design. Refer to Section 2.9 for more information.
- `ExpDesign.time`: the time elapsed (in seconds) in order to obtain the experimental design and the corresponding model evaluations.

***A posteriori* error estimates**

The Leave-One-Error (LOO) of the metamodel is stored in `myKriging.Error`:

```
myKriging.Error
ans =
    LOO: 0.4642
```

It is calculated as follows:

$$E_{LOO} = \frac{1}{N} \sum_{i=1}^{N} \left(\mathcal{M}(\boldsymbol{x}_i) - \mu_{\hat{Y},(-i)}(\boldsymbol{x}_i) \right)^2 / \mathrm{Var}\left[Y\right] \tag{2.4}$$

where $\mu_{\hat{Y},(-i)}(\boldsymbol{x}_i)$ denotes the Kriging metamodel that is obtained by using all the points of the experimental design $\mathcal{X}$ except $\boldsymbol{x}_i$.

2.4 Set-up of the Kriging metamodel

In the following subsections the various configuration options of each of the ingredients of a Kriging metamodel are presented. These ingredients are summarized below:

- `Metaopts.ExpDesign` contains options regarding the generation or specification of the Experimental Design. The list of all available options can be found in Table 5 and the way to use each option is discussed in Section 2.4.1.
- `Metaopts.Trend` contains options regarding the term $\sum_{j=1}^{p} \beta_j(\boldsymbol{\theta}) f_j(\mathbf{x})$ in Eq. (2.2), see Section 1.1 for a theoretical introduction. The list of all available options can be found in Table 6 and the way to use each option is discussed in Section 2.4.2.
- `Metaopts.Corr` contains options regarding the correlation function that is used in order to obtain $\boldsymbol{R}$ in Eq. (2.2), see Section 1.3 for a theoretical introduction. The list of all available options can be found in Table 8 and the way to use each option is discussed in Section 2.4.3.
- `Metaopts.EstimMethod` refers to the method that is used for estimating the hyperparameters $\boldsymbol{\theta}$ (maximum-likelihood or cross-validation). The choice of estimation method corresponds to different ways of calculating $\sigma^2(\boldsymbol{\theta})$ in Eq. (2.2) as well as $J(\boldsymbol{\theta})$ in Eq. (2.3). See Section 1.4 for a theoretical introduction of each estimation method.
- `Metaopts.Optim` contains options related to the method to be used for solving the optimization problem in Eq. (2.3), see Section 1.5 for a brief description of the available optimization methods. The list of all available options can be found in Table 10 and the way to use each option is discussed in Section 2.4.5.

2.4.1 Generation/Specification of the experimental design

2.4.1.1 Using already existing data

Depending on where the experimental design data is stored the following alternatives are offered:

- If data is stored in the MATLAB workspace, *e.g.*in the variables `X`, `Y`:

```
Metaopts.ExpDesign.Sampling = 'user';
Metaopts.ExpDesign.X = X;
Metaopts.ExpDesign.Y = Y;
```

- If data is stored in a .mat file, *e.g.*`mydata.mat`:

```
Metaopts.ExpDesign.Sampling = 'data' ;
Metaopts.ExpDesign.DataFile = 'mydata.mat' ;
```

Note: The experimental design that is contained in `mydata.mat` must be saved with variable names `X` and `Y` respectively.

The input and output dimension of the problem M, N_{out} are automatically retrieved by the number of columns of `X` and `Y` respectively.

2.4.1.2 Using INPUT and MODEL objects

First we have to create the probabilistic INPUT vector. For the reference problem in Eq. (2.1) this reads:

```
IOpts.Marginals.Type = 'Uniform';
IOpts.Marginals.Parameters = [0 15];
myInput = uq_createInput(IOpts);
```

The input and output dimension of the problem are automatically retrieved by the configuration of the INPUT object `myInput` and MODEL object `myModel`. For more information about the configuration options available of an INPUT and a MODEL object, please refer to the UQLAB User Manual – the INPUT module and the UQLAB User Manual – the MODEL module respectively.

Then we create a MODEL object as follows:

```
MOpts.mString = 'X.*sin(X)' ;
myModel = uq_createModel(MOpts);
```

Now we include the INPUT and MODEL objects to the Kriging metamodel configuration:

```
Metaopts.Input = myInput;
Metaopts.FullModel = myModel;
```

Finally, we need to define the size of the experimental design, *e.g.*for $N = 8$ samples:

```
Metaopts.ExpDesign.NSamples = 8;
```

Note that, by default Monte Carlo sampling is used in order to obtain $\mathcal{X}$. However other sampling strategies can be selected through the option `Metaopts.ExpDesign.Sampling`, see Table 5 in Section 3.1.1 for a list of the available sampling strategies.

2.4.2 Trend

2.4.2.1 Standard options

Some typical configuration options are the following:

- When considering a constant (*ordinary* Kriging), linear or quadratic trend the field `Metaopts.Trend.Type` must contain the appropriate string value, that is, `'ordinary'`, `'linear'` and `'quadratic'` respectively.
- For a polynomial trend of arbitrary degree we need to set

  ```
  Metaopts.Trend.Type = 'polynomial';
  Metaopts.Trend.Degree = q;
  ```

  where `q` is an integer equal to the polynomial degree.
- For a constant trend with fixed value (*simple* Kriging) we need to set the following:

  ```
  Metaopts.Trend.Type = 'simple';
  Metaopts.Trend.CustomF = V;
  ```

  where `V` is a real number.

A summary of the available trend options in UQLab along with the corresponding formula and the additionally required options is given in Table 2.

Table 2: Trend options combinations

`Trend.Type`	Formula	`Trend.Degree`	`Trend.CustomF`
`'simple'`	$f(\boldsymbol{x})$ (no regression)	Not required	Required
`'ordinary'`	β_0	Not required	Not required
`'linear'`	$\beta_0 + \Sigma_{i=1}^{M} \beta_i x_i$	Not required	Not required
`'quadratic'`	$\beta_0 + \Sigma_{i=1}^{M} \beta_i x_i + \Sigma_{i=1}^{M} \Sigma_{j=1}^{M} \beta_{ij} x_i x_j$	Not required	Not required
`'polynomial'`	see Eq. (1.12)	Required	Not required
`'custom'`	$\Sigma_{i=1}^{P} \beta_i f_i(\boldsymbol{x})$	Not required	Required

In Figure 9 the mean and variance of various Kriging predictors having different trend configurations are plotted using the example script `uq_Example_Kriging_03_TrendTypes`.

Note: The field `Metaopts.Trend.Type` determines the trend type of a Kriging meta model. When no trend type has been defined by the user, the default `Metaopts.Trend.Type = 'ordinary'` is used (Ordinary Kriging).

2.4.2.2 Advanced options

- For a custom arbitrary trend, first we need to define it

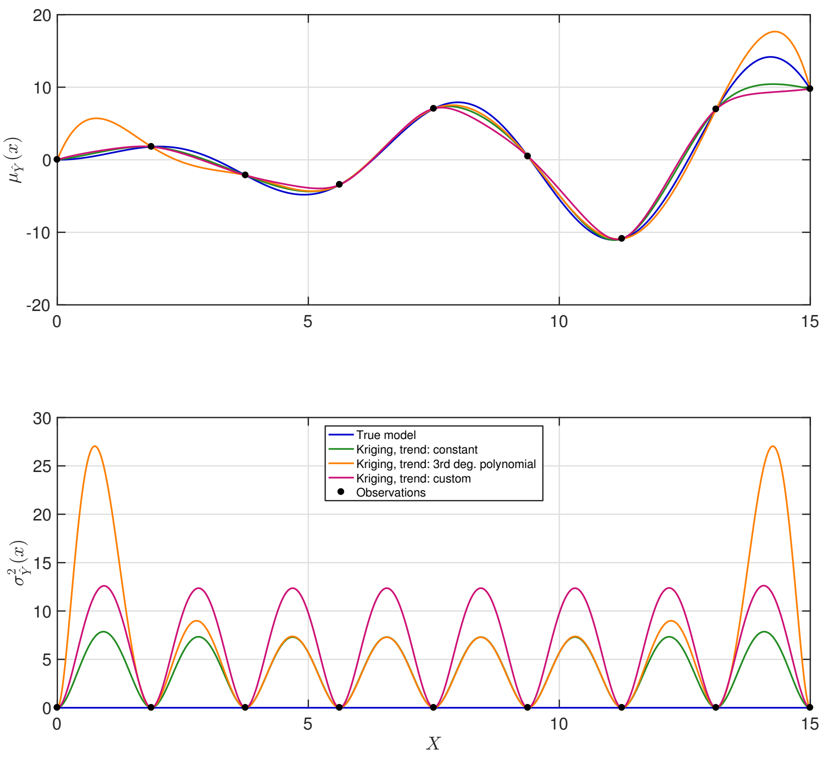


Figure 9: The output of the example script`uq_Example_Kriging_03_TrendTypes`. The mean and variance of various Kriging predictors is plotted, using different trend configurations.

```
Metaopts.Trend.Type = 'custom';
```

Then we can either define the trend as a string or a function handle. If for example the trend we want is $\sin(x)$ then

```
Metaopts.Trend.CustomF = 'sin';
```

or

```
Metaopts.Trend.CustomF = @sin;
```

The function can be any built-in MATLAB, UQLAB or user defined function (as long as it is contained in the MATLAB path). It is also possible to assign to `Metaopts.Trend.CustomF` a cell array of strings or function handles of arbitrary length.

In general the dimension of the information matrix $\boldsymbol{F}$ is $N \times P$. Depending on the selection of the trend type (`Metaopts.Trend.Type`) the value of P varies:

- When `Metaopts.Trend.Type` is one of `'ordinary'`, `'linear'`, `'quadratic'`, `'polynomial'` then P equals to $1, M+1, \frac{M(M+1)}{2}$ respectively.

- When `Metaopts.Trend.Type='polynomial'` then depending on the polynomial degree q, defined in `Metaopts.Trend.Degree`, then P equals to $\binom{M+q}{q}$.
- When `Metaopts.Trend.Type` is `'simple'` then P equals to 1.
- When `Metaopts.Trend.Type` is `'custom'` :
  - If `Metaopts.Trend.CustomF` is a string or double then P equals to 1.
  - If `Metaopts.Trend.CustomF` is a cell array of strings or function handles then P equals the length of the cell array. In other words, each element of the array corresponds to a single column of $\boldsymbol{F}$.

2.4.3 Correlation function

2.4.3.1 Standard options

The main ingredients for specifying the correlation function are:

- **Type:** The type of the autocorrelation function as described in Section 1.3.1. For example, in order to use a separable correlation function the following option is needed:

  ```
  Metaopts.Corr.Type = 'separable';
  ```

  If not specified otherwise by the user the correlation type is assumed to be `'ellipsoidal'`.

- **Family:** The correlation function family is a 1-D function $R(\cdot)$. Its inputs depend on the type of correlation function.
  - For *ellipsoidal* correlation functions the correlation family is a function of the form $R(h)$ where $h > 0$ corresponds to the standardized Euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{x}'$ (see Eq. (1.13)). Each coordinate difference between the rows of $\boldsymbol{x}$ and $\boldsymbol{x}'$ is standardized by a positive scalar θ.
  - For *separable* correlation functions the correlation family is a function of the form $R(x, x'; \theta)$ where θ is a positive scalar.

  For using a built-in correlation family its name needs to be set, see Table 8 in Reference List (Section 3.1) for the name of each built-in correlation family and Section 1.3.4 for a brief description of each family. For example, in order to use the exponential family the following option is set:

  ```
  Metaopts.Corr.Family = 'exponential';
  ```

Note: The default correlation family `'matern-3_2'` is used if not specified otherwise by the user.

- **Isotropic:** This is a Boolean flag (with `true` or `false` value) that specifies whether the correlation function is isotropic or not. By default the correlation function is considered anisotropic, unless the following option is set:

```
Metaopts.Corr.Isotropic = true;
```

In Figure 10 the mean and variance of various Kriging predictors having different correlation families are plotted using the example script `uq_Example_Kriging_01_1D`.

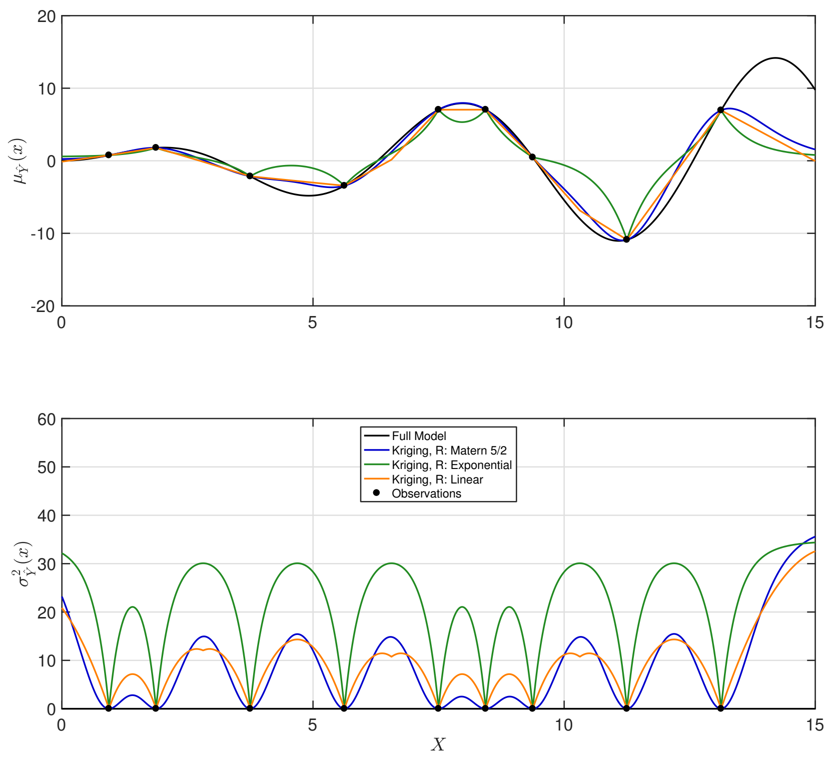


Figure 10: The output of the example script `uq_Example_Kriging_01_1D`. The mean and variance of various Kriging predictors is plotted, using different correlation families.

2.4.3.2 Advanced options

- **User-defined correlation family**: Apart from the already existing correlation families user-defined ones can be used. Depending on the type of correlation function that has been selected (separable or ellipsoidal) the user-defined correlation family is expected to accept different inputs.
  - **Separable case:** A user-defined correlation family is expected to be of the form $R(x, x'; \theta)$ where x, x' are scalars and θ a positive scalar. For example if the following correlation family needs to be used:

$$R(x, x'; \theta) = \exp\left[-\left(\frac{x - x'}{\theta}\right)^{1.5}\right]$$

the following options should be provided:

```
Metaopts.Corr.Type = 'separable';
Metaopts.Corr.Family = @(x1,x2,th) exp(-((x1-x2)/th).^1.5);
```

Notice that for the separable case the user-defined correlation function can also be *non-stationary*.

> **Note:** The user-defined correlation familiy is expected to be *vectorized*, *i.e.*it should be able to handle inputs `x1` and `x2` that are vectors either with the same or different length.

– **Ellipsoidal case:** A user-defined correlation family is expected to be of the form $R(h)$ where h is a positive scalar. The value of h is calculated by UQLab as in Eq. (1.13) and then sent to the user-defined function. For example if the following correlation family needs to be used:

$$R(h) = \exp\left(-h^{1.5}\right)$$

the following options should be provided:

```
Metaopts.Corr.Type = 'ellipsoidal';
Metaopts.Corr.Family = @(h) exp(-h.^1.5);
```

> **Note:** The user-defined correlation family is expected to be *vectorized*, *i.e.*it should be able to handle inputs `h` that are vectors of arbitrary length.

- **Nugget**: A *nugget* is a small numerical value added to the diagonal of the $\boldsymbol{R}$ matrix to improve the stability of its numerical inversion. To add a constant nugget $\nu = 10^{-4}$ one can write:

```
Metaopts.Corr.Nugget = 1e-4;
```

If the nugget is specified as a vector $\boldsymbol{\nu}$ of length N, each of its elements is added to the corresponding diagonal element of $\boldsymbol{R}$: $R_{ii} = 1 + \nu_i, \;\; \{i = 1, \cdots, N\}$.

> **Note:** By default, the nugget value in UQLab is set to $\nu = 10^{-10}$ (see Section 1.3.3).

- **User-specified routine to calculate the correlation matrix (R)**: All the aforementioned options regarding the correlation function are handled by the function `uq_Kriging_eval_R`. If the user wants to use a fully custom correlation function in order to obtain the correlation matrix $\boldsymbol{R}$ the following rules need to be followed.

- The function should return the correlation matrix $\boldsymbol{R}$ and accept x_1, x_2, θ and a structure of options like the following:

  ```
  function R = my_eval_R(x1,x2,theta, options)
  ...
  ```

- Inputs `x1`, `x2` are matrices with arbitrary number of rows and M number of columns (where M is the input dimension).
- When the custom correlation function `my_eval_R` is called, *e.g.*during the creation or usage of a Kriging metamodel, the structure `options` will contain all options that were set inside `Metaopts.Corr`.
- The input `theta` of the function `my_eval_R` corresponds to the hyperparameters $\boldsymbol{\theta}$ and it is expected to be a vector of arbitrary length. There is no limitation regarding its length but its dimension should be reflected in the choice of either initial value and/or optimization bounds(see Section 2.4.5 for more information about the optimization options).

It can be used for calculating the Kriging metamodel as well as within the Kriging predictor, as long as the following option has been set:

```
Metaopts.Corr.Handle = @my_eval_R;
```

2.4.4 Hyperparameters estimation methods

The hyperparameters estimation method affects the type of the objective function that is minimized in Eq. (2.3) as well as the way the (constant) Gaussian process variance σ^2 is calculated (for more information see Section 1.4).
Currently, the maximum likelihood and cross-validation methods are available for estimating the Kriging parameters and it suffices to select either `'ML'` or `'CV'` as the value of the field `Metaopts.EstimMethod`.

Note: If not specified otherwise by the user, the leave-one-out cross-validation method is used as the default.

A comparison of the resulting Kriging predictors using different estimation (and optimization) methods can be found in the example script `uq_Example_Kriging_02_VariousMethods` (see Figure 11).

2.4.4.1 Advanced Options

When the cross-validation method is used it is by default the leave-one-out approach. It can be set to be leave-K-out by setting:

```
Metaopts.CV.LeaveKOut = K;
```

where K is an integer that should be smaller than the size of the experimental design.

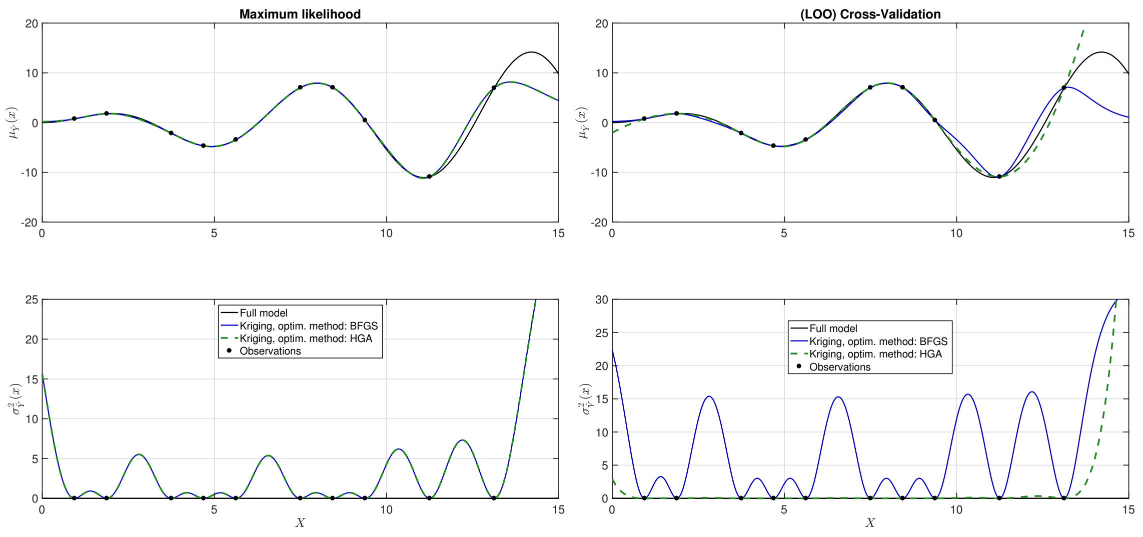


Figure 11: The output of the example script `uq_Example_Kriging_02_VariousMethods`. The mean and variance of various Kriging predictors is plotted, using different optimization methods. On the left hand side the maximum likelihood estimation method is used and on the right hand side the (LOO) cross validation method.

2.4.5 Hyperparameter optimization

Various configuration options are available regarding the method to solve the optimization problem in Eq. (2.3). The available optimization methods can be divided in two categories:

- **Gradient-based methods**

  Currently the BFGS method is available. In order to use this method the following syntax is used:

  ```
  Metaopts.Optim.Method = 'BFGS';
  ```

  As in any gradient-based method, an initial value of the hyperparameters (here denoted as $\boldsymbol{\theta}_0$) is required. Note that the hyperparameters refer to the space $\mathcal{U}$ as described in Section 2.9. By default, $\boldsymbol{\theta_0} = 1$ in each dimension. However, a different initial value can be selected, *e.g.*$\theta_0 = 0.5$ (in each dimension):

  ```
  Metaopts.Optim.InitialValue = 0.5;
  ```

  Different initial value per dimension can be selected by specifying an $M \times 1$ vector instead of a scalar.

- **Global methods**

  Currently the Genetic Algorithm (GA), the covariance matrix adaptation-evolution scheme (CMA-ES) and their hybrid counterparts (respectively HGA and HCMAES) are available in UQLab. To use one of these methods for solving the optimization of the hyperparameters, *e.g.* the Genetic Algorithm, one sets:

```
Metaopts.Optim.Method = 'GA';
```

As in any global method the bounds of the optimization variable(s) needs to be set. Note that the hyperparameters refer to the space $\mathcal{U}$ as described in Section 2.9. By default, the bounds (lower, upper) $[0.001, 10]^T$ are used. However, different bounds, *e.g.* $[0.01, 1]^T$ can be used as follows:

```
Metaopts.Optim.Bounds = [0.01 ; 1];
```

Additional options which are specific to each method may exist. For example, when using the Genetic Algorithm method one might need to set a different value of stall generations, *e.g.* 20. This can by accomplished by setting:

```
 Metaopts.Optim.HGA.nStall = 20;
```

Moreover, regardless of the method, we can manually specify the following options:

- The number of iterations or generations (depending on the optimization method):

```
    Metaopts.Optim.MaxIter = 100;
```

- The convergence tolerance:

```
    Metaopts.Optim.Tol = 1e-5;
```

- The verbosity of the optimization process, *e.g.*showing only the final result:

```
    Metaopts.Optim.Display = 'final';
```

For a list of all the available options for each method, please refer to Table 10 in Section 3.1.5.

Note: When no method has been defined by the user, the default `Metaopts.Optim.Method = 'HGA'` (Hybrid Genetic Algorithm) is used.

A comparison of the resulting Kriging predictors using different optimization (and estimation) methods can be found in the example script `uq_Example_Kriging_02_VariousMethods` (see Figure 11).

When no optimization of the hyperparameters is required because a prescribed value `theta_user` is to be used, the following syntax shall be used:

```
Metaopts.Optim.Method = 'none';
Metaopts.Optim.InitialValue = theta_user;
```

2.5 Use of a validation set

If a validation set is provided (see Table 17 in Section 3.1.7.), UQLAB automatically computes the validation error given in Eq. (1.32). To provide a validation set, the following command shall be used:

```
MetaOpts.ValidationSet.X = XVal;
MetaOpts.ValidationSet.Y = YVal;
```

The value of the validation error is stored in `myKriging.Error.Val` (see Table 21) and will also be displayed when typing `uq_print(myKriging)`.

2.6 Kriging metamodels of vector-valued models

All the examples presented so far in this chapter dealt with scalar-valued models. In case the model (or the experimental design, if manually specified) produces multi-component outputs, UQLab performs an independent Kriging metamodel for each output component on the shared experimental design. No additional configuration is needed to enable this behaviour. A Kriging example with multi-component outputs can be found in the UQLab example script `uq_Example_Kriging_04_MultipleOutputs`.

2.6.1 Accessing the results

Running a Kriging calculation on a multi-component output model will result in a multi-component output structure. As an example, a model with 9 outputs will produce the following output structure:

```
myKriging.Kriging
ans =
1x9 struct array with fields:
beta
sigmaSQ
theta
```

Each element of the `Kriging` structure is functionally identical to its scalar counterpart in Section 2.3.1. Similarly, the `myKriging.Error` structure becomes a multi-element structure:

```
myKriging.Error
ans =
1x9 struct array with fields:
LOO
```

2.7 Using a Kriging predictor as a model

Regardless of the configuration options that were used in order to derive the Kriging metamodel Eqs. (1.5)-(1.6) can be used to predict new points. Indeed, after a Kriging MODEL object is created in UQLab, it can be used just like an ordinary model (for details, see the UQLab User Manual – the MODEL module).

Consider the example in Section 2.3. After obtaining a Kriging metamodel using any set of configuration options described in Section 2.4, one can evaluate the mean of the Kriging predictor on point $x = 1$ as follows:

```
x = 1;
% Evaluate the mean of the Kriging predictor on point x
YmuKG = uq_evalModel(x)
YmuKG =
0.8782
```

The variance of the predictor can be also evaluated by using a two-component output:

```
% Evaluate the mean and variance of the Kriging predictor on point x
[YmuKG, YvarKG] = uq_evalModel(x)
YmuKG =
0.8782
YvarKG =
53.5691
```

As most functions within UQLAB, model evaluations are vecorized, *i.e.*evaluating multiple points at a time is much faster than repeatedly evaluating one point at a time:

```
x = transpose(0:0.1:15);
% Evaluate the mean and variance of the Kriging predictor on points x
[YmuKG, YvarKG] = uq_evalModel(x);
```

2.8 Manually specifying a Kriging predictor (*predictor-only* mode)

It is possible to use the Kriging module in UQLAB to build custom Kriging-based models to be used as predictors as in Section 2.7. This allows, *e.g.* to import a metamodel calculated with another software into the UQLAB framework, or even to create one *ad-hoc* model.
In the following, we exemplify how to create a custom Kriging metamodel that is similar to the one that was obtained in Section 2.3.

```
% Create the experimental design and calculate the response of the model
X = transpose(0 : 2 : 15);
Y = X.*sin(X);

% Create a custom Kriging object
Metaopts.Type =  'Metamodel';
Metaopts.MetaType = 'Kriging';
Metaopts.ExpDesign.Sampling = 'User';
Metaopts.ExpDesign.X = X;
Metaopts.ExpDesign.Y = Y;
% Select ordinary Kriging
Metaopts.Kriging.Trend.Type = 'ordinary';
% Set the values of beta, sigma^2 and theta
Metaopts.Kriging.beta = 69.84 ;
Metaopts.Kriging.sigmaSQ = 2.566e5 ;
Metaopts.Kriging.theta = 9.999;
% Select ellipsoidal, Matern-3/2 correlation function
Metaopts.Kriging.Corr.Type = 'ellipsoidal';
Metaopts.Kriging.Corr.Family = 'matern-3_2';
% Create the metamodel
myCustomKriging = uq_createModel(Metaopts);
```

```
% Evaluate the metamodel on some new points
Xnew = transpose(1 : 2 : 13);
Ynew = uq_evalModel(Xnew);
```

In case the metamodel has more than one output, it is sufficient to specify the same information for each output coordinate by adding an index `i` to the `Metaopts.Kriging(i)` structure.

Note: By default the nugget value is set to $\nu = 10^{-10}$ when creating a custom Kriging metamodel (see Section 1.3.3).

2.9 Performing Kriging on an Auxiliary space (Scaling)

The Kriging module offers various options for scaling the experimental design before computing the metamodel. Instead of $\mathcal{X}$, the module always uses the *scaled* experimental design $\mathcal{U}$. Depending on the value of the option `Metaopts.Scaling` and whether a probabilistic input model has been defined (in `Metaopts.Input`), the transformation $\mathcal{X} \mapsto \mathcal{U}$ varies. All the possible cases are summarised in Table 3.

Table 3: Available experimental design transformations

`Input` / `Scaling`	No INPUT defined	INPUT defined
1	$\mathcal{U} = (\mathcal{X} - \mu_{\mathcal{X}})/\sigma_{\mathcal{X}}$	$\mathcal{U} = (\mathcal{X} - \mu_{\boldsymbol{X}})/\sigma_{\boldsymbol{X}}$
0	$\mathcal{U} = \mathcal{X}$	$\mathcal{U} = \mathcal{X}$
INPUT object	-	$\mathcal{U} = \tau(\mathcal{X})$

By setting `Metaopts.Scaling = 1` the experimental design is scaled so that it has zero mean and unit variance. When no input distribution has been specified $(\mu_{\mathcal{X}}, \sigma_{\mathcal{X}})$ are empirically estimated from the available data specified in `Metaopts.ExpDesign.X`. When an input distribution has been specified via an INPUT object, say `myInput`, as follows:

```
Metaopts.Input = myInput;
```

then $(\mu_{\boldsymbol{X}}, \sigma_{\boldsymbol{X}})$ are estimated from the moments of the marginal distribution of each component in $\boldsymbol{X}$.

When, an additional INPUT object, say `myScaledInput` is set as follows:

```
Metaopts.Scaling = myScaledInput;
```

then $\mathcal{U} = \tau(\mathcal{X})$ where τ denotes the generalised *isoprobabilistic transformation* between the two probability spaces (for more information refer to the UQLAB User Manual – the INPUT module).

Note: By default `Metaopts.Scaling = 1`.

2.10 Drawing samples from a Gaussian process posterior

In this Section the introductory example in Section 2.3 is considered. Instead of retrieving the mean and confidence bounds of the Gaussian process posterior (*w.r.t.* the experimental design) one may also retrieve an arbitrary number of samples (*a.k.a.* realisations) of it. This can be achieved by exploiting the Gaussian property of any Kriging surrogate. Each sample of the Gaussian process $\boldsymbol{Y}$, discretized over a set of points $\mathcal{X}_s = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_s}\}$ follows a multivariate Gaussian distribution, *i.e.* $\boldsymbol{Y} \sim \mathcal{N}_{N_s}(\boldsymbol{\mu}_{\hat{Y}}, \boldsymbol{C}_{\hat{Y}})$ where $\boldsymbol{\mu}_{\hat{Y}}$ corresponds to the posterior mean and $\boldsymbol{C}_{\hat{Y}}$ to posterior covariance matrix of the Kriging predictor evaluated on $\mathcal{X}_s$.

Given the `myKriging` object that was calculated in Section 2.3, the posterior mean and covariance matrix of the Kriging predictor are obtained as follows:

```
Ns = 100
Xs = linspace(0, 15, Ns)';
[mu, sigma2, C] = uq_evalModel(myKriging, Xs);
```

Then, one sample of the posterior Gaussian process can be obtained as follows:

```
Ys = mu + chol(C)*mvnrnd(zeros(Ns,1), eye(Ns));
```

A set of 15 such samples is plotted in Figure 12 together with the mean of the Kriging predictor.

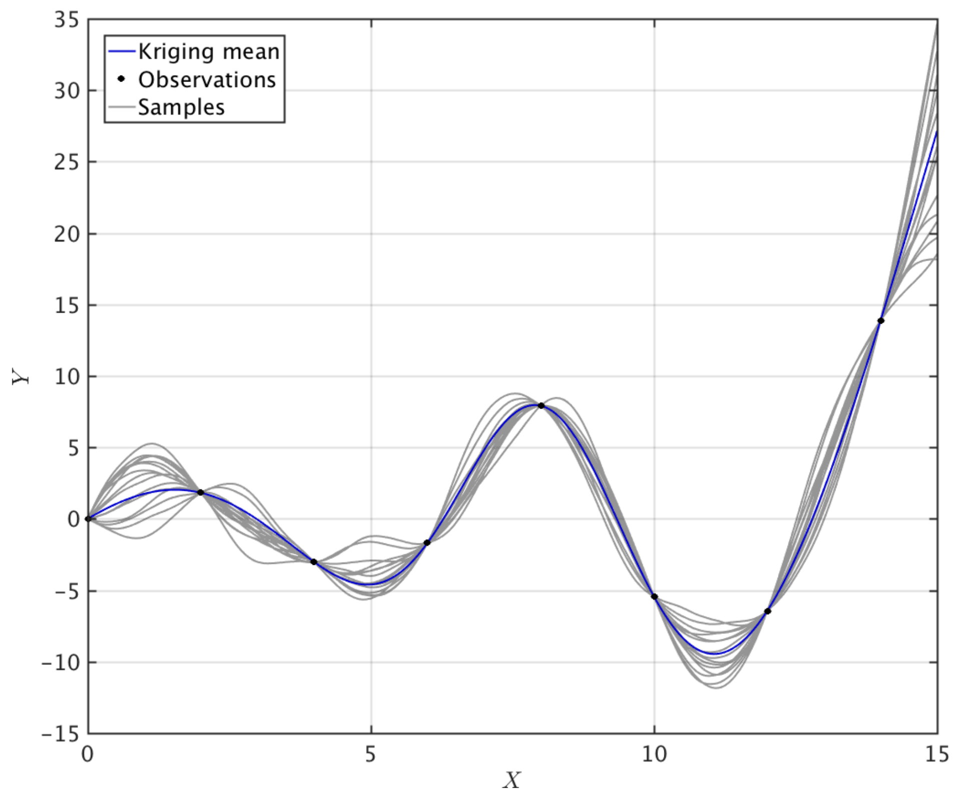


Figure 12: The mean of the Kriging predictor together with 15 samples of the posterior Gaussian process.

2.11 Using Kriging with constant parameters

In some analyses, one may need to assign a constant value to one or to a set of parameters. When this is the case, the Kriging metamodel is built by internally removing the constant parameters from the inputs. This process is transparent to the users as they shall still evaluate the model using the full set of parameters (including those which were set constant). UQLAB will automatically and appropriately account for the set of input parameters which were declared constant.
To set a parameter to constant, the following command can be used (See UQLAB User Manual – the INPUT module):

```
inputOpts.Marginals.Type = 'Constant' ;
inputOpts.Marginals.Parameters = value;
```

Furthermore, when the standard deviation of a parameter is set to zero, UQLAB automatically sets this parameter's marginal to the type `Constant`. For example, the following uniformly distributed variable whose upper and lower bounds are identical is automatically set to a constant with value 1:

```
inputOpts.Marginals.Type = 'Uniform' ;
inputOpts.Marginals.Parameters = [1 1];
```

Chapter 3

Reference List

How to read the reference list

Structures play an important role throughout the UQLab syntax. They offer a natural way to group configuration options and output quantities semantically. Due to the complexity of the algorithms implemented, it is not uncommon to employ nested structures to fine-tune inputs/outputs. Throughout this reference guide, we adopt a table-based description of the configuration structures.
The simplest case is given when a field of the structure is a simple value/array of values:

Table X: `Input`			
●	`.Name`	String	A description of the field is put here

which corresponds to the following syntax:

```
Input.Name = 'My Input';
```

The columns correspond to name, data type and a brief description of each field. At the beginning of each row a symbol is given to inform as to whether the corresponding field is mandatory, optional, mutually exclusive, etc. The comprehensive list of symbols is given in the following table:

●	Mandatory
☐	Optional
⊕	Mandatory, mutually exclusive (only one of the fields can be set)
⊞	Optional, mutually exclusive (one of them can be set, if at least one of the group is set, otherwise none is necessary)

When one of the fields of a structure is a nested structure, we provide a link to a table that describes the available options, as in the case of the `Options` field in the following example:

Table X: `Input`

●	.Name	String	Description
☐	.Options	Table Y	Description of the Options structure

Table Y: Input.Options			
●	.Field1	String	Description of Field1
☐	.Field2	Double	Description of Field2

In some cases an option value gives the possibility to define further options related to that value. The general syntax would be

```
Input.Option1 = 'VALUE1' ;
Input.VALUE1.Val1Opt1 = ...;
Input.VALUE1.Val1Opt2 = ...;
```

This is illustrated as follows:

Table X: Input			
●	.Option1	String	Short description
		'VALUE1'	Description of 'VALUE1'
		'VALUE2'	Description of 'VALUE2'
⊞	.VALUE1	Table Y	Options for 'VALUE1'
⊞	.VALUE2	Table Z	Options for 'VALUE2'

Table Y: Input.VALUE1			
☐	.Val1Opt1	String	Description
☐	.Val1Opt2	Double	Description

Table Z: Input.VALUE2			
☐	.Val2Opt1	String	Description
☐	.Val2Opt2	Double	Description

Note: In the sequel, double/doubles mean a real number represented in double precision (resp. a set of such real numbers).

3.1 Create a Kriging metamodel

Syntax

```
myKriging = uq_createModel(Metaopts)
```

Input

The struct variable `Metaopts` contains the configuration information for a Kriging metamodel. The detailed list of options available is reported in Table 4.

Table 4: `Metaopts`

●	`.Type`	`'Metamodel'`	Select the metamodelling tool
●	`.MetaType`	`'Kriging'`	Select Kriging
☐	`.Name`	String	Unique identifier for the metamodel
☐	`.Display`	String default: `'standard'`	Level of information displayed by the methods.
		`'quiet'`	Minimum display level, displays nothing or very few information.
		`'standard'`	Default display level, shows the most important information.
		`'verbose'`	Maximum display level, shows all the information on runtime, like updates on iterations, etc.
☐	`.ExpDesign`	Table 5	Experimental Design options (Section 2.4.1)
☐	`.Input`	String	The name of the INPUT object that describes the inputs of the metamodel
		INPUT object	The INPUT object that describes the inputs of the metamodel
☐	`.FullModel`	String	The name of the MODEL object that is used to calculate the model responses
		MODEL object	The MODEL object that is used to calculate the model responses
☐	`.Trend`	Table 6	Trend-related options (Section 2.4.2)
☐	`.Corr`	Table 8	Correlation function options (Section 2.4.3)
☐	`.EstimMethod`	String default: `'CV'`	Hyperparameters estimation method (Section 2.4.4)

		'CV'	Cross-Validation estimation
		'ML'	Maximum-Likelihood estimation
☐	.CV	Table 9	Options relevant to the Cross-Validation estimation method. It is only taken into account if the selected method is Cross-Validation
☐	.Optim	Table 10	Hyperparameter optimization-related options (Section 2.4.5)
☐	.Kriging	Table 16	Field used for specifying the parameters of a custom Kriging metamodel (Section 2.8)
☐	.KeepCache	Logical default: true	When set to 0 some cached matrices are removed after the metamodel has been created. By keeping the cached matrices the speed of the Kriging predictor may be increased for some problems.
☐	.Scaling	Logical default: true	If set to 1, the experimental design will be scaled before calculating the Kriging metamodel. For more details see Section 2.9
		INPUT object	the INPUT object that describes the auxiliary probabilistic space
☐	.ValidationSet	Table 17	Validation set components (Section 2.5)

3.1.1 Experimental design options

Table 5: Metaopts.ExpDesign

●	.Sampling	String	Sampling type
		'MC' (default)	Monte Carlo sampling
		'LHS'	Latin Hypercube sampling
		'Sobol'	Sobol sequence sampling
		'Halton'	Halton sequence sampling
		'Data'	Obtain samples directly from an external .mat file (defined in ExpDesign.DataFile)
		'User'	User defined experimental design. (defined in ExpDesign.X, ExpDesign.Y)

☐	`.DataFile`	String	A string containing the name of the mat file that contains the Experimental Design. It is only taken into account (and is necessary) when `ExpDesign.Sampling = 'Data'`.
☐	`.NSamples`	Integer	The number of samples to draw
☐	`.X`	$N \times M$ Double	User defined model input X (only applies when `ExpDesign.Sampling = 'user'`)
☐	`.Y`	$N \times O$ Double	User defined model response Y (only applies when `ExpDesign.Sampling = 'user'`)

3.1.2 Trend options

Table 6: `Metaopts.Trend`

☐	`.CustomF`	Double	constant trend in simple Kriging
		String	The function name that returns $f(\boldsymbol{x})$
		Cell containing strings	The respective names of functions $f_i(\boldsymbol{x}), i = 1, \ldots, P$ such that each function returns the i'th column of $\boldsymbol{F}$
		Cell containing handles	The respective handles of functions $f_i(\boldsymbol{x}), i = 1, \ldots, P$ such that each function returns the i'th column of $\boldsymbol{F}$
☐	`.Degree`	Double default: 3	The polynomial degree. Only applies when `Metaopts.Trend.Type = 'polynomial'`
☐	`.Type`	String default: `'ordinary'`	The type of trend
		`'simple'`	Simple Kriging: a given mean is used as a trend. No regression takes place. The mean should be given in `Metaopts.Trend.CustomF`
		`'ordinary'`	Ordinary Kriging: the mean is an unknown constant
		`'linear'`	The trend is a linear polynomial
		`'quadratic'`	The trend is a quadratic polynomial
		`'polynomial'`	The trend is a polynomial of arbitrary degree, defined in `Metaopts.Trend.Degree`
		`'custom'`	A user-defined trend is used (Section 2.4.2.2)
☐	`.TruncOptions`	Table 7	Polynomial basis truncation options

Table 7: `Metaopts.Trend.TruncOptions`			
☐	`.qNorm`	Double default: 1	Hyperbolic truncation scheme (see the UQLab User Manual – Polynomial chaos expansions)
☐	`.MaxInteraction`	Integer	Limit basis terms to up to MaxInteraction variables. See the UQLab User Manual – Polynomial chaos expansions for more information
☐	`.Custom`	Double	User-defined basis

3.1.3 Correlation-function options

Table 8: Metaopts.Corr

☐	.Type	String default: 'Ellipsoidal'	The correlation function type
		'Ellipsoidal'	Ellipsoidal correlation function
		'Separable'	Separable correlation function
☐	.Family	String or function handle default: 'Matern-5_2'	The correlation family, that is the elementary 1-D function that is used by the correlation function. For a description of each family refer to Section 1.3.4
		'Exponential'	Exponential correlation function
		'Gaussian'	Gaussian correlation function
		'Linear'	Linear correlation function
		'Matern-3_2'	Matérn-3/2 correlation function
		'Matern-5_2	Matérn-5/2 correlation function
		Function handle	The user defined correlation family function handle
☐	.Isotropic	Logical default: false	Determines whether the correlation function is isotropic or anisotropic
☐	.Nugget	Double default: 10^{-10}	• If scalar, adds this quantity to the diagonal elements of the correlation matrix $\boldsymbol{R}$ • If vector, adds each element to the corresponding diagonal element of $\boldsymbol{R}$
☐	.Handle	Function handle default: @uq_Kriging_eval_R	The handle of the function that is used to calculate the correlation matrix $\boldsymbol{R}$ (Section 2.4.3.2)

3.1.4 Estimation method options

Table 9: Metaopts.CV

☐	.LeaveKOut	Integer default: 1	• The number of elements in i, in Eq. (1.30). It can be any integer between 1 and N (the number of samples). • The default value is 1 (known as *Leave-One-Out Cross-Validation*)

3.1.5 Hyperparameter optimization options

Table 10: Metaopts.Optim

☐	.InitialValue	$M \times 1$ or 1×1 Double default: 0.5	• A vector containing the initial estimate of the correlation parameters • If $M > 1$ and a scalar is selected the same value is assumed for all dimensions
☐	.Bounds	$2 \times M$ or 2×1 Double default: $[10^{-3}, 10]^T$	• A matrix containing the bounds of admissible values of the correlation parameters, in the form `[lower_bound ; upper_bound]`. • If a single value is assigned for each bound the same value is used for all the elements of the vector $\boldsymbol{\theta}$ (in case of $M > 1$)
☐	.Display	String default: 'iter'	Determines the verbosity of the optimization process
		'none'	Nothing will be printed to the command window
		'iter'	The state of the optimization process will be printed after each iteration
		'final'	Only the final result of the optimization process will be printed
☐	.MaxIter	Integer default: 10	The maximum number of iterations or generations
☐	.Method	String default: 'HGA'	Optimization method (Section 2.4.5)
		'none'	No optimization. The value Metaopts.Optim.InitialValue is used
		'LBFGS'	Gradient based optimization (L-BFGS) using MATLAB's built-in fmincon function
		'GA'	Genetic Algorithm optimization using MATLAB's built-in ga function
		'HGA'	Hybrid Genetic Algorithm optimization using MATLAB's built-in ga and fmincon functions
⊞	.BFGS	Table 11	Options relevant to the LBFGS optimization method
⊞	.GA	Table 12	Options relevant to the GA optimization method
⊞	.HGA	Table 13	Options relevant to the HGA optimization method

⊞	.CMAES	Table 14	Options relevant to the CMA-ES optimization method
⊞	.HCMAES	Table 15	Options relevant to the HCMA-ES optimization method
☐	.Tol	Double default: 10^{-3}	Optimization Convergence tolerance

Note: The convergence tolerance defined under `Metaopts.Optim.Tol` may slightly differ depending on the optimization method. For MATLAB built-in optimization methods check the definition of the option `'TolFun'` of the respective method

Note: Some of the above optimization methods require special optimization toolboxes licenses in MATLAB. An error is issued when the user specifies an algorithm for which the corresponding toolbox is not available. The default optimizer, *i.e.* `'HGA'`, is replaced by:

– `'GA'` when the local optimization toolbox is not available;
– `'HCMAES'` when the global optimization toolbox is not available;
– `'CMAES'` when the global and local optimization toolboxes are not available.

Table 11: `Metaopts.Optim.BFGS`

☐	.nLM	Integer (≥ 1) default: 5	The limited memory size of LBFGS method

Table 12: `Metaopts.Optim.GA`

☐	.nPop	Integer default: see description	The population size of each generation. By default set to $n_{pop} = \min(20, \text{floor}(4 + 3\log M))$
☐	.nStall	Integer default: 2	The maximum number of stall generations

Table 13: `Metaopts.Optim.HGA`

☐	.nPop	Integer default: see description	The population size of each generation. By default set to $n_{pop} = \min(20, \text{floor}(4 + 3\log M))$
☐	.nStall	Integer default: 2	The maximum number of stall generations
☐	.nLM	Integer (≥ 1) default: 5	The limited memory size of LBFGS method that is executed starting from the optimal value that was obtained after the Genetic Algorithm optimization

Table 14: Metaopts.Optim.CMAES

☐	.nPop	Integer default: see description	The population size of each generation. By default set to $n_{pop} = \min(20, \text{floor}(4 + 3\log M))$
☐	.nStall	Integer default: 5	The maximum number of stall generations

Table 15: Metaopts.Optim.HCMAES

☐	.nPop	Integer default: see description	The population size of each generation. By default set to $n_{pop} = \min(20, \text{floor}(4 + 3\log M))$
☐	.nStall	Integer default: 5	The maximum number of stall generations
☐	.nLM	Integer (≥ 1) default: 5	The limited memory size of LBFGS method that is executed starting from the optimal value that was obtained after the Genetic Algorithm optimization

3.1.6 Custom Kriging options

When a user-defined Kriging metamodel is to be created, the `Metaopts.Kriging` field must be set with the following parameters (notice that all are *mandatory* in this case):

Table 16: Metaopts.Kriging

●	.Trend	Table 6	The Trend definition
●	.beta	Double	The values of the regression coefficients $\boldsymbol{\beta}$ in Eq. (2.2)
●	.sigmaSQ	Double	The value of σ^2 in Eq. (2.2)
●	.theta	Double	The value of $\boldsymbol{\theta}$ in Eq. (2.2)
●	.Corr	Table 8	The correlation function definition *See note below*

Note: In the current version of UQLAB not all the options regarding the correlation function (as reported in Table 8) are available for custom Kriging. Only the following are supported:

- `.Type` : mandatory
- `.Family` : mandatory
- `.Isotropic` : optional. By default anisotropic correlation function is assumed.

If the metamodel has more than one outputs then `Metaopts.Kriging` is expected to be a *structure array* with length equal to the number of outputs and the parameters in Table 16 defined for each `Metaopts.Kriging(index)` (where `index` ranges from 1 to the total number of outputs).

Note: For Custom Kriging the fields `Metaopts.ExpDesign.X` and `Metaopts.ExpDesign.Y` are also required (see Table 5).

For a usage example of custom Kriging, see Section 2.8.

3.1.7 Validation Set

If a validation set is provided, UQLAB automatically calculates the validation error of the created Kriging model. The required information is listed in Table 17.

Table 17: `MetaOpts.ValidationSet`

●	`.X`	$N \times M$ Double	User-specified validation set $\mathcal{X}_{Val}$
●	`.Y`	$N \times N_{Out}$ Double	User-specified validation set response $\mathcal{Y}_{Val}$

Output

Table 18: myKriging = uq_createModel(...)

.Name	String	The name of the Kriging metamodel
.Kriging	Table 20	Kriging results
.Internal	Table 22	Internal fields
.Error	Table 21	Error metrics of the metamodelling result
.Options	Table 4	The options that were defined in Metaopts variable
.ExpDesign	Table 19	Experimental Design related values

Table 19: myKriging.ExpDesign

.NSamples	Double	The number of samples
.Sampling	String	The sampling method
.X	$N \times M$ Double	The experimental design values
.muX	$1 \times M$ Double	The mean of the experimental design that was used for scaling. For more details see Section 2.9
.sigmaX	$1 \times M$ Double	The standard deviation of the experimental design that was used for scaling. For more details see Section 2.9
.U	$N \times M$ Double	The experimental design values in the auxiliary (scaled) space. For more details see Section 2.9
.Y	$N \times O$ Double	The output Y that corresponds to the input X

Table 20: myKriging.Kriging

.beta	$p \times 1$ Double	The value of $\boldsymbol{\beta}(\boldsymbol{\theta})$ in Eq. (2.2)
.sigmaSQ	Double	The value of $\sigma^2(\boldsymbol{\theta})$ in Eq. (2.2)
.theta	Variable size double	The value of $\boldsymbol{\theta}$ in Eq. (2.2)

Note: In general the fields myKriging.Kriging and myKriging.Error are structure arrays with length equal to the number of outputs of the metamodel.

Table 21: myKriging.Error

.LOO	Double	The Leave-One-Out error, calculated using Eq. (2.4)

`.Val`	Double	Validation error (see Eq. (1.32) and Section 2.5). Only available if a validation set is provided (see Table 17).

The Leave-one-out error is calculated using Eq. (2.4).

3.1.8 Internal fields (advanced)

Note: The internal fields of the Kriging module are not intended to be accessed or changed at most typical usage scenarios of the module. Also note that some internal fields are not documented here.

Table 22: `myKriging.Internal`

`.Runtime`	Structure	Variables that are used during the calculation of the kriging metamodel
`.Error`	Table 24	Internal fields related to the metamodelling error
`.ExpDesign`	Table 25	Internal fields related to the experimental design
`.Scaling`	Logical or `uq_input` object	The scaling of the experimental design. For more details see Section 2.9
`.Kriging`	Table 23	Internal fields with data related to the Kriging metamodel

Table 23: `myKriging.Internal.Kriging`

`.Trend`	Table 26	Internal fields related to the trend
`.GP`	Table 27	Internal fields related to the Gaussian process
`.Optim`	Table 28	Internal fields related to the Optimization of the hyperparameters
`.Cached`	Structure	Internal fields related to cached variables

Note: In general the fields `myKriging.Internal.Kriging` and `myKriging.Internal.Error` are structure arrays with length equal to the number of outputs of the metamodel. Therefore in order to access the parameters that correspond to the respective output one should use for example `myKriging.Internal.Kriging(k).field` where `k` $= 1, \ldots, N_{\text{out}}$.

Table 24: `myKriging.Internal.Error`

`.LOOmean`	$N \times 1$ Double	The numerator in Eq. (1.31)
`.LOOsd`	$N \times 1$ Double	The denominator in Eq. (1.31) (excluding K)

`.varY`	Double	The variance of the output Y. One can multiply this quantity with the relative LOO error in `myKriging.Error.LOO` to get the absolute value of the LOO error

Table 25: `myKriging.Internal.ExpDesign`

`.muX`	$1 \times M$ Double	The sample-based mean of each input component.
`.stdX`	$1 \times M$ Double	The sample-based standard deviation of each input component.
`.varY`	Double	The variance of the output Y. One can multiply this quantity with the relative LOO error in `myKriging.Error.LOO` to get the absolute value of the LOO error

Table 26: `myKriging.Internal.Kriging.Trend`

`.F`	$N \times p$ Double	The trend's basis function evaluated on the experimental design (see Eq. (2.2))
`.beta`	$p \times 1$ Double	The value of $\beta(\boldsymbol{\theta})$ in Eq. (2.2)

Table 27: `myKriging.Internal.Kriging.GP`

`.R`	$N \times N$ Double	The correlation matrix value
`.sigmaSQ`	Double	The value of $\sigma^2(\boldsymbol{\theta})$ in Eq. (2.2)

Table 28: `myKriging.Internal.Kriging.Optim`

`.Theta`	Double	The optimum value of $\boldsymbol{\theta}$ that was obtained
`.ObjFun`	Double	The objective function value when $\boldsymbol{\theta}$ is optimum
`.InitialObjFun`	Double	The value of the objective function using the initial estimate of $\boldsymbol{\theta}$
`.nEval`	Double	The number of objective function evaluations that took place during the optimization
`.nIter`	Double	• The number of iterations (or generations) that took place during the optimization • For hybrid methods only the number of generations is taken into account

3.2 Kriging predictor

Syntax

```
 Y_mu = uq_evalModel(X)
 Y_mu = uq_evalModel(myKriging, X)
[Y_mu, Y_sigma2] = uq_evalModel(...)
[Y_mu, Y_sigma2, Y_cov] = uq_evalModel(...)
```

Description

`Y_mu = uq_evalModel(X)` returns the mean of the Kriging predictor ($N \times N_{\text{out}}$ Double) on the points `X` ($N \times M$ Double) using the last created Kriging model.

Note: by default, the *last created* model or surrogate model is the currently active model.

`Y_mu = uq_evalModel(myKriging,X)` returns the mean of the Kriging predictor ($N \times N_{\text{out}}$ Double) on the points `X` ($N \times M$ Double) using the Kriging metamodel object `myKriging`.

`[Y_mu, Y_sigma2] = uq_evalModel(...)` additionally returns the variance of the Kriging predictor ($N \times N_{\text{out}}$ Double).

`[Y_mu, Y_sigma2, Y_cov] = uq_evalModel(...)` additionally returns the covariance matrix of the Kriging predictor ($N \times N \times N_{\text{out}}$ Double). Note that the diagonal elements of `Y_cov` (in each output dimension) are equal to the corresponding elements in `Y_sigma2`, *i.e.* `Y_sigma2 = diag(Y_cov)`.

3.3 Printing/visualizing a Kriging metamodel

UQLAB offers two commands to conveniently print reports containing contextually relevant information for a given object.

3.3.1 Printing the results: uq_print

Syntax

```
uq_print(myKriging);
uq_print(myKriging,outidx);
uq_print(myKriging,outidx, option1, option2, ...);
```

Description

`uq_print(myKriging)` prints a report of the configuration options and results of the Kriging metamodel object `myKriging`. If the model has multiple outputs, only the results for the first output variable are printed.

`uq_print(myAnalysis, outidx)` prints a report of the configuration options and results of the the Kriging metamodel object `myKriging` for the output variables specified in the array `outidx`.

`uq_print(myKriging,outidx, option1, option2, ...)` prints a report *only* of the configuration options or results that are specified via the options `option1, option2, ...`etc, of the Kriging metamodel object `myKriging` for the output variables specified in the array `outidx`. See Table Table 29 for the available options.

Table 29: `uq_print` options

`'beta'`	Prints out the regression coefficients $\boldsymbol{\beta}$.
`'theta'`	Prints out the final hyperparameters $\boldsymbol{\theta}$ value(either the optimal value or the user-specified value if the user manually specified it).
`'F'`	Prints out the information matrix $\boldsymbol{F}$.
`'R'`	Prints out the correlation matrix $\boldsymbol{R}$.

Examples:

`uq_print(myKriging, [1 3])` will print the Kriging metamodelling results for output variables 1 and 3.

`uq_print(myKriging, 3, 'theta','R')` will only print the hyperparameters $\boldsymbol{\theta}$ and correlation matrix $\boldsymbol{R}$ values for output variable 3.

3.3.2 Graphically display the results: uq_display

Syntax

```
uq_display(myKriging);
uq_display(myKriging, outidx);
uq_display(myKriging, outidx, 'R');
```

Description

`uq_display(myKriging)` create a visualization of the results of the Kriging metamodel in object `myKriging`, if possible. If the model has multiple outputs, only the results for the first output variable are visualized.

`uq_display(myKriging, outidx)` create a visualization of the results of the Kriging metamodel in object `myKriging` for the output variables specified in the array `outidx`.

`uq_display(myKriging, outidx, 'R')` create a visualization of the correlation matrix $\boldsymbol{R}$ in object `myKriging` for the output variables specified in the array `outidx`.

Note: For visualizing the Kriging metamodelling results using `uq_display` the input should be either 1- or 2-dimensional. However `uq_display(myKriging, outidx, 'R')` can be used for arbitrary input dimension.

Examples:

`uq_display(myKriging, [1 3])` will display the Kriging metamodelling results for output variables 1 and 3.

References

Abramovitz, M. and I. A. Stegun (1965). *Handbook of mathematical functions*. New York: Dover Publications Inc. 7

Bachoc, F. (2013). Cross-validation and maximum likelihood estimations of hyper-parameters of Gaussian processes with model misspecification. *Computational Statistics and Data Analysis 66*, 55–69. 10

Cressie, N. A. C. (1993). *Statistics for spatial data*. John Wiley & Sons Inc. 10

Dubourg, V. (2011). *Adaptive surrogate models for reliability analysis and reliability-based design optimization*. Ph. D. thesis, Université Blaise Pascal, Clermont-Ferrand, France. 2, 3, 5, 9

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional. 11

Hansen, N. and A. Ostermeier (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation 9*(2), 159–195. 11

Nocedal, J. (1980). Updating Quasi-Newton Matrices with Limited Storage. *Mathematics of Computation 35*(151), 773–782. 11

Rasmussen, C. and C. Williams (2006). *Gaussian processes for machine learning*. Adaptive computation and machine learning. Cambridge, Massachusetts: MIT Press. 5, 8

Richard H. Byrd, Mary E. Hribar Y, J. N. Z. (1999). An interior point algorithm for large scale nonlinear programming. *SIAM Journal on Optimization 9*(4), 877–900. 11

Sacks, J., W. J. Welch, T. J. Mitchell, and H. P. Wynn (1989). Design and analysis of computer experiments. *Statistical Science 4*, 409–435. 5

Santner, T., B. Williams, and W. Notz (2003). *The design and analysis of computer experiments*. Springer series in Statistics. Springer. 1, 2, 9, 10