ETH zürich

UQ[PY]LAB USER MANUAL - PYTHON KRIGING (GAUSSIAN PROCESS MODELING)

C. Lataniotis, D. Wicaksono, 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 UQ[PY]LAB

C. Lataniotis, S. Marelli, B. Sudret, Uncertainty Quantification in the cloud with UQCloud Proceedings of the 4th International Conference on Uncertainty Quantification in Computational Sciences and Engineering (UNCECOMP 2021), Athens, Greece, June 27–30, 2021.

How to cite this manual

C. Lataniotis, D. Wicaksono, S. Marelli, B. Sudret, UQ[py]Lab user manual - Python – Kriging (Gaussian process modeling), Report # UQ[py]Lab -V0.9-105, Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland, 2022

BIBTEX entry

```
@TechReport{UQdoc_09_105,
author = {Lataniotis, C. and Wicaksono, D. and Marelli, S. and Sudret, B.},
title = {{UQLab user manual -- Kriging (Gaussian process modeling) }},
institution = {Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich,
Switzerland},
year = {2022},
note = {Report \#  UQ[py]Lab -V0.9-105}
}
```

List of contributors:

Name	Contribution
A. Giannoukou, A. Hlobilová	Translation from the UQLab manual

Document Data Sheet

Document Ref.	UQ[PY]LAB-V0.9-105
Title:	UQ[PY]LAB user manual - Python – Kriging (Gaussian process modeling)
Authors:	C. Lataniotis, D. Wicaksono, S. Marelli, B. Sudret Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland
Date:	06/12/2022

Doc. Version	Date	Comments
V0.9	06/12/2022	Updated figures and other minor updates in the document structure
V0.8	28/09/2022	Initial release

Abstract

Kriging is a stochastic modeling algorithm that has many applications in most fields of engineering and applied mathematics. The UQ[PY]LAB metamodeling tool provides an efficient, modular, and easy to use Kriging module that allows one to obtain efficient Kriging predictors with minimal effort. For experienced users, numerous more advanced configuration options can be easily set.

This guide is an extensive manual of the Kriging metamodeling module and divided into three parts:

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

Keywords: UQCloud, UQ[PY]LAB, Kriging, Gaussian process metamodeling, Gaussian process regression

Contents

1 Theory **1**

1.1 Introduction . . . 1

1.2 Kriging basics . . . 2

1.2.1 Prediction with noise-free responses (interpolation) . . . 2

1.2.2 Prediction with noisy responses (regression) . . . 3

1.2.3 Kriging metamodel ingredients . . . 5

1.3 Trend types . . . 6

1.3.1 Commonly used trends . . . 7

1.4 Correlation functions . . . 7

1.4.1 Correlation families . . . 8

1.4.2 Correlation function types . . . 11

1.4.3 Isotropic correlation functions . . . 11

1.4.4 Nugget and numerical stability . . . 12

1.5 Estimation methods . . . 12

1.5.1 Maximum-likelihood estimation . . . 12

1.5.2 Cross-validation estimation . . . 14

1.6 Optimization methods . . . 16

1.7 *A posteriori* error estimation . . . 17

1.7.1 Leave-one-out cross-validation error . . . 17

1.7.2 Validation error . . . 17

2 Usage **19**

2.1 Reference problem . . . 19

2.2 Problem setup . . . 19

2.3 Kriging metamodel calculation: noise-free case . . . 20
2.3.1 Accessing the results . . . 22
2.4 Kriging metamodel setup . . . 23
2.4.1 Specification and generation of experimental design . . . 24
2.4.2 Trend . . . 25
2.4.3 Correlation function . . . 27
2.4.4 Estimation methods . . . 30
2.4.5 Optimization methods . . . 31
2.5 Kriging metamodels with noise (regression) . . . 33
2.5.1 Unknown homoscedastic noise . . . 33
2.5.2 Known noise . . . 35
2.6 Kriging metamodels of vector-valued models . . . 37
2.6.1 Accessing the results . . . 37
2.7 Using a validation set . . . 38
2.8 Using Kriging predictor as a model . . . 38
2.9 Specifying manually a Kriging predictor (*predictor-only* mode) . . . 38
2.10 Performing Kriging on an auxiliary space (scaling) . . . 39
2.11 Drawing sample paths from a Gaussian process posterior . . . 40
2.12 Using Kriging with constant inputs . . . 41

3 Reference List **43**
3.1 Create a Kriging metamodel . . . 46
3.1.1 Experimental design options . . . 47
3.1.2 Trend options . . . 48
3.1.3 Correlation function options . . . 49
3.1.4 Estimation method options . . . 50
3.1.5 Hyperparameters optimization options . . . 50
3.1.6 Regression options . . . 53
3.1.7 Custom Kriging options . . . 54
3.1.8 Validation Set . . . 55
3.2 Accessing the Results . . . 56

3.2.1 Internal fields (advanced) 57
3.3 Kriging predictor 61
3.4 Printing and visualizing a Kriging metamodel 62
3.4.1 Printing the results: `uq.print` 62
3.4.2 Visualize the results: `uq.display` 64
References 65

Chapter 1

Theory

1.1 Introduction

In modern applied sciences and engineering, computational models have become more and more complex and hence expensive to evaluate. In this context, *metamodeling* (or *surrogate modeling*) can reduce the associated computational costs by substituting an expensive computational model with a metamodel, a functional approximation of the original model that is much faster to evaluate. It is constructed by learning the approximation from a relatively small set of input parameters and their corresponding model responses generated from running the original expensive model. Because it is faster to evaluate, a metamodel allows for more sophisticated analyses, including, e.g., sensitivity and reliability analysis, Bayesian model calibration, etc.

In its original form, Kriging (also known as *Gaussian process modeling*) is a statistical interpolation method that capitalizes on Gaussian processes to interpolate a wide range of complex functions. It was first developed as a spatial interpolation tool in geostatistics by Krige dating back to the 1950s (Krige, 1951) and formalized by Matheron in the 1960s (Matheron, 1963). Kriging was later introduced in the context of metamodeling and computer experiments in the work of Sacks et al. (1989) in which Kriging was used to represent an input/output mapping of an expensive computational model. In the mid-2000s, Gaussian processes enjoyed renewed interest due to their applications in machine learning regression and classification, thanks to the introduction of GP-regression, which supports noisy data(Rasmussen and Williams, 2006). The reader is referred to Santner et al. (2003) for an in-depth introduction of Kriging as a metamodeling tool and to Rasmussen and Williams (2006) for a similar introduction to Kriging as a regression and classification tool.

This user manual presents the Kriging metamodeling tool of UQ[PY]LAB that supports Kriging for both interpolation and regression. This chapter briefly presents the basics of Kriging, including its formulation, main ingredients, as well as the estimation of its parameters from data. Section 2 then details the usage of Kriging within UQ[PY]LAB. Finally, Section 3 provides a comprehensive list of the available commands and options.

1.2 Kriging basics

Kriging (or Gaussian process modeling) is a stochastic 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 (*i.e.*, its *trend*); it consists of P arbitrary functions $\{f_j; j = 1, \ldots, P\}$ and the corresponding coefficients $\{\beta_j; j = 1, \ldots, P\}$. The second term in Eq. (1.1) consists of the (constant) variance of the Gaussian process σ^2 and a zero-mean, unit-variance, stationary Gaussian process $Z(\boldsymbol{x}, \omega)$. The underlying probability space is represented by ω and is defined in terms of a *correlation function* R (*i.e.*, correlation family) and its hyperparameters $\boldsymbol{\theta}$. The correlation function $R = R(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta})$, in turn, describes the correlation between two sample points in the output space, that depends on $\boldsymbol{x}$, $\boldsymbol{x}'$, and the hyperparameters $\boldsymbol{\theta}$.

1.2.1 Prediction with noise-free responses (interpolation)

In the context of metamodeling, it is of interest to predict $\mathcal{M}^K(\boldsymbol{x})$ for a new point $\boldsymbol{x}$, given the (observed) experimental design $\mathcal{X} = \left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\right\}$ and the corresponding noise-free model responses $\mathcal{Y} = \left\{y^{(1)} = \mathcal{M}\left(\boldsymbol{x}^{(1)}\right), \ldots, y^{(N)} = \mathcal{M}\left(\boldsymbol{x}^{(N)}\right)\right\}$. A Kriging metamodel (or Kriging predictor) provides the prediction based on the Gaussian properties of the process.

The Gaussian assumption states that the vector formed by the prediction at $\boldsymbol{x}$, *i.e.*, $\widehat{Y}(\boldsymbol{x})$ and the true model responses $\mathcal{Y}$, has a joint Gaussian distribution defined by

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

where:

- $\boldsymbol{F}$ is the observation (design) matrix of the Kriging metamodel trend. It reads

$$F_{ij} = f_j\left(\boldsymbol{x}^{(i)}\right), \; i = 1, \ldots, N; \; j = 0, \ldots, P. \tag{1.3}$$

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

$$r_i = R\left(\boldsymbol{x}, \boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right), \; i = 1, \ldots, N. \tag{1.4}$$

- $\boldsymbol{R}$ is the correlation matrix with elements:

$$R_{ij} = R\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}; \boldsymbol{\theta}\right), \; i, j = 1, \ldots, N. \tag{1.5}$$

Consequently, the mean and variance of the Gaussian random variate $\widehat{Y}(\boldsymbol{x})$ conditional on the observed data $\mathcal{X}$ and $\mathcal{Y}$ can be computed as follows (Santner et al., 2003; Dubourg, 2011)

$$\mu_{\widehat{Y}}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x})^T \widehat{\boldsymbol{\beta}} + \boldsymbol{r}(\boldsymbol{x})^T \boldsymbol{R}^{-1} \left(\mathcal{Y} - \boldsymbol{F}\widehat{\boldsymbol{\beta}}\right), \tag{1.6}$$

$$\sigma^2_{\widehat{Y}}(\boldsymbol{x}) = \sigma^2 \left(1 - \boldsymbol{r}^T(\boldsymbol{x}) \boldsymbol{R}^{-1} \boldsymbol{r}(\boldsymbol{x}) + \boldsymbol{u}^T(\boldsymbol{x}) \left(\boldsymbol{F}^T \boldsymbol{R}^{-1} \boldsymbol{F}\right)^{-1} \boldsymbol{u}(\boldsymbol{x})\right), \tag{1.7}$$

where

$$\widehat{\boldsymbol{\beta}} = \left(\boldsymbol{F}^T \boldsymbol{R}^{-1} \boldsymbol{F}\right)^{-1} \boldsymbol{F}^T \boldsymbol{R}^{-1} \mathcal{Y} \tag{1.8}$$

is the generalized least-squares estimate of $\boldsymbol{\beta}$ and

$$\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{F}^T \boldsymbol{R}^{-1} \boldsymbol{r}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x}). \tag{1.9}$$

Eqs. (1.6) and (1.7) are referred to as the mean and variance of the *Kriging predictor*, respectively. An important property of this particular predictor is that the variance of the prediction at an experimental design point $\boldsymbol{x} \in \mathcal{X}$ collapses to zero. In other words, the Kriging predictor is an *interpolant* with respect to the experimental design points.

Another useful corollary of the Gaussian assumption is that

$$\widehat{Y}(\boldsymbol{x}) \sim \mathcal{N}\left(\mu_{\widehat{Y}}(\boldsymbol{x}), \sigma^2_{\widehat{Y}}(\boldsymbol{x})\right) \tag{1.10}$$

and therefore

$$\mathbb{P}\left(\widehat{Y}(\boldsymbol{x}) \leq t\right) = \Phi\left(\frac{t - \mu_{\widehat{Y}}(\boldsymbol{x})}{\sigma_{\widehat{Y}}(\boldsymbol{x})}\right), \tag{1.11}$$

where $\Phi(\cdot)$ denotes the Gaussian cumulative density function. Note that this is the probability that $\widehat{Y}(\boldsymbol{x})$ is smaller than t *at a given* $\boldsymbol{x}$[1]. Based on Eq. (1.11), the confidence intervals on the predictor can be calculated by

$$\widehat{Y}(\boldsymbol{x}) \in \left[\mu_{\widehat{Y}}(\boldsymbol{x}) - \Phi^{-1}\left(1 - \frac{\alpha}{2}\right)\sigma_{\widehat{Y}}(\boldsymbol{x}) \;,\; \mu_{\widehat{Y}}(\boldsymbol{x}) + \Phi^{-1}\left(1 - \frac{\alpha}{2}\right)\sigma_{\widehat{Y}}(\boldsymbol{x})\right] \tag{1.12}$$

with probability $1 - \alpha$.

1.2.2 Prediction with noisy responses (regression)

There are cases in which the observed responses are noisy, that is

$$y = \mathcal{M}(\boldsymbol{x}) + \varepsilon, \tag{1.13}$$

where it is often assumed that the additive noise ε follows a zero-mean Gaussian distribution

$$\varepsilon \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{\mathrm{n}}), \tag{1.14}$$

where $\boldsymbol{\Sigma}_{\mathrm{n}}$ is the covariance matrix of the noise term.

[1] This shall not be confused with a probability of failure computed in the framework of rare event simulation. For details, see UQ[PY]LAB User Manual – Structural Reliability.

Depending on the properties of $\boldsymbol{\Sigma}_{\mathrm{n}}$, three classes of noise can be identified:

- $\boldsymbol{\Sigma}_{\mathrm{n}} = \sigma_{\mathrm{n}}^2 \boldsymbol{I}$ (where $\boldsymbol{I}$ is an identity matrix) corresponds to homogeneous (*homoscedastic*) noise, in which the variance σ_{n}^2 is constant and the same for all observed responses. In other words, the noise is an independent and identically distributed (*iid*) Gaussian.
- $\boldsymbol{\Sigma}_{\mathrm{n}} = \operatorname{diag}\left(\boldsymbol{\sigma}_{\mathrm{n}}^2\right)$ corresponds to *independent heterogeneous (heteroscedastic) noise*, in which the noise variances $\boldsymbol{\sigma}_n^2$ differ for each observed response but are not correlated.
- $\boldsymbol{\Sigma}_{\mathrm{n}}$ corresponds to *general heteroscedastic noise*, in which the noise variance can differ for each observed response and correlations between observations are possible.

The joint Gaussian distribution formed by the prediction at $\boldsymbol{x}$, *i.e.*, $\widehat{Y}(\boldsymbol{x})$ and the observed, albeit noisy, model responses $\mathcal{Y}$ now becomes:

$$\begin{Bmatrix} \widehat{Y}\left(\boldsymbol{x}\right) \\ \mathcal{Y} \end{Bmatrix} \sim \mathcal{N}_{N+1}\left(\begin{Bmatrix} \boldsymbol{f}^T\left(\boldsymbol{x}\right)\boldsymbol{\beta} \\ \boldsymbol{F}\boldsymbol{\beta} \end{Bmatrix}, \begin{Bmatrix} \sigma^2 & \sigma^2\boldsymbol{r}^T\left(\boldsymbol{x}\right) \\ \sigma^2\boldsymbol{r}\left(\boldsymbol{x}\right) & \sigma^2\boldsymbol{R}+\boldsymbol{\Sigma}_{\mathrm{n}} \end{Bmatrix}\right). \tag{1.15}$$

Because in general the noise variance cannot be factored out from the covariance matrix (*i.e.*, due to heteroscedasticity), the covariance matrix $\boldsymbol{C} = \sigma^2\boldsymbol{R} + \boldsymbol{\Sigma}_{\mathrm{n}}$ is used in the subsequent formulation. Consequently, the Kriging mean and variance predictors (see Eq (1.6) and (1.7)) become

$$\mu_{\widehat{Y}}\left(\boldsymbol{x}\right) = \boldsymbol{f}\left(\boldsymbol{x}\right)^T\widehat{\boldsymbol{\beta}} + \boldsymbol{c}(\boldsymbol{x})^T\boldsymbol{C}^{-1}\left(\mathcal{Y} - \boldsymbol{F}\widehat{\boldsymbol{\beta}}\right), \tag{1.16}$$

$$\sigma_{\widehat{Y}}^2\left(\boldsymbol{x}\right) = \left(\sigma^2 - \boldsymbol{c}^T\left(\boldsymbol{x}\right)\boldsymbol{C}^{-1}\boldsymbol{c}(\boldsymbol{x}) + \boldsymbol{u}_c^T(\boldsymbol{x})(\boldsymbol{F}^T\boldsymbol{C}^{-1}\boldsymbol{F})^{-1}\boldsymbol{u}_c\left(\boldsymbol{x}\right)\right) \tag{1.17}$$

where $\boldsymbol{c} = \sigma^2\boldsymbol{r}\left(\boldsymbol{x}\right)$ is the cross-covariance vector,

$$\widehat{\boldsymbol{\beta}} = \left(\boldsymbol{F}^T\boldsymbol{C}^{-1}\boldsymbol{F}\right)^{-1}\boldsymbol{F}^T\boldsymbol{C}^{-1}\mathcal{Y}, \tag{1.18}$$

is the generalized least-square estimate of the regression coefficients $\widehat{\boldsymbol{\beta}}$, and

$$\boldsymbol{u}_c\left(\boldsymbol{x}\right) = \boldsymbol{F}^T\boldsymbol{C}^{-1}\boldsymbol{c}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x}). \tag{1.19}$$

In the special case of homoscedastic noise, $\boldsymbol{C} = \sigma^2\boldsymbol{R} + \sigma_{\mathrm{n}}^2\boldsymbol{I}$, thus with

$$\sigma_{\text{total}}^2 = \sigma^2 + \sigma_{\mathrm{n}}^2 \tag{1.20}$$

and

$$\tau = \frac{\sigma_n^2}{\sigma_{\text{total}}} \tag{1.21}$$

Eq. (1.15) can be written as

$$\begin{Bmatrix} \widehat{Y}\left(\boldsymbol{x}\right) \\ \mathcal{Y} \end{Bmatrix} \sim \mathcal{N}_{N+1}\left(\begin{Bmatrix} \boldsymbol{f}^T\left(\boldsymbol{x}\right)\boldsymbol{\beta} \\ \boldsymbol{F}\boldsymbol{\beta} \end{Bmatrix}, \sigma_{\text{total}}^2\begin{Bmatrix} (1-\tau) & \widetilde{\boldsymbol{r}}^T\left(\boldsymbol{x}\right) \\ \widetilde{\boldsymbol{r}}\left(\boldsymbol{x}\right) & \widetilde{\boldsymbol{R}} \end{Bmatrix}\right) \tag{1.22}$$

where

$$\widetilde{\boldsymbol{r}} = (1-\tau)\,\boldsymbol{r} \tag{1.23}$$

and

$$\widetilde{\boldsymbol{R}} = (1-\tau)\,\boldsymbol{R} + \tau \boldsymbol{I}. \tag{1.24}$$

In this case, the mean and variance of the Gaussian random variate $\widehat{Y}\,(\boldsymbol{x})$ can be computed as follows (Rasmussen and Williams, 2006)

$$\mu_{\widehat{Y}}\,(\boldsymbol{x}) = \boldsymbol{f}\,(\boldsymbol{x})^T\,\widehat{\boldsymbol{\beta}} + \widetilde{\boldsymbol{r}}(\boldsymbol{x})^T\,\widetilde{\boldsymbol{R}}^{-1}\left(\mathcal{Y} - \boldsymbol{F}\widehat{\boldsymbol{\beta}}\right) \tag{1.25}$$

$$\sigma^2_{\widehat{Y}}\,(\boldsymbol{x}) = \sigma^2_{\text{total}}\left(1 - \widetilde{\boldsymbol{r}}^T\,(\boldsymbol{x})\,\widetilde{\boldsymbol{R}}^{-1}\widetilde{\boldsymbol{r}}(\boldsymbol{x}) + \boldsymbol{u}^T(\boldsymbol{x})(\boldsymbol{F}^T\widetilde{\boldsymbol{R}}^{-1}\boldsymbol{F})^{-1}\boldsymbol{u}(\boldsymbol{x})\right) \tag{1.26}$$

where $\widehat{\boldsymbol{\beta}}$ and $\boldsymbol{u}$ in the above follow Eqs. (1.8) and (1.9), but with $\boldsymbol{R}$ and $\boldsymbol{r}$ are replaced by $\widetilde{\boldsymbol{R}}$ and $\widetilde{\boldsymbol{r}}$, respectively.

Unlike in the noise-free case in Section 1.2.1, the variance of the prediction at an experimental design point $\boldsymbol{x} \in \mathcal{X}$ does not collapse to zero and the Kriging predictor becomes a *regression* model. Other corollaries following the Gaussian assumptions, however, still apply.

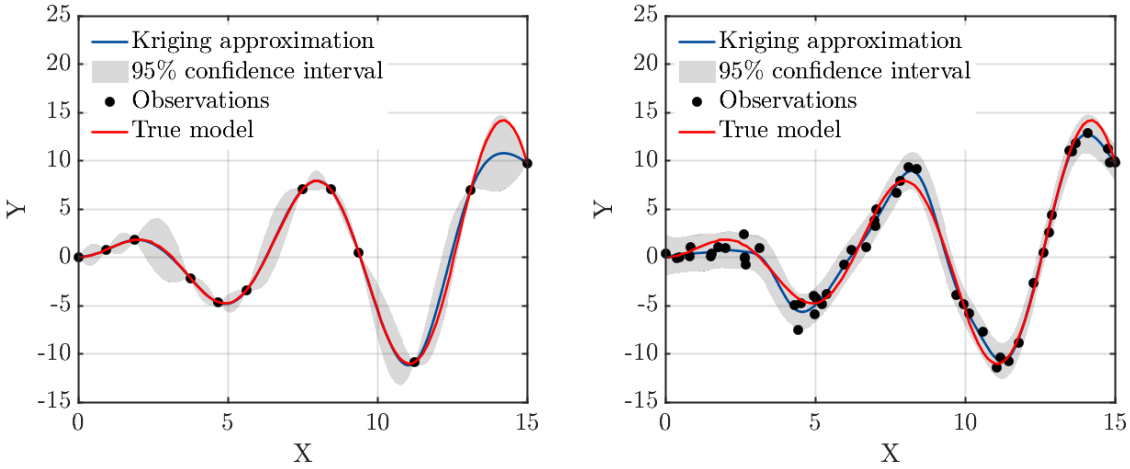
1.2.3 Kriging metamodel ingredients

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

- Select a functional basis of the Kriging trend. This is further discussed in Section 1.3.
- Select a correlation function $R(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta})$. This is further discussed in Section 1.4.
- If the hyperparameters $\boldsymbol{\theta}$ are unknown, they need to be estimated from the available data. This involves setting up an optimization problem (see Section 1.5) and solving it (see Section 1.6).
- In the case of regression, then either the noise variance σ^2_{n} (if it is unknown) or the Gaussian process variance σ^2 (if σ^2_{n} is known) need to be estimated.
- Using the optimal value of $\boldsymbol{\theta}$, calculate the rest of the unknown Kriging parameters (*e.g.*, $\boldsymbol{\beta}$) and, if necessary, σ^2.

Predictions at new points can then be made in terms of the mean and variance of $\widehat{Y}(\boldsymbol{x})$.

Two examples of one-dimensional Kriging metamodels are shown in Figure 1. They are constructed using two sets of experimental design and the corresponding responses; both sets come from the same underlying model. In the case of noise-free responses (Figure 1(a)), the Kriging predictor returns the mean and the variance of a Gaussian process that interpolates the design points. In the case of noisy responses (Figure 1(b)), the predictor serves as a regression model that fits the noisy responses. Using Eq. (1.12), the 95% confidence intervals of both predictors are also calculated and shown in the plot.



(a) noise-free responses (interpolation)

(b) noisy responses (regression)

Figure 1: Examples of one-dimensional Kriging metamodels with two different responses.

1.3 Trend types

The trend refers to the mean of a Kriging metamodel, that is, the term $\boldsymbol{\beta}^T \boldsymbol{f}(\boldsymbol{x})$ in Eq. (1.1). While using a non-zero trend is optional, it is commonly preferred[2]. 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**
  In simple Kriging, the trend is known

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

  where f_j's are arbitrary but fully specified functions. Note that no estimation on $\boldsymbol{\beta}$ is taken place and the coefficients $\boldsymbol{\beta}$ are all 1's

- **Ordinary Kriging**
  In ordinary Kriging, the trend has a constant yet unknown value

$$\boldsymbol{\beta}^T \boldsymbol{f}(\boldsymbol{x}) = \beta_0 f_0(\boldsymbol{x}) = \beta_0$$

  where by convention $f_0(\boldsymbol{x}) = 1$.

- **Universal Kriging**
  Universal Kriging, the most general and flexible formulation, assumes that the trend is

[2]Note that the mean of the Kriging predictor given in Eq. (1.6), (1.16), or (1.25) is not confined to be zero even though the trend is zero.

a linear combination of prescribed arbitrary functions (*e.g.*, polynomials)

$$\boldsymbol{\beta}^T \boldsymbol{f}(\boldsymbol{x}) = \sum_{j=0}^{P} \beta_j f_j(\boldsymbol{x}).$$

According to the above, the simple and ordinary Kriging are special cases of universal Kriging. In the current version of UQ[PY]LAB, the functions f_j's can either be constants, polynomials, or any other user-defined functions.

1.3.1 Commonly used trends

The most commonly used trends are based on polynomial basis and summarized in Table 1.

Table 1: Trend options combinations

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

Based on the Table 1, a multivariate polynomial trends can be written general form as follows

$$\boldsymbol{\beta}^T \boldsymbol{f}(\boldsymbol{x}) = \sum_{\boldsymbol{\alpha} \in \mathcal{A}^{M,P}} \beta_{\boldsymbol{\alpha}} f_{\boldsymbol{\alpha}}(\boldsymbol{x}) \ , \ f_{\boldsymbol{\alpha}}(\boldsymbol{x}) = \prod_{i=1}^{M} x_i^{\alpha_i} \tag{1.27}$$

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

The Kriging module in UQ[PY]LAB offers the possibility to use a multivariate polynomial of arbitrary degree P (Eq. (1.27)) as well as different types of basis functions. See Section 2.4.2 for more information and Section 3.1.2 for a list of all the available options regarding the trend.

1.4 Correlation functions

The correlation function (also called the *kernel* or *covariance*[3] function in the literature) is a crucial ingredient for a Kriging predictor, since it contains the assumptions about the approximation function. Generally speaking, it describes the "similarity" between observations and new points, *i.e.*, how similar such points are, depending on the distance between the input points. In this sense, input points that are close to each other are expected to have similar outputs or responses.

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:

[3]The covariance function is to the correlation function multiplied by the Gaussian process variance σ^2.

- The correlation matrix[4] whose elements are computed as $R_{ij} = R\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right)$, $(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{X} \times \mathcal{X}$ is positive semi-definite for any choice of number of sample points N in experimental design $\mathcal{X}$.
- The function 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, a correlation function 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 parameter might correspond to each input dimension. For notational clarity, however, it is assumed that one element of $\boldsymbol{\theta}$ is used per input dimension in the subsequent discussion.

1.4.1 Correlation families

All the correlation families described below are *stationary* correlation functions that depend only on the relative position of its two inputs. Moreover, these families are one-dimensional and defined for a pair of one-dimensional input $x, x' \in \mathbb{R}$ and parametrized by $\theta \in \mathbb{R}_{>0}$, often referred to as the *characteristic length scale* or simply the *scale* parameter.

For each of the available correlation families in UQ[PY]LAB, the function is plotted in Figures 2-6 with different scale values, along with the corresponding sample paths (*i.e.*, realizations) of a zero-mean, unit-variance Gaussian process having this correlation function.

- **Linear**:

$$R\left(x, x'; \theta\right) = \max\left(0, 1 - \frac{|x - x'|}{\theta}\right). \tag{1.28}$$

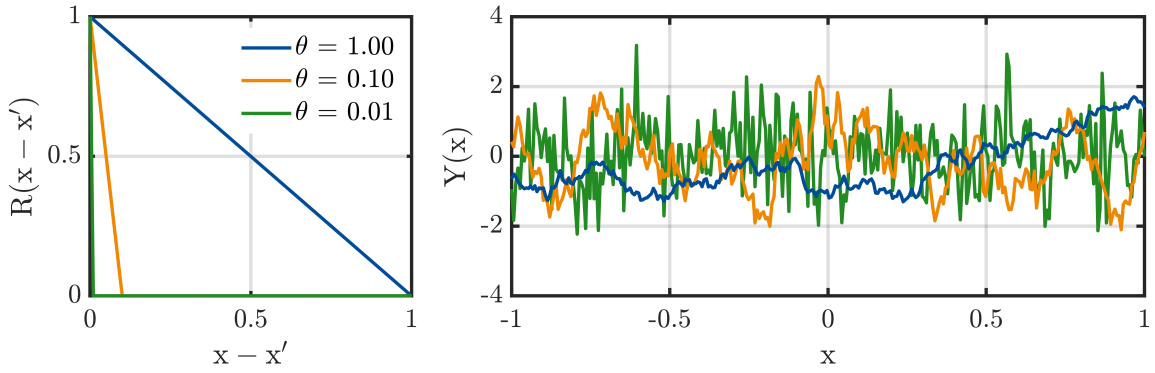


Figure 2: The linear correlation function (Eq. (1.28)) and sample paths drawn from the corresponding zero-mean unit-variance Gaussian process for various scale parameters.

- **Exponential**:

$$R\left(x, x'; \theta\right) = \exp\left[-\frac{|x - x'|}{\theta}\right]. \tag{1.29}$$

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

[4] It is also a Gram matrix.

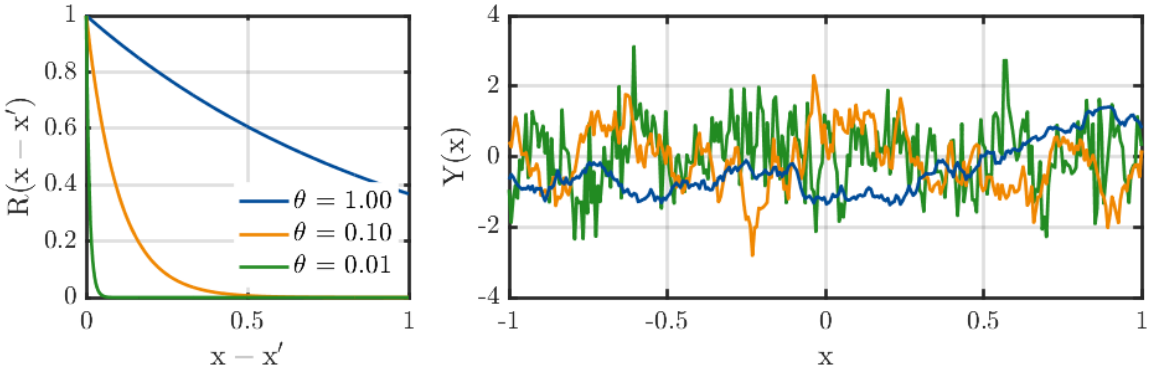


Figure 3: The exponential correlation function (Eq. (1.29)) and sample paths drawn from the corresponding zero-mean unit-variance Gaussian process for various scale parameters.

- **Gaussian** (or squared exponential):

$$R\left(x,x';\theta\right)=\exp\left[-\frac{1}{2}\left(\frac{|x-x'|}{\theta}\right)^2\right]. \tag{1.30}$$

The resulting sample paths of the corresponding process are infinitely differentiable.

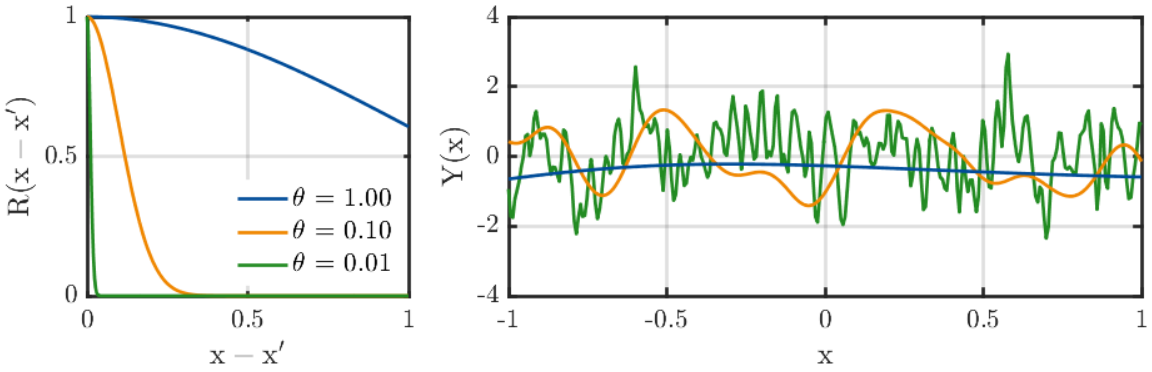


Figure 4: The Gaussian correlation function (Eq. (1.30)) and sample paths of the corresponding zero-mean unit-variance Gaussian process for various scale parameters.

- **Matérn**: The general form of the Matérn correlation function is given by

$$R\left(x,x';\theta,v\right)=\frac{1}{2^{v-1}\Gamma(v)}\left(2\sqrt{v}\frac{|x-x'|}{\theta}\right)^{v}\mathcal{K}_v\left(2\sqrt{v}\frac{|x-x'|}{\theta}\right) \tag{1.31}$$

where θ is the correlation function scale parameter, $v\geq 1/2$ is the shape parameter, Γ is the Euler's Gamma function, and $\mathcal{K}_\nu$ is the modified Bessel function of the second kind[5].

An interesting feature of this correlation function family 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

[5]It is also also known as the Bessel function of the third kind. For details, see Abramovitz and Stegun (1965).

function which generates $\mathcal{C}^0$ sample paths. For $v \to \infty$, it tends towards the Gaussian correlation function which generates $\mathcal{C}^\infty$ sample paths.

The Matérn functions can be computed using simplified formulas when $v = p + 1/2$, where p is a non-negative integer (Rasmussen and Williams, 2006). The Matérn functions with $v = 3/2$ and $5/2$ (abbreviated as Matérn-3/2 and Matérn-5/2, respectively) are the most commonly used and available in UQ[PY]LAB. Their formulas are given below.

For $v = 3/2$

$$R\left(h;\theta,v=3/2\right) = \left(1+\sqrt{3}\frac{|x-x'|}{\theta}\right)\exp\left[-\sqrt{3}\frac{|x-x'|}{\theta}\right], \tag{1.32}$$

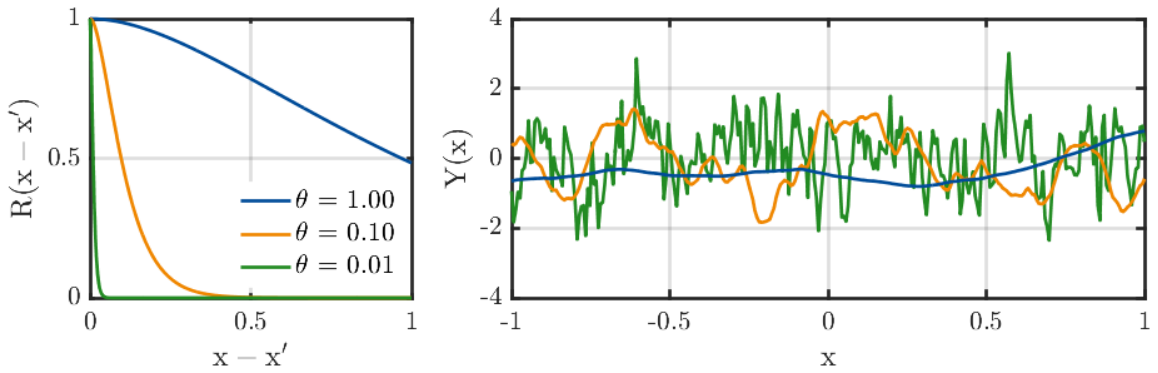


Figure 5: The Matérn $3/2$ correlation function (Eq. (1.32)) and sample paths drawn from the corresponding zero-mean unit-variance Gaussian process for various scale parameters.

and for $v = 5/2$

$$R(x,x';\theta,v=5/2) = \left(1+\sqrt{5}\frac{|x-x'|}{\theta}+\frac{5}{3}\left(\frac{|x-x'|}{\theta}\right)^2\right)\exp\left[-\sqrt{5}\frac{|x-x'|}{\theta}\right]. \tag{1.33}$$

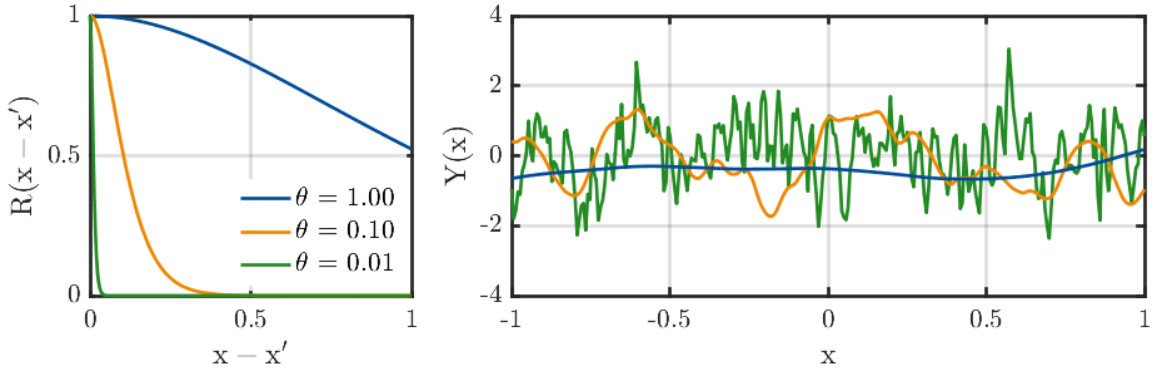


Figure 6: The Matérn $5/2$ correlation function (Eq. (1.33)) and sample paths drawn from the corresponding zero-mean unit-variance Gaussian process for various scale parameters.

1.4.2 Correlation function types

When the input dimension M is greater than one, multi-dimensional correlation function can be constructed from one-dimensional correlation families using one of the following constructions:

- *Ellipsoidal* correlation functions (Rasmussen and Williams, 2006), calculated as follows

$$R\left(\boldsymbol{x},\boldsymbol{x}';\boldsymbol{\theta}\right)=R\left(h\right)\ ,\ h=\left[\sum_{i=1}^{M}\left(\frac{x_i-x_i'}{\theta_i}\right)^2\right]^{0.5}. \tag{1.34}$$

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

$$R\left(\boldsymbol{x},\boldsymbol{x}';\boldsymbol{\theta}\right)=\prod_{i=1}^{M}R\left(x_i,x_i',\theta_i\right). \tag{1.35}$$

The function $R(\cdot)$ (resp, $R(\cdot,\cdot;\cdot)$) that appears on the right-hand side of Eq. (1.34) (resp. Eq. (1.35)) corresponds to one-dimensional correlation functions described in Section 1.4.1.

Note: When using one of the correlation families in Section 1.4.1 to build a multi-dimensional ellipsoidal correlation function, the term $\frac{|x-x'|}{\theta}$ inside the function is replaced with h.

1.4.3 Isotropic correlation functions

The multi-dimensional correlation functions given in Section 1.4.2 above correspond to the *anisotropic* case, in which there is a unique scale parameter for each input dimension. On the other hand, a multi-dimensional correlation function is called *isotropic* when a single scale parameter θ is associated with all the input dimensions. 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 given as

$$R\left(\boldsymbol{x},\boldsymbol{x}';\theta\right)=R\left(h\right);\ h=\frac{1}{\theta}\left[\sum_{i=1}^{M}\left(x_i-x_i'\right)^2\right]^{0.5}. \tag{1.36}$$

- Isotropic separable correlation function given as

$$R\left(\boldsymbol{x},\boldsymbol{x}';\theta\right)=\prod_{i=1}^{M}R\left(x_i,x_i';\theta\right). \tag{1.37}$$

1.4.4 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, for example, Eq. (1.6) to (1.9)). This inversion is well known to suffer from numerical instabilities, especially when the distances between the design points $\boldsymbol{x}_i$ are small and the responses are noise-free. To circumvent this limitation, a *nugget* $\boldsymbol{\nu}$ can be introduced. Nugget is a set of values that are added to the main diagonal of $\boldsymbol{R}$, such that

$$R_{ii} = 1 + \nu_i. \tag{1.38}$$

1.5 Estimation methods

To obtain a Kriging metamodel of Eq. (1.1), usually the hyperparameters $\boldsymbol{\theta}$ are unknown and thus must be estimated. The estimation is achieved by solving an optimization problem that differs depending on the selected estimation method, and whether the responses are noisy. The available estimation methods in UQ[PY]LAB are discussed in the following subsections.

1.5.1 Maximum-likelihood estimation

The idea behind the maximum-likelihood (ML) estimation method is to find the set of Kriging parameters $\boldsymbol{\beta}$, σ^2, $\boldsymbol{\theta}$, and if apply, σ_{n}^2, such that the likelihood of the observations $\mathcal{Y} = \left\{\mathcal{M}\left(\boldsymbol{x}^{(1)}\right), \ldots, \mathcal{M}\left(\boldsymbol{x}^{(N)}\right)\right\}^T$ is maximized. Since $\mathcal{Y}$ is assumed to follow a multivariate Gaussian distribution (recall basic Kriging assumptions), the likelihood function $\mathcal{L}\left(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta}; \mathcal{Y}\right)$ reads

$$\mathcal{L}\left(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta}; \mathcal{Y}\right) = \frac{(\det \boldsymbol{C})^{-1/2}}{(2\pi)^{N/2}} \exp\left[-\frac{1}{2}(\mathcal{Y} - \boldsymbol{F}\boldsymbol{\beta})^T \boldsymbol{C}^{-1} (\mathcal{Y} - \boldsymbol{F}\boldsymbol{\beta})\right] \tag{1.39}$$

where the covariance matrix $\boldsymbol{C}$ sums up the covariance matrix of the Gaussian process $\sigma^2\boldsymbol{R}$ and covariance matrix of the noise of the responses $\boldsymbol{\Sigma}_{\mathrm{n}}$ as follows

$$\boldsymbol{C} = \sigma^2 \boldsymbol{R} + \boldsymbol{\Sigma}_{\mathrm{n}}. \tag{1.40}$$

Depending on whether such noise is present as well as the nature of the noise, different optimization problems can be set up as described in the sequel.

1.5.1.1 Noise-free Kriging

For models with noise-free responses, the covariance matrix $\boldsymbol{C}$ reduces to $\sigma^2\boldsymbol{R}$, and the likelihood function can be written as

$$\mathcal{L}\left(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta}; \mathcal{Y}\right) = \frac{(\det \boldsymbol{R})^{-1/2}}{(2\pi\sigma^2)^{N/2}} \exp\left[-\frac{1}{2\sigma^2}(\mathcal{Y} - \boldsymbol{F}\boldsymbol{\beta})^T \boldsymbol{R}^{-1} (\mathcal{Y} - \boldsymbol{F}\boldsymbol{\beta})\right]. \tag{1.41}$$

By maximizing the quantity described in Eq. (1.41), the following analytical estimates of

$\boldsymbol{\beta}$ and $\boldsymbol{\sigma}^2$ that are strictly functions of $\boldsymbol{\theta}$ are obtained (for proof and more details, refer to Dubourg (2011); Santner et al. (2003))

$$\widehat{\boldsymbol{\beta}} = \boldsymbol{\beta}\left(\boldsymbol{\theta}\right) = \left(\boldsymbol{F}^T \boldsymbol{R}^{-1} \boldsymbol{F}\right)^{-1} \boldsymbol{F}^T \boldsymbol{R}^{-1} \mathcal{Y}, \tag{1.42}$$

$$\widehat{\sigma}^2 = \sigma^2(\boldsymbol{\theta}) = \frac{1}{N} \left(\mathcal{Y} - \boldsymbol{F}\boldsymbol{\beta}\right)^T \boldsymbol{R}^{-1} \left(\mathcal{Y} - \boldsymbol{F}\boldsymbol{\beta}\right). \tag{1.43}$$

The hyperparameters $\boldsymbol{\theta}$, in turn, are obtained from solving the following optimization problem[6]

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathcal{D}_{\boldsymbol{\theta}}}{\arg\min} \ \left[-\log \mathcal{L}\left(\boldsymbol{\theta}; \mathcal{Y}\right)\right]. \tag{1.44}$$

Based on Eqs. (1.41) to (1.43), the optimization problem of Eq. (1.44) can be written as

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathcal{D}_{\boldsymbol{\theta}}}{\arg\min} \ \frac{1}{2} \left[\log\left(\det \boldsymbol{R}\right) + N \log\left(2\pi\sigma^2\right) + N\right]. \tag{1.45}$$

1.5.1.2 Kriging with unknown homogeneous noise (homoscedastic)

In the case of noisy responses with an *unknown homogeneous (homoscedastic)* noise variance, the covariance matrix $\boldsymbol{C}$ reduces to $\sigma^2_{\text{total}}\widetilde{\boldsymbol{R}}$, where $\sigma^2_{\text{total}} = \sigma^2 + \sigma^2_n$, $\widetilde{\boldsymbol{R}} = (1-\tau)\,\boldsymbol{R} + \tau \boldsymbol{I}$, and $\tau = \frac{\sigma^2_n}{\sigma^2_{\text{total}}}$ (Section 1.2.2). The corresponding likelihood then reads

$$\mathcal{L}\left(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta}, \tau; \mathcal{Y}\right) = \frac{\left(\det \widetilde{\boldsymbol{R}}\right)^{-1/2}}{\left(2\pi\sigma^2_{\text{total}}\right)^{N/2}} \exp\left[-\frac{1}{2\sigma^2_{\text{total}}} (\mathcal{Y} - \boldsymbol{F}\boldsymbol{\beta})^T \widetilde{\boldsymbol{R}}^{-1} (\mathcal{Y} - \boldsymbol{F}\boldsymbol{\beta})\right], \tag{1.46}$$

where σ^2 and $\boldsymbol{R}$ in Eq. (1.41) have been replaced by σ^2_{total} and $\widetilde{\boldsymbol{R}}$, respectively.

Similarly, the estimates for $\boldsymbol{\beta}$ and σ^2_{total} are written as

$$\widehat{\boldsymbol{\beta}} = \boldsymbol{\beta}\left(\boldsymbol{\theta}\right) = \left(\boldsymbol{F}^T \widetilde{\boldsymbol{R}}^{-1} \boldsymbol{F}\right)^{-1} \boldsymbol{F}^T \widetilde{\boldsymbol{R}}^{-1} \mathcal{Y}, \tag{1.47}$$

$$\widehat{\sigma}^2_{\text{total}} = \sigma^2_{\text{total}}\left(\boldsymbol{\theta}\right) = \frac{1}{N} \left(\mathcal{Y} - \boldsymbol{F}\boldsymbol{\beta}\right)^T \widetilde{\boldsymbol{R}}^{-1} \left(\mathcal{Y} - \boldsymbol{F}\boldsymbol{\beta}\right). \tag{1.48}$$

Based on Eqs. (1.46) to (1.48), the optimization problem can be formulated as follows

$$\widehat{\boldsymbol{\theta}}, \widehat{\tau} = \underset{\boldsymbol{\theta} \in \mathcal{D}_{\boldsymbol{\theta}}, \tau \in (0,1)}{\arg\min} \ \frac{1}{2} \left[\log\left(\det \widetilde{\boldsymbol{R}}\right) + N \log\left(2\pi\widehat{\sigma}^2_{\text{total}}\right) + N\right]. \tag{1.49}$$

Notice that compared to Eq. (1.45), there is an additional parameter τ to be optimized and that it is bounded in $(0, 1)$.

[6] The logarithm of the likelihood is usually taken in the optimization to avoid numerical underflow problem.

1.5.1.3 Kriging with known noise (homo- and heteroschedastic)

Finally, in the case of noisy responses with *known* noise variance, the covariance matrix is given by Eq. (1.40).

An analytical form for the estimate of $\boldsymbol{\beta}$ as function of $\boldsymbol{\theta}$ is nevertheless available (Rasmussen and Williams, 2006) and reads

$$\widehat{\boldsymbol{\beta}} = \boldsymbol{\beta}\left(\boldsymbol{\theta}, \sigma^2, \sigma_{\mathrm{n}}^2\right) = \left(\boldsymbol{F}^T \boldsymbol{C}^{-1} \boldsymbol{F}\right)^{-1} \boldsymbol{F}^T \boldsymbol{C}^{-1} \mathcal{Y}. \tag{1.50}$$

There is, however, no analytical expression for the Gaussian process variance σ^2. and it must be simultaneously optimized with the rest of the hyperparameters $\boldsymbol{\theta}$. The optimization problem can be formulated as follows:

$$\widehat{\boldsymbol{\theta}}, \widehat{\sigma}^2 = \underset{\boldsymbol{\theta}\in\mathcal{D}_{\boldsymbol{\theta}}, \sigma^2\in\mathcal{D}_{\sigma^2}}{\arg\min} \frac{1}{2}\left[\log\left(\det \boldsymbol{C}\right) + N\left(\log(2\pi) + N + \left(\mathcal{Y} - \boldsymbol{F}\widehat{\boldsymbol{\beta}}\right)^T \boldsymbol{C}^{-1}\left(\mathcal{Y} - \boldsymbol{F}\widehat{\boldsymbol{\beta}}\right)\right]. \tag{1.51}$$

Note that, if the known noise is independent (*i.e.*, a vector $\boldsymbol{\sigma}_{\mathrm{n}}^2$), the noise covariance matrix $\boldsymbol{\Sigma}_{\mathrm{n}}$ is simplified to diag $\left(\boldsymbol{\sigma}_{\mathrm{n}}^2\right)$; whereas, if the known noise variance is homoscedastic (*i.e.*, a scalar σ_{n}^2), the noise covariance matrix $\boldsymbol{\Sigma}_{\mathrm{n}}$ is simplified to $\sigma_{\mathrm{n}}^2 \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix.

1.5.2 Cross-validation estimation

The general principle of a cross-validation (CV) method known as the K-fold cross-validation is to split the whole data set of observations $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right), i = 1, \ldots, N\right\}$ into K mutually exclusive and collectively exhaustive subsets $\{\mathcal{D}_k, k = 1, \ldots, K\}$ such that

$$\mathcal{D}_i \cap \mathcal{D}_j = \emptyset\ ,\ \forall (i,j) \in \{1, \ldots, K\}^2 \text{ and } \bigcup_{k=1}^{K} \mathcal{D}_k = \mathcal{D}. \tag{1.52}$$

The k-th subset of cross-validated predictions is obtained by estimating the model using all the subsets, except for the k-th one, and using the model to predict that specific k-th subset that was left apart. The leave-one-out (LOO) CV procedure corresponds to the special case that the number of subsets is equal to the number of observations (*i.e.*, $K = N$).

The cross-validation error of the k-th set is computed as

$$\begin{aligned} \epsilon_{\mathrm{CV},k} &= \sum_{\left(x^{(i)}, y^{(i)}\right)\in\mathcal{D}_k} \left(y^{(i)} - \mu_{\widehat{Y}}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta}, \sigma_n^2, \mathcal{D}\setminus\mathcal{D}_k\right)\right)^2 \\ &= \sum_{\left(x^{(i)}, y^{(i)}\right)\in\mathcal{D}_k} \left(y^{(i)} - \mu_{\widehat{Y},\setminus\mathcal{D}_k}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta}, \sigma_n^2\right)\right)^2 \end{aligned} \tag{1.53}$$

where $\mu_{\widehat{Y},\setminus\mathcal{D}_k}$ denotes the mean of the Kriging predictor (see, for instance, Eq. (1.6) for the noise-free prediction) on the k-th CV subset $\left\{\boldsymbol{x}^{(i)}, y^{(i)}\right\} \in \mathcal{D}_k$ conditioned on the other $K-1$

subsets. In the case of LOO CV, the number of elements in $\mathcal{D}_k$ is equal to one.

The overall cross-validation error then reads

$$\epsilon_{\text{CV}}\left(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta}, \sigma_{\text{n}}^2; \mathcal{Y}\right) = \frac{1}{N}\sum_{k=1}^{K} \epsilon_{\text{CV},k}. \tag{1.54}$$

The idea behind CV estimation method is to find the set of Kriging parameters $\boldsymbol{\beta}$, σ^2, $\boldsymbol{\theta}$, and, if apply, σ_{n}^2, that minimizes the cross-validation error. As in the case of maximum-likelihood estimation, depending on whether the noise in the response is present and the nature of the noise, different optimization problems based on the CV estimation can also be set up.

1.5.2.1 Noise-free Kriging

For models with noise-free responses, the optimization problem in a CV estimation reduces to (Santner et al., 2003; Bachoc, 2013)

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\mathcal{D}_{\boldsymbol{\theta}}}{\arg\min}\ \epsilon_{\text{CV}}\left(\boldsymbol{\theta}; \mathcal{Y}\right) \tag{1.55}$$

Notice that in this case the CV error of Eq. (1.54) can be written only as a function of $\boldsymbol{\theta}$ because of the analytical formula for β (Eq. (1.8)), the non-dependence of σ^2 from the predictor mean (Eq. (1.6)), and the absence of σ_{n}^2.

Using $\widehat{\boldsymbol{\theta}}$, the estimate of $\boldsymbol{\beta}$ is computed as in Eq. (1.8), while the estimate of σ^2 is computed using the following equation (Cressie, 1993; Bachoc, 2013)

$$\widehat{\sigma}^2 = \sigma^2(\widehat{\boldsymbol{\theta}}) = \frac{1}{N}\sum_{k=1}^{K}\sum_{i\in\mathcal{D}_k} \frac{\left(y^{(i)} - \mu_{\widehat{Y},\setminus\mathcal{D}_k}\left(\boldsymbol{x}^{(i)}; \widehat{\boldsymbol{\theta}}\right)\right)^2}{c^2_{\widehat{Y},\setminus\mathcal{D}_k}\left(\boldsymbol{x}^{(i)}; \widehat{\boldsymbol{\theta}}\right)} \tag{1.56}$$

where $c^2_{\widehat{Y},\setminus\mathcal{D}_k}$ denotes the normalized variance of the Kriging predictor on the k-th CV subset $\left\{\boldsymbol{x}^{(i)}, y^{(i)}\right\} \in \mathcal{D}_k$ conditional on all points of the other $K-1$ subsets. In other words,

$$c^2_{\widehat{Y},\setminus\mathcal{D}_k} = \frac{\sigma^2_{\widehat{Y},\setminus\mathcal{D}_k}}{\sigma^2} \tag{1.57}$$

where $\sigma^2_{\widehat{Y},\setminus\mathcal{D}_k}$ denotes the variance of the Kriging predictor (Eq. (1.7)) on the same k-th CV subset conditional on all points of the other $K-1$ subsets.

1.5.2.2 Kriging with unknown homogeneous noise (homoscedastic)

In the case of noisy responses with *unknown homogeneous* noise variance, an additional parameter τ is to be optimized along with the hyperparameters $\boldsymbol{\theta}$ as follows

$$\widehat{\boldsymbol{\theta}}, \widehat{\tau} = \underset{\boldsymbol{\theta}\in\mathcal{D}_{\boldsymbol{\theta}}, \tau\in(0,1)}{\arg\min}\ E_{\text{CV}}\left(\boldsymbol{\theta}, \tau; \mathcal{Y}\right). \tag{1.58}$$

in which $\mu_{\widehat{Y},\setminus\mathcal{D}_k}\left(\boldsymbol{x}^{(i)};\boldsymbol{\theta},\tau^2\right)$ given in Eq. (1.25) is used as the Kriging predictor.

Using $\widehat{\boldsymbol{\theta}}$ and $\widehat{\tau}$, the estimate of $\boldsymbol{\beta}$ is computed as in Eq. (1.47), while the estimate of σ^2_{total} is computed using the following equation

$$\widehat{\sigma}^2_{\text{total}} = \sigma^2_{\text{total}}\left(\widehat{\boldsymbol{\theta}},\widehat{\tau}\right) = \frac{1}{N}\sum_{k=1}^{K}\sum_{i\in\mathcal{D}_k}\frac{\left(y^{(i)}-\mu_{\widehat{Y},\setminus\mathcal{D}_k}\left(\boldsymbol{x}^{(i)};\widehat{\boldsymbol{\theta}},\widehat{\tau}\right)\right)^2}{c^2_{\widehat{Y},\setminus\mathcal{D}_k}\left(\boldsymbol{x}^{(i)};\widehat{\boldsymbol{\theta}},\widehat{\tau}\right)} \tag{1.59}$$

where $c^2_{\widehat{Y},\setminus\mathcal{D}_k}$ above is analogous to the one appeared in Eq. (1.56) but based on Eq. (1.26), instead of Eq. (1.7).

1.5.2.3 Kriging with known noise (homo- and heteroschedastic)

Similarly to the case of the ML estimation (Section 1.5.1.3), models with known responses noise variance also require that the Gaussian process variance σ^2 is optimized with rest of the hyperparameters $\boldsymbol{\theta}$. The optimization problem now reads

$$\widehat{\boldsymbol{\theta}},\widehat{\sigma}^2 = \underset{\boldsymbol{\theta}\in\mathcal{D}_{\boldsymbol{\theta}},\sigma^2\in\mathcal{D}^2_\sigma}{\arg\min}\ \epsilon_{\text{CV}}\left(\boldsymbol{\theta},\sigma^2;\mathcal{Y},\boldsymbol{\Sigma}_{\text{n}}\right) \tag{1.60}$$

where $\boldsymbol{\Sigma}_{\text{n}}$ is the known noise covariance matrix. Here, the Kriging predictor used in the computation of the CV error uses the formulation of Eq. (1.16).

If the noise is independent, the noise covariance matrix $\boldsymbol{\Sigma}_{\text{n}}$ is simplified to $\text{diag}\left(\boldsymbol{\sigma}^2_{\text{n}}\right)$, where $\boldsymbol{\sigma}^2_{\text{n}}$ is the vector of noise variances; while if the noise is homoscedastic (*i.e.*, a scalar σ^2_{n}), the noise covariance matrix $\boldsymbol{\Sigma}_{\text{n}}$ is simplified to $\sigma^2_{\text{n}}\boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix.

1.6 Optimization methods

To solve the optimization problems described in Section 1.5, there are trade-offs between choosing some local (usually gradient-based) or choosing global (*e.g.*, evolutionary algorithms) methods. Local methods tend to converge faster and require fewer objective function evaluations, but may perform poorly due to the existence of flat regions and multiple local minimas, especially for increasing input dimension. Alternatively, the so-called *hybrid* methods combine the features of both methods.

The currently available optimization methods are briefly discussed below:

- **Interior point with L-BFGS Hessian approximation**
  An interior point gradient-based method is used to approximate the Hessian matrix using a limited-memory variant of the quasi-Newton Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) method (Byrd et al., 1999; Nocedal, 1980).
- **Genetic Algorithm (GA)**
  GA 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, in

which the final solution of the genetic algorithm is used as a starting point of the gradient-based method previously mentioned.

- **Covariance matrix adaptation–evolution strategy (CMA-ES)**
  CMA-ES 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. Similar to GA, a hybrid approach can also be used with CMAES.

An example of the objective function landscape for a one-dimensional problem is given in Figure 7. The original model is $\mathcal{M}(x) = \left(1+25x^2\right)^{-1}$ (*i.e.*, the Runge function). The experimental design consists of 8 points as shown in Figure 7(a). The evaluations of objective functions as a function of θ based on the ML and CV estimations (both are to be minimized) are shown in Figures 7(c) and 7(b), respectively.

1.7 *A posteriori* error estimation

After the Kriging metamodel is set up, its predictive accuracy on new set of data can be assessed either by using leave-one-out (LOO) cross-validation error or *validation error*.

1.7.1 Leave-one-out cross-validation error

The leave-one-out (LOO) cross-validation (CV) error is calculated on the initial experimental design $\mathcal{X}$, and its corresponding responses $\mathcal{Y} = \mathcal{M}(\mathcal{X})$ as follows

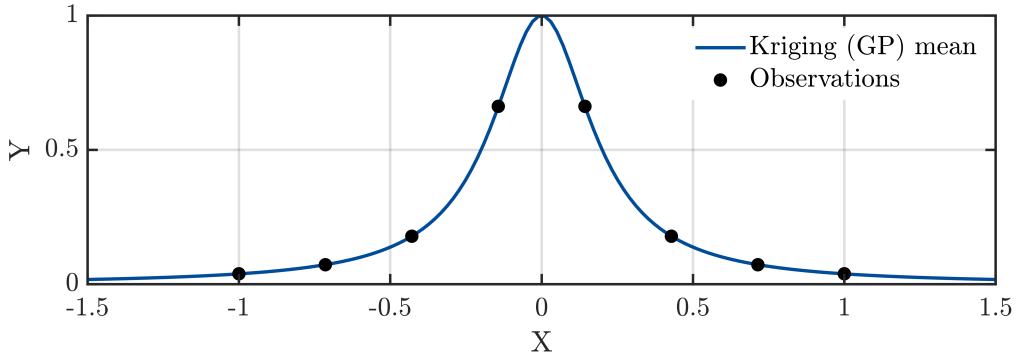
$$\epsilon_{LOO} = \frac{1}{N}\left[\frac{\sum_{i=1}^{N}\left(\mathcal{M}(\boldsymbol{x}_i) - \mu_{\widehat{Y},(-i)}(\boldsymbol{x}_i)\right)^2}{\mathrm{Var}\left[\mathcal{Y}\right]}\right] \tag{1.61}$$

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

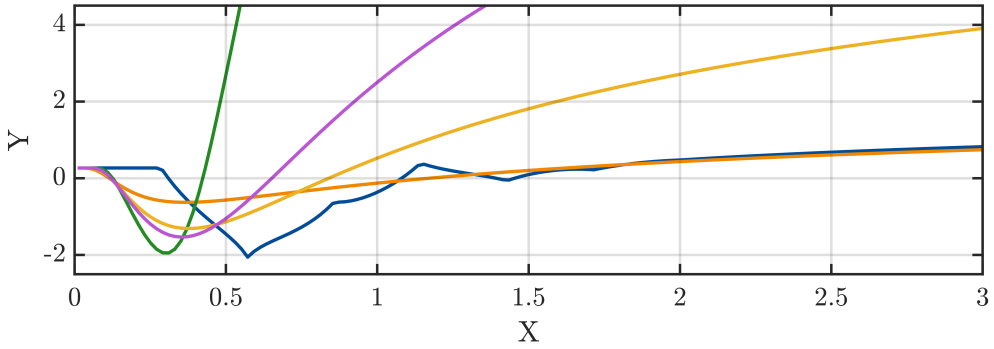
1.7.2 Validation error

The validation error is calculated as the relative generalization error on an *independent* set of data $\mathcal{D}_{\text{val}} = \left\{\left(\boldsymbol{x}_{\text{val}}^{(i)}, y_{\text{val}}^{(i)}\right), i = 1, \ldots, N_{\text{val}}\right\}$ as follows

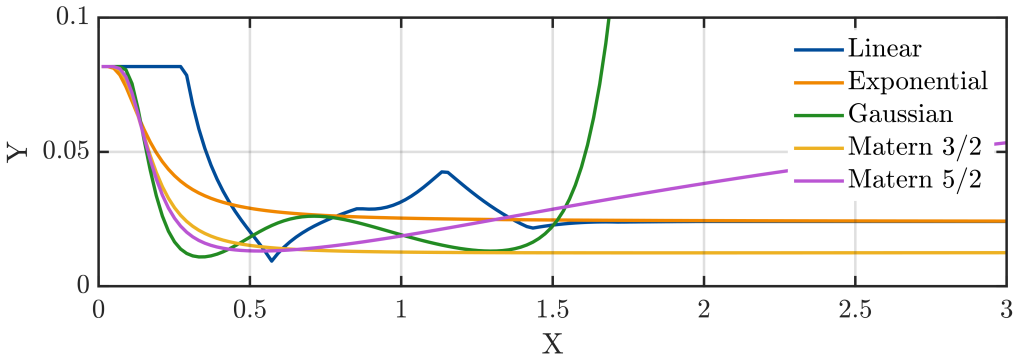
$$\epsilon_{\text{val}} = \frac{N_{\text{val}}-1}{N_{\text{val}}}\left[\frac{\sum\limits_{i=1}^{N_{\text{val}}}\left(\mathcal{M}(\boldsymbol{x}_{\text{val}}^{(i)} - \mathcal{M}^K\left(\boldsymbol{x}_{\text{val}}^{(i)}\right)\right)^2}{\sum\limits_{i=1}^{N_{\text{val}}}\left(\mathcal{M}\left(\boldsymbol{x}_{\text{val}}^{(i)}\right) - \hat{\mu}_{Y_{\text{val}}}\right)^2}\right] \tag{1.62}$$



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



(b) Maximum likelihood (ML) estimation objective function landscape.



(c) Cross-validation (CV) estimation objective function landscape.

Figure 7: Examples of one-dimensional objective function landscapes as functions of scale parameter θ for ML and CV estimation methods using various Gaussian process correlation families.

where $\hat{\mu}_{Y_{\text{val}}} = \frac{1}{N_{\text{val}}} \sum_{i=1}^{N_{\text{val}}} \mathcal{M}\left(\boldsymbol{x}_{\text{val}}^{(i)}\right)$ is the sample mean of the validation set responses. This error measure is useful to compare the performance of different metamodels evaluated on the same validation set.

Chapter 2

Usage

In this chapter, a reference problem is set up to showcase how each of the Kriging ingredients described in Section 1.2.3 can be used in UQ[PY]LAB.

2.1 Reference problem

In this manual, Kriging is used to build a surrogate of a model given a set of observed inputs 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 setup

The Kriging module creates a MODEL object. The basic options common to any Kriging metamodel read

```
MetaOpts = {
    "Type": "Metamodel",
    "MetaType": "Kriging"
}
```

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

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

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

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\mathcal{D}_{\boldsymbol{\theta}}}{\arg\min}\ J(\boldsymbol{\theta}) \tag{2.3}$$

In practice, the objective function $J(\boldsymbol{\theta})$ differs depending on the choice of estimation method (*i.e.*, maximum-likelihood or cross-validation). The main ingredients that need to be set up

to obtain a Kriging metamodel are the following:

- An experimental design, $\mathcal{X}$, and the corresponding model responses, $\mathcal{Y}$. If they are not available, they can be generated by defining a full computational (*true*) model and a probabilistic input model[1].
- A trend specification, *i.e.*, selection of the type of trend component functions $f_j(\boldsymbol{x})$, $j = 1, \ldots, P$.
- An appropriate correlation function $R(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta})$.
- A hyperparameter estimation method that defines the objective function $J(\boldsymbol{\theta})$ in Eq. (2.3); this choice also defines the approach to estimate the Gaussian process variance σ^2.
- An optimization method to solve the problem in Eq. (2.3).

Note that the minimal amount of information that a user needs to supply is the experimental design and the corresponding model responses. The user can either select and tune each of the ingredients, or leave them to their default values.

2.3 Kriging metamodel calculation: noise-free case

Below a minimal configuration example to create a Kriging metamodel in UQ[PY]LAB is given as was discussed in Section 2.2.

```
from uqpylab import sessions
import numpy as np

# Start the session
mySession = sessions.cloud(host=UQCloudhost,
token=UQCloudtoken)
# (Optional) Get a convenient handle to the command line interface
uq = mySession.cli
# Reset the session
mySession.reset()

uq.rng(100,'twister')   # For reproducible results

# Create experimental design
X = np.arange(0, 15, 2)
Y = X * np.sin(X)

# Define a Kriging metamodel
MetaOpts = {
    'Type': 'Metamodel',
    'MetaType': 'Kriging',
    'ExpDesign':{
        'Sampling': 'User',
        'X': X.tolist(),
        'Y': Y.tolist()
```

[1]Note that this step belongs to the general context of metamodeling, *i.e.*, it is not specific to Kriging (see, for instance, UQ[PY]LAB User Manual – Polynomial Chaos Expansions)

```
    }
}

# Create the Kriging metamodel
myKriging = uq.createModel(MetaOpts)

# Terminate the remote UQCloud session
mySession.quit()
```

Once the metamodel is created, a report of the Kriging results can be printed on screen by:

```
uq.print(myKriging)
```

which then shows:

```
    %--------------- Kriging metamodel ---------------%
    Object Name:            Model 1
    Input Dimension:        1
    Output Dimension:       1

    Experimental Design
    Sampling:               User
    X size:                [8x1]
    Y size:                [8x1]

    Trend
    Type:                   ordinary
    Degree:                 0
    Beta:                   [ 31.66776  ]

    Gaussian Process (GP)
    Corr. type:             ellipsoidal
    Corr. isotropy:         anisotropic
    Corr. family:           matern-5_2
    sigma^2:                1.18220e+05
    Estimation method:      Cross-validation

    Hyperparameters
    theta:                  [ 2.90596 ]
    Optim. method:          Hybrid Genetic Alg.

    GP Regression
    Mode:                   interpolation

    Error estimates
    Leave-one-out:          5.55516e-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)
```

The figure produced by `uq.display` is shown in Figure 8.

Note: Note that `uq.display` for Kriging MODEL objects can only be used for model one- and two-dimensional inputs.

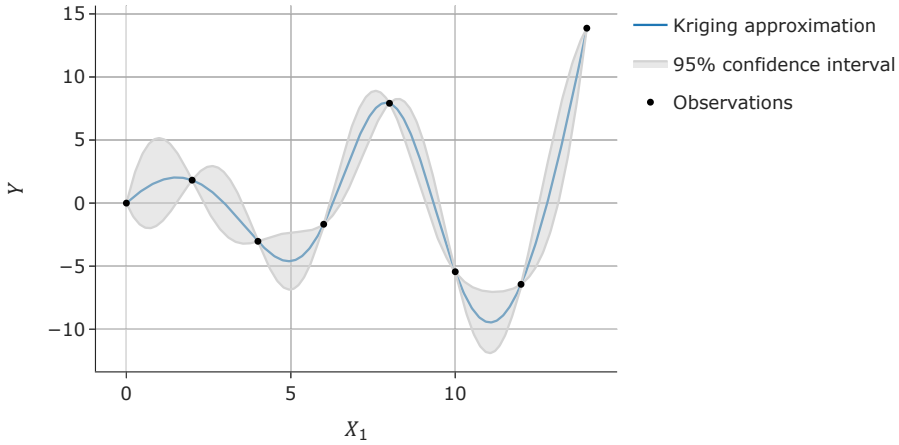


Figure 8: The output of `uq.display` of a one-dimensional Kriging MODEL object.

2.3.1 Accessing the results

The Kriging MODEL object `myKriging` created in Section 2.3 uses all the default configuration options (as listed in the output of `uq.print(myKriging)`). This object can be directly accessed to obtain various results and information of the metamodel.

2.3.1.1 Kriging parameters

The values of the basic parameters of the Kriging metamodel, namely $\boldsymbol{\theta}$, σ^2, and $\boldsymbol{\beta}$ are contained in the dictionary `myKriging['Kriging']`:

```
{
    'beta': 31.66474977533767,
    'sigmaSQ': 118181.5463011093,
    'theta': 2.905772037286146
}
```

2.3.1.2 Model evaluations

The experimental design and the corresponding model responses are stored in the `myKriging['ExpDesign']` dictionary:

```
{
    'Sampling': 'User',
    'X': [0, 2, 4, 6, 8, 10, 12, 14],
    'Y': [0, 1.8185948536513634, -3.027209981231713, -1.6764929891935552,
    7.914865972987054, -5.440211108893697, -6.43887501600522,
    13.868502979728184],
    'NSamples': 8,
```

```
    'U': [-1.4288690166235207, -1.0206207261596576, -0.6123724356957946,
    -0.20412414523193154, 0.20412414523193154, 0.6123724356957946,
    1.0206207261596576, 1.4288690166235207]
}
```

The fields contain the following information:

- `ExpDesign['Sampling']`: the source of the experimental design
- `ExpDesign['NSamples']`: the size of the experimental design
- `ExpDesign['X']`: the experimental design $\mathcal{X}$
- `ExpDesign['Y']`: the corresponding full model responses, $\mathcal{Y} = \mathcal{M}(\mathcal{X})$
- `ExpDesign['U']`: the scaled experimental design (refer to Section 2.10 for details)

2.3.1.3 *A posteriori* error estimates

The Leave-One-Out (LOO) cross-validation error of the metamodel is stored in `myKriging['Error']`:

```
{
    'LOO': 0.5555163005242105
}
```

This error is calculated according tofollowing Eq. (1.61). Refer to Section 1.7.1 for details.

2.4 Kriging metamodel setup

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 the options regarding the generation or specification of the experimental design. The way to use each option is discussed in Section 2.4.1 and list of all available options can be found in Table 5.
- `MetaOpts['Trend']` contains the options regarding the term $\sum_{j=1}^{P} \beta_j(\boldsymbol{\theta}) f_j(\boldsymbol{x})$ in Eq. (2.2), see Section 1.3 for a theoretical introduction. The way to use each option is discussed in Section 2.4.2 and the list of all available options can be found in Table 6.
- `MetaOpts['Corr']` contains the options regarding the correlation function that is used in order to compute $\boldsymbol{R}$ in Eq. (2.2), see Section 1.4 for a theoretical introduction. The way to use each option is discussed in Section 2.4.3 and the list of all available options can be found in Table 8.
- `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 $\sigma^2(\boldsymbol{\theta})$ as well as $J(\boldsymbol{\theta})$. See Section 1.5 for a theoretical introduction of each estimation method.

- `MetaOpts['Optim']` contains the options related to the method for solving the optimization problem in Eq. (2.3). See Section 1.6 for a brief description of the available optimization methods. The way to use each option is discussed in Section 2.4.5 and the list of all available options can be found in Table 10.

2.4.1 Specification and generation of experimental design

An experimental design for constructing a Kriging metamodel can either be specified using existing data or generated from a MODEL and an INPUT objects.

2.4.1.1 Using existing data

If data is stored in the variables `X`, `Y`:

```
MetaOpts['ExpDesign'] = {
    'Sampling': 'User',
    'X': X.tolist(),
    'Y': Y.tolist()
}
```

The input and output dimensions of the problem M, N_{out} are automatically inferred from the number of columns of `X` and `Y`, respectively.

2.4.1.2 Using INPUT and MODEL objects

In order to use an INPUT object, it first needs to be created. For the reference problem in Eq. (2.1) this reads:

```
InputOpts = {
    'Marginals': [{
        'Type': 'Uniform',
        'Parameters': [0, 15]
    }]
}
myInput = uq.createInput(InputOpts)
```

The input and output dimensions of the problem are automatically inferred from the configuration of the INPUT object `myInput` and MODEL object `myModel` . For details about the configuration options available for INPUT and MODEL objects, please refer to the UQ[PY]LAB User Manual – the INPUT module and the UQ[PY]LAB User Manual – the MODEL module, respectively.

Then, a MODEL object is created as follows:

```
ModelOpts = {
    'Type' : 'Model',
    'mString' : 'X.*sin(X)'
}
myModel = uq.createModel(ModelOpts)
```

Now, the INPUT and MODEL objects are specified into the Kriging metamodel configuration:

```
MetaOpts['Input'] = myInput
MetaOpts['FullModel'] = myModel
```

Finally, the size of the experimental design is specified, *e.g.*, for $N = 8$ sample points:

```
MetaOpts['ExpDesign']['NSamples'] = 8
```

Note that, by default, Latin Hypercube sampling (LHS) 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 an arbitrary degree (Eq. (1.27)), set the following option:

  ```
  MetaOpts['Trend'] = {
      'Type': 'polynomial',
      'Degree': q
  }
  ```

  where `q` is an integer equal to the polynomial degree.

- For a constant trend with fixed value (*i.e.*, *simple* Kriging), set the following option:

  ```
  MetaOpts['Trend'] = {
      'Type': 'simple',
      'Degree': V
  }
  ```

  where `V` is a real number.

A summary of the available trend options in UQ[PY]LAB 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 trend estimation)	Not required	Required
`'ordinary'`	β_0	Not required	Not required
`'linear'`	$\beta_0 + \sum_{i=1}^{M} \beta_i x_i$	Not required	Not required

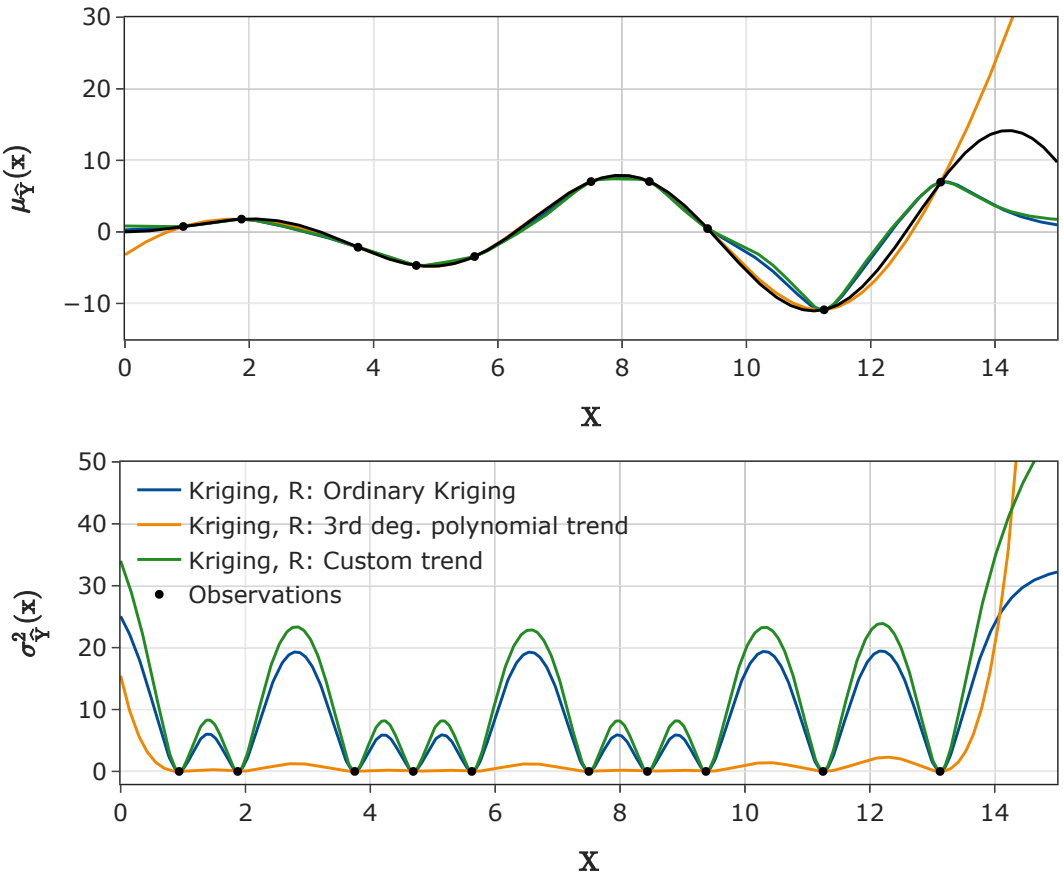


Figure 9: The output of the example script `uq_Example_Kriging_03_TrendTypes`. The mean and variance of various Kriging predictors are plotted, using different trend configurations. In the top figure, black line indicates the true model prediction.

`'quadratic'`	$\beta_0 + \sum_{i=1}^{M} \beta_i x_i + \sum_{i=1}^{M} \sum_{j=1}^{M} \beta_{ij} x_i x_j$	Not required	Not required
`'polynomial'`	see Eq. (1.27)	Required	Not required
`'custom'`	$\sum_{i=1}^{P} \beta_i f_i(\boldsymbol{x})$	Not required	Required

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

Note: The field `MetaOpts['Trend']['Type']` determines the trend type of a Kriging metamodel. If the user does not define a trend type, the default `MetaOpts['Trend']['Type'] = 'ordinary'` (*i.e.*, ordinary Kriging) is used.

2.4.2.2 Advanced options

A user can define arbitrary custom trends via advanced options. First, the trend type is specified as follows:

```
MetaOpts['Trend']['Type'] = 'custom'
```

Then the trend can be defined as a string. If, for example, the trend is a sine function $\sin(x)$, then the option is set as follows:

```
MetaOpts['Trend']['CustomF'] = 'sin'
```

UQCloud uses the MATLAB engine to interpret the function strings. Therefore, the function can be any of the built-in MATLAB functions. The list of supported elementary math functions can be found here.

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

- If `MetaOpts['Trend']['Type']` is `'simple'`, then P equals to 1.
- If `MetaOpts['Trend']['Type']` is either `'ordinary'`, `'linear'`, or `'quadratic'`, then P equals to 1, $M+1$, $\frac{(M+2)(M+1)}{2}$, respectively.
- If `MetaOpts['Trend']['Type']`=`'polynomial'`, then depending on the polynomial degree q, defined in `MetaOpts['Trend']['Degree']`, P equals to $\binom{M+q}{q}$.
- If `MetaOpts['Trend']['Type']` is `'custom'`, then P depends on how `MetaOpts['Trend']['CustomF']` is specified:
  - If `MetaOpts['Trend']['CustomF']` is a string or float, then P equals to 1.
  - If `MetaOpts['Trend']['CustomF']` is a list of strings, then P equals the length of the list. In other words, each element of the list corresponds to a single column of $\boldsymbol{F}$.

2.4.3 Correlation function

The key ingredients for specifying a correlation function is done through the standard options, while urther flexibility such as using a custom correlation function can be accessed through the advanced options.

2.4.3.1 Standard options

The three key ingredients for specifying a correlation function are:

- **Type** that specifies the type of the autocorrelation function as described in Section 1.4.2. By default, the type is set to `'ellipsoidal'`. To use instead a separable correlation function, the following option is set:

```
MetaOpts['Corr'] = {
'Type': 'separable'
}
```

Note: If the user does not specify the correlation type, it is, by default, set to be `'ellipsoidal'`.

- **Family** that specifies the one-dimensional correlation function family $R(\cdot)$ of the Gaussian process as described in Section 1.4. 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 normalized Euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{x}'$ (see Eq. (1.34)). Each coordinate difference between the rows of $\boldsymbol{x}$ and $\boldsymbol{x}'$ is normalized 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.

  To use a built-in correlation family, its name must be set, see Table 8 in Section 3.1 for the name of each built-in correlation family and Section 1.4.1 for a brief description of each family. For example, to use the exponential family, the following option is set:

```
MetaOpts['Corr']['Family'] = 'exponential'
```

Note: By default, the correlation family is set to be `'matern-5_2'`.

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

```
MetaOpts['Corr']['Isotropic'] = True
```

Note: By default, the correlation function is considered anisotropic, that is, `MetaOpts['Corr']['Isotropic'] = False`.

In Figure 10, the mean and variance of various Kriging predictors having different correlation families (as in the example script `uq_Example_Kriging_01_1D`) are plotted.

2.4.3.2 Advanced options

Through the advanced options, users can specify custom correlation function families, custom routines to compute the whole correlation matrix $\boldsymbol{R}$, as well as a nugget term to stabilize the inversion of $\boldsymbol{R}$.

Note: In the current version of UQ[PY]LAB there is limited support for advanced correlation functions options.

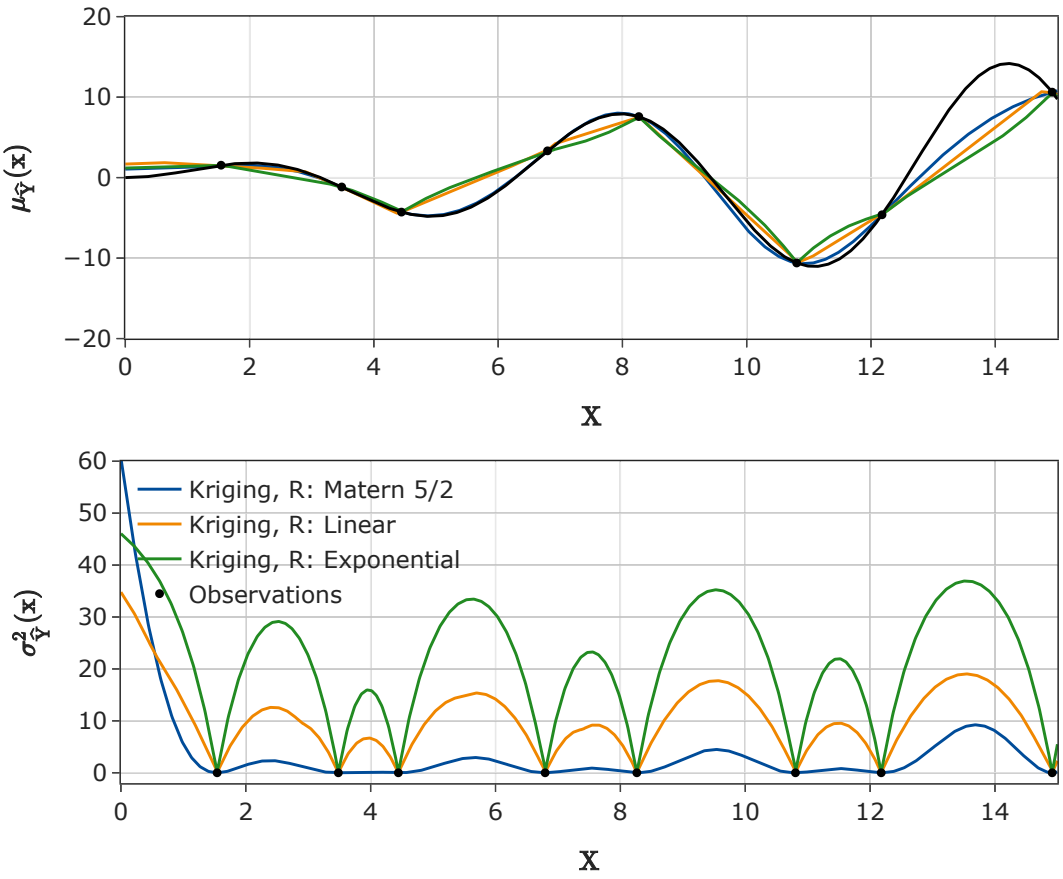


Figure 10: The output of the example script `uq_Example_Kriging_01_1D`. The mean and variance of various Kriging predictors are plotted, using different correlation families. In the top figure, black line indicates the true model prediction.

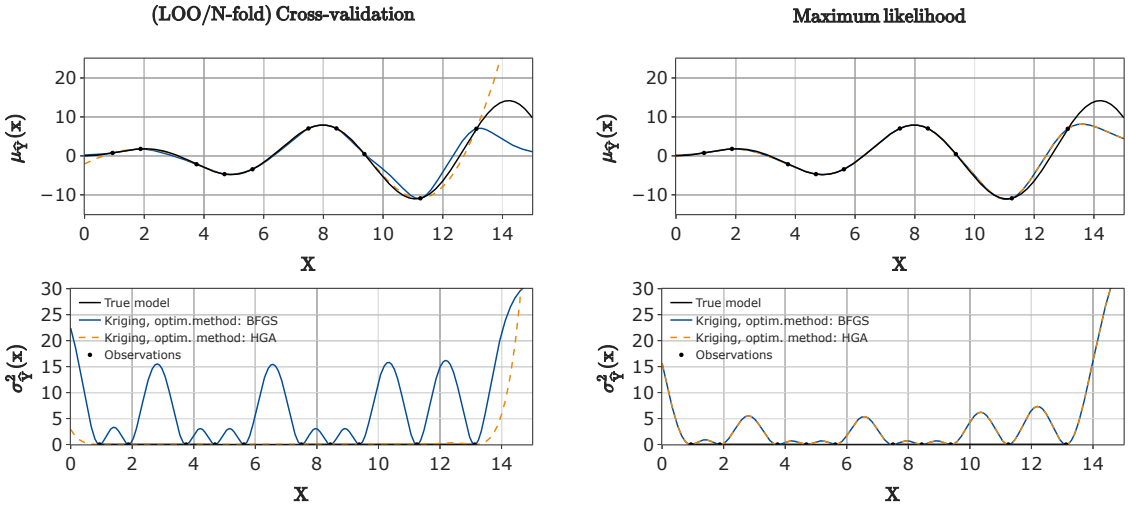


Figure 11: The output of the example script `uq_Example_Kriging_02_VariousMethods`. The mean and variance of various Kriging predictors are plotted, using two different optimization methods. On the left-hand side, the maximum likelihood estimation method is used and on the right-hand side the LOO cross-validation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

- **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, say $\nu = 10^{-4}$, the following option is set:

```
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 $\boldsymbol{R}$, that is, $R_{ii} = 1 + \nu_i,\ \{i = 1, \ldots, N\}$.

> **Note:** By default, the nugget value in UQ[PY]LAB is set to $\nu = 10^{-10}$.

2.4.4 Estimation methods

The estimation method determines the objective function that is minimized in Eq. (2.3) to estimate the hyperparameters. If required, this choice also determines how the (constant) Gaussian process variance σ^2 is calculated. For details, see Section 1.5.

The maximum likelihood and cross-validation methods are available for estimating the Kriging parameters and it suffices to select either `'ML'` or `'CV'` in the field `MetaOpts['EstimMethod']`, for the maximum-likelihood and cross-validation estimation methods, respectively.

An example that illustrates different Kriging predictors created using different estimation is given in the script `uq_Example_Kriging_02_VariousMethods` (Figure 11).

2.4.4.1 Advanced Options

If the cross-validation (CV) method is selected as the estimation method, then by default, UQ[PY]LAB uses the N-fold CV (or equivalently, the *leave-one-out*) approach.

Other K-fold CV approaches ($K \leq N$) can be used by changing the number of sample points left out from the estimation via the `MetaOpts.CV.LeaveKOut` option as follows:

```
MetaOpts['CV'] = {
    'LeaveKOut': k
}
```

where k is an integer. The number of folds (subsets) in K-fold CV is computed as follows

$$K = \lceil N/k \rceil \tag{2.4}$$

As an example, for $N = 100$ and $k = 3$, the number of folds in K-fold CV equals to 34.

Note: By default, the estimation method is set to be the N-fold (leave-one-out) cross-validation method, that is, `MetaOpts['EstimMethod'] = 'CV'` and `MetaOpts['CV']['LeaveKOut'] = 1`.

Note: If K-fold CV is used, UQ[PY]LAB randomly permutes the experimental design before creating the folds. Furthermore, the `MetaOpts['CV']['LeaveKOut']` must be smaller than the size of the experimental design N, such that at least two folds are available for cross-validation. Otherwise UQ[PY]LAB will throw an error.

2.4.5 Optimization methods

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

- **Gradient-based methods**

  Currently, only BFGS method is available as a local gradient-based method. To use this method, the following option is set:

  ```
  MetaOpts['Optim'] = {
  'Method': 'BFGS'
  }
  ```

  As in any gradient-based methods, an initial value of the hyperparameters (denoted here as $\boldsymbol{\theta}_0$) is required. By default, $\theta_0 = 1$ 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
  ```

Furthermore, different initial value per dimension can also be specified using an $M \times 1$ vector instead of a scalar.

Note: Note that, by default, the hyperparameters are defined in to the scaled (auxiliary) space $\mathcal{U}$ as described in Section 2.10.

- **Global and hybrid methods**

  The Genetic Algorithm (GA), the covariance matrix adaptation–evolution strategy (CMA-ES), and their hybrid counterparts (HGA and HCMAES, respectively) are available as optimization methods in UQ[PY]LAB. To use one of these methods, *e.g.*, GA, the following option is set:

  ```
  MetaOpts['Optim']['Method'] = 'GA'
  ```

  As in any global methods, the bounds of the optimization variable(s) needs to be set. By default, the bounds (lower, upper) $[0.001, 10]^T$ are used. To specify different bounds, for example, $[0.01, 1]^T$, the following options is set:

  ```
  MetaOpts['Optim']['Bounds'] = [0.01, 1]
  ```

  Note: Note that, by default, the hyperparameters are defined in to the scaled (auxiliary) space $\mathcal{U}$ as described in Section 2.10.

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

  ```
  MetaOpts['Optim']['GA']= {
  'nStall': 20
  }
  ```

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

- The number of iterations or generations (its actual meaning depends on the optimization method), for example:

  ```
  MetaOpts['Optim']['MaxIter'] = 100
  ```

- The convergence tolerance, for example:

  ```
  MetaOpts['Optim']['Tol'] = 1e-5
  ```

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

  ```
  MetaOpts['Optim']['Display'] = 'final'
  ```

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

Note: If the user does not specify any method, the Hybrid Genetic Algorithm (HGA) is used by default.

If no optimization of the hyperparameters is required, due to, for example, a prescribed value `theta_user` is used, the following options are set:

```
MetaOpts['Optim']['Method'] = 'none'
MetaOpts['Optim']['InitialValue'] = theta_user
```

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).

2.5 Kriging metamodels with noise (regression)

To demonstrate the Kriging metamodel with noisy responses, we consider once more the reference problem described in Section 2.1 and add to the observed response an additive noise

$$y = \mathcal{M}(x) + \varepsilon \tag{2.5}$$

The following subsections demonstrate how to create several Kriging metamodels in UQ[PY]LAB for different cases of noisy responses (see Section 1.2.2).

2.5.1 Unknown homoscedastic noise

Assuming that the noise is *homoscedastic* (*i.e.* its variance is constant), UQ[PY]LAB can estimate the unknown noise variance from the observed data during the calculation of the Kriging model. To estimate the noise, the following option is set:

```
MetaOpts['Regression'] = {
    'SigmaNSQ': 'auto'
}
```

The example below demonstrates how to create a Kriging model with an unknown homoscedastic noise in the response. First, an illustrative noisy data set is created:

```
import numpy as np
import math
# Create experimental design with noisy responses
uq.rng(100,'twister')   # For reproducible results

X = np.linspace(0,14,100)
noise_var = 3.0
Y = X * np.sin(X) + math.sqrt(noise_var) * np.random.randn(X.size)
```

Then, a Kriging metamodel as in Section 2.3 is specified, now with the additional option:

```
# Define a Kriging metamodel
MetaOpts = {
    'Type': 'Metamodel',
    'MetaType': 'Kriging',
    'ExpDesign': {
        'Sampling': 'User',
        'X': X.tolist(),
        'Y': Y.tolist()
    },
    'Regression': {   # Estimate the noise
        'SigmaNSQ': 'auto'
    }
}
```

Finally, the metamodel is created:

```
myKriging = uq.createModel(MetaOpts)
```

The report produced by `uq.print` now displays an additional information about the noise estimation process. Notice that the estimate noise variance is now reported.

```
    %---------------- Kriging metamodel ----------------%
    Object Name:            Model 1
    Input Dimension:        1
    Output Dimension:       1

    Experimental Design
    Sampling:               User
    X size:                 [100x1]
    Y size:                 [100x1]

    Trend
    Type:                   ordinary
    Degree:                 0
    Beta:                   [ 4.18461 ]

    Gaussian Process (GP)
    Corr. type:             ellipsoidal
    Corr. isotropy:         anisotropic
    Corr. family:           matern-5_2
    sigma^2:                6.43875e+01
    Estimation method:      Cross-validation

    Hyperparameters
    theta:                  [ 0.82604 ]
    Optim. method:          Hybrid Genetic Alg.

    GP Regression
    Mode:                   regression
    Est. noise:             true
    sigmaN^2:               2.89456e+00

    Error estimates
    Leave-one-out:          9.03089e-02
    %-----------------------------------------------------%
```

A visual representation of the Kriging metamodel can be obtained via the `uq.display` function whose result is shown in Figure 12.

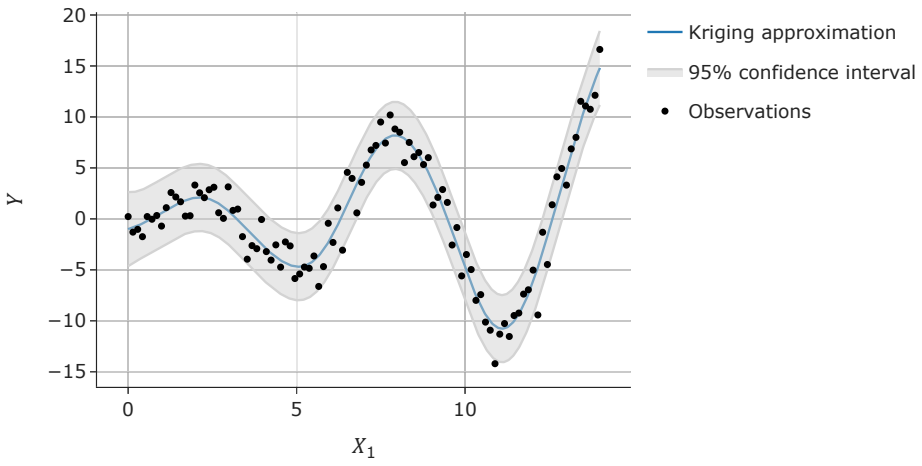


Figure 12: The output of `uq.display` of a Kriging MODEL object having a one-dimensional input, with noisy responses.

2.5.2 Known noise

If the noise variance is known *a priori*, it can be used directly in the regression model without being estimated. To specify a known noise variance, the following options are set:

```
MetaOpts['Regression'] = {
'SigmaNSQ': constSigmaNSQ
}
```

where `constSigmaNSQ` is the known noise variance. The specification `constSigmaNSQ` depends on the nature of the noise as follows:

- *Homoscedastic* noise: `constSigmaNSQ` is a scalar. This is the case in which the noise in the response is constant everywhere.
- *Independent heteroscedatic* noise: `constSigmaNSQ` is a $N \times 1$ array. This is the case in which the noises are independent, but may differ at each observation point.
- *Correlated heteroscedastic* noise: `constSigmaNSQ` is a $N \times N$ matrix. This is the case in which the noises are neither constant nor independent. In this case, `constSigmaNSQ` is the covariance matrix of the noise.

The case of known homoscedastic noise is illustrated using the previous example with smaller set of data as follows:

```
# Create a hypothetical experimental design with noisy responses
uq.rng(100,'twister')   # For reproducible results

X = np.linspace(0,14,50)
noise_var = 3.0
Y = X * np.sin(X) + math.sqrt(noise_var) * np.random.randn(X.size)
```

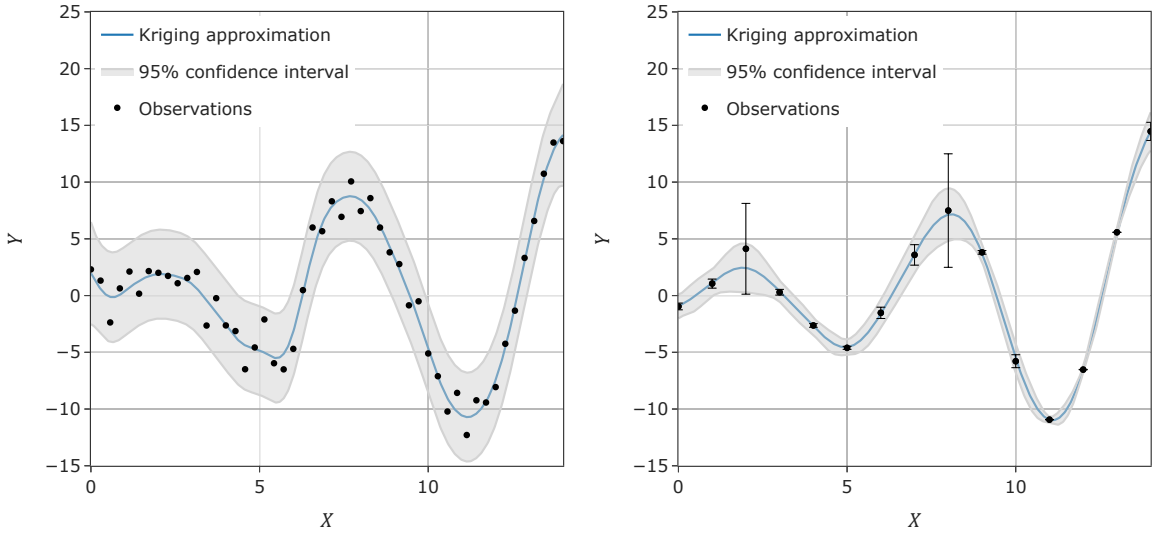
```
# Define a Kriging metamodel
MetaOpts = {
    'Type': 'Metamodel',
    'MetaType': 'Kriging',
    'ExpDesign': {
        'Sampling': 'User',
        'X': X.tolist(),
        'Y': Y.tolist()
    },
    # Impose known homoscedastic noise variance
    'Regression': {
        'SigmaNSQ': noise_var
    }
}
# Create the Kriging metamodel
myKriging = uq.createModel(MetaOpts)
```

Finally, below is another example of GP regression for an independent heteroscedastic noise case in which the noise variances at individual observation locations are given as vector:

```
# Create a hypothetical experimental design with noisy responses
X = np.linspace(0,14,15)
# Define a vector of noise variance at individual data locations
noise_var = np.array([0.3, 0.4, 4, 0.25, 0.16,
0.133, 0.5, 0.9, 5, 0.1600,
0.571, 0.02, 0.0225, 0.02, 0.8])
# Create noisy responses data set with the noise
Y = X * np.sin(X) + np.sqrt(noise_var) * np.random.randn(X.size)

# Define a Kriging metamodel
MetaOpts = {
    'Type': 'Metamodel',
    'MetaType': 'Kriging',
    'ExpDesign': {
        'Sampling': 'User',
        'X': X.tolist(),
        'Y': Y.tolist()
    },
    # Impose the known heteroscedastic noise variance
    'Regression': {
        'SigmaNSQ': noise_var.tolist()
    }
}
# Create the Kriging metamodel
myKriging = uq.createModel(MetaOpts)
```

The results of both cases are visualized in Figure 13. Notice that in Figure 13(b), the noise variance associated for each experimental design (observation) points differs. A Kriging example for the different cases of noise variance specification is provided in `uq_Example_Kriging_08_Regression`.



(a) known homoscedastic noise

(b) known heteroscedastic noise

Figure 13: Example of one-dimensional GP regression models predictions with known homoscedastic and independent heteroscedastic noise variances.

2.6 Kriging metamodels of vector-valued models

All the examples presented so far in this chapter dealt with scalar-valued models. If the model (or when manually specified, the experimental design) is having vector-valued (multiple) outputs, UQ[PY]LAB performs an independent Kriging metamodel for each output component with shared (common) experimental design and trend basis functions. No additional configuration is needed to enable this behavior. A Kriging example for model with multiple outputs can be found in the UQ[PY]LAB example script `uq_Example_Kriging_07_MultipleOutputs`.

2.6.1 Accessing the results

Calculating a Kriging metamodel on a model with multiple outputs will result in an output list of dictionaries. As an example, a model with nine outputs will produce a list with nine dictionaries for `myKriging['Kriging']`, each dictionary having the following keys:

```
beta
sigmaSQ
theta
```

Each element of the `Kriging` dictionary is functionally identical to its scalar counterpart in Section 2.3.1. Similarly, the `myKriging['Error']` dictionary becomes a list of dictionaries with key:

```
LOO
```

2.7 Using a validation set

If an independent validation set is provided (see Table 19 in Section 3.1.8), UQ[PY]LAB automatically computes the validation error according to Eq. (1.62). To provide a validation set stored in, for example, `X_val` and `Y_val`, the following option is set:

```
MetaOpts['ValidationSet'] = {
    'X': X_val,
    'Y': Y_val
}
```

The value of the validation error is stored in `myKriging['Error']['Val']` (see Table 22) and will also be displayed when invoking `uq.print(myKriging)`.

2.8 Using Kriging predictor as a model

Regardless of the configuration options that are used to construct a Kriging metamodel (as described in Section 2.4), the resulting Kriging metamodel can be used to predict new points according to, for instance in the case of noise-free responses, Eqs. (1.6)-(1.7). Indeed, after a Kriging MODEL object is created in UQ[PY]LAB, it can be used just like an ordinary UQ[PY]LAB MODEL object (for details, see the UQ[PY]LAB User Manual – the MODEL module).

Consider the example in Section 2.1. After obtaining a Kriging metamodel, users can now evaluate the mean of the Kriging predictor on point $x = 1$ as follows:

```
x = 1
YmuKG = uq.evalModel(myKriging, x)
```

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

```
YmuKG,YvarKG = uq.evalModel(myKriging, x)
```

As most functions within UQ[PY]LAB, model evaluations are vectorized. Therefore, evaluating multiple points at a time is much faster than repeatedly evaluating one point at a time. For example:

```
# We use reshape(-1,1) to ensure that X is a column vector
X = np.arange(0, 15, 0.1).reshape(-1,1)
# Evaluate the mean and variance of the Kriging predictor on X
YmuKG,YvarKG = uq.evalModel(myKriging, X)
```

2.9 Specifying manually a Kriging predictor (*predictor-only* mode)

The Kriging module in UQ[PY]LAB can also be used to build custom Kriging-based models which may be used as predictors (as in Section 2.8). This allows, for instance, to import a Kriging metamodel calculated with another software into the UQ[PY]LAB framework, or to

create an *ad-hoc* Kriging model.

Below is an example of how to create a custom Kriging metamodel that is similar to the one that was obtained in Section 2.3.

```
X = np.arange(0, 15, 2)
# Calculate the response of the model
Y = X * np.sin(X)

# Define a custom Kriging object
MetaOpts = {
    'Type': 'Metamodel',
    'MetaType': 'Kriging',
    'ExpDesign': {
        'Sampling': 'User',
        'X': X.tolist(),
        'Y': Y.tolist()
    },
    'Kriging': {
        'Trend': { #Select ordinary Kriging
            'Type': 'ordinary'
        },
        # Set the values of beta, sigma^2 and theta
        'beta': 69.84,
        'sigmaSQ': 2.566e5,
        'theta': 9.999,
        # Select ellipsoidal, Matern-3/2 correlation function
        'Corr': {
            'Type': 'ellipsoidal',
            'Family': 'matern-3_2'
        }
    }
}

# Create the metamodel
myCustomKriging = uq.createModel(MetaOpts)

# Evaluate the metamodel on some new points
X_new = np.arange(1, 13, 2).reshape(-1,1)
Y_new = uq.evalModel(myCustomKriging, X_new)
```

If the metamodel has more than one output, it is sufficient to specify the same information for each output in `MetaOpts['Kriging']`, which now needs to be a list of dictionaries. Furthermore, to define a custom regression model, a value to an additional field `MetaOpts['Kriging']['sigmaNSQ']` must be specified.

Note: By default, the nugget value is set to $\nu = 10^{-10}$ when creating a custom Kriging metamodel (see Section 2.4.3.2 on nugget).

2.10 Performing Kriging on an auxiliary space (scaling)

The Kriging module offers various options for scaling the experimental design before calculating the metamodel. That is, instead of $\mathcal{X}$, UQ[PY]LAB may use the *scaled* experimental design $\mathcal{U}$. Depending on the value of the option `MetaOpts['Scaling']` and whether a prob-

abilistic input model has been defined (in `MetaOpts['Input']`), the transformation $\mathcal{X} \mapsto \mathcal{U}$ varies. All the possible cases are summarized in Table 3.

Table 3: Available experimental design transformations

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

By setting `MetaOpts['Scaling'] = True`, the experimental design is scaled so that it has a zero mean and a unit variance (*i.e.*, standardized). If no input model has been specified, then the mean and variance $(\mu_{\mathcal{X}}, \sigma_{\mathcal{X}})$ are empirically estimated from the available data specified in `MetaOpts['ExpDesign']['X']`. If, however, an input model 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 $\boldsymbol{X}$.

If an additional INPUT object, say `my_scaled_input`, is set to the `MetaOpts['Scaling']` option:

```
MetaOpts['Scaling'] = my_scaled_input
```

then $\mathcal{U} = \tau(\mathcal{X})$ where τ denotes the generalized *isoprobabilistic transformation* between the two probability spaces (for more information, refer to the UQ[PY]LAB User Manual – the INPUT module).

Note: By default, `MetaOpts['Scaling'] = True`, that is the experimental design is scaled.

2.11 Drawing sample paths from a Gaussian process posterior

As mentioned in Section 1.2, a Kriging predictor is a Gaussian process posterior (with respect to the experimental design). Thus, instead of retrieving the mean and confidence bounds of the Gaussian process posterior, users may also draw an arbitrary number of sample paths or realizations from the process. This can be achieved by exploiting the Gaussian property of any Kriging metamodel. Each sample path of the Gaussian process $\boldsymbol{Y}$, discretized over a set of points $\mathcal{X}_s = \left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N_s)}\right\}$, follows a multivariate Gaussian distribution, *i.e.*, $\boldsymbol{Y} \sim \mathcal{N}_{N_s}\left(\boldsymbol{\mu}_{\widehat{\boldsymbol{Y}}}, \boldsymbol{C}_{\widehat{\boldsymbol{Y}}}\right)$ where $\boldsymbol{\mu}_{\widehat{\boldsymbol{Y}}}$ and $\boldsymbol{C}_{\widehat{\boldsymbol{Y}}}$ corresponds to the posterior mean and the posterior covariance matrix of the Kriging predictor evaluated on $\mathcal{X}_s$, respectively.

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

```
Ns = 500
Xs = np.linspace(0, 15, Ns).reshape(-1, 1)
mu, _, C = uq.evalModel(myKriging, Xs, nargout=3)
```

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

```
Ys = mu + np.matmul(np.linalg.cholesky(C), \
np.random.multivariate_normal(np.zeros(Ns), np.eye(Ns), 1).T)
```

A set of 15 such sample paths is plotted in Figure 14 together with the mean of the Kriging predictor.

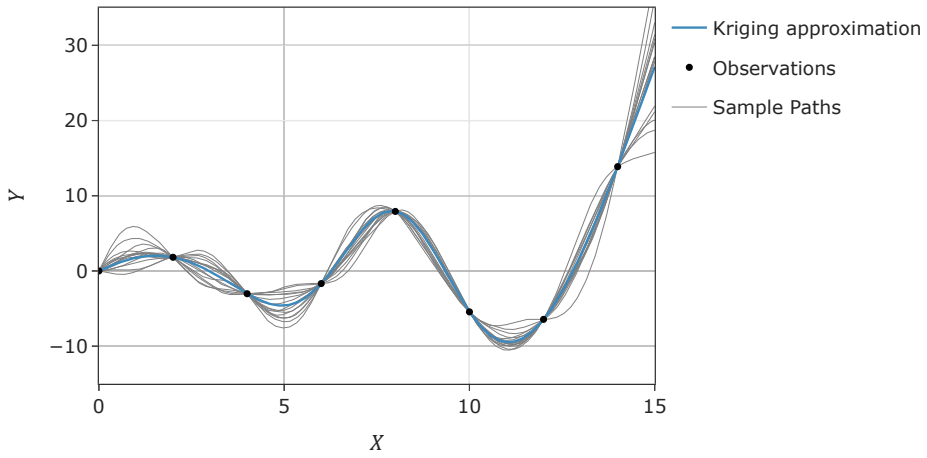


Figure 14: The mean of the Kriging predictor together with 15 sample paths of the posterior Gaussian process.

2.12 Using Kriging with constant inputs

In some analyses, users may need to assign a constant value to one or more input variables. When this is the case, the Kriging metamodel is calculated by internally removing the constant parameters from the full inputs. This process is transparent to users as the model is still evaluated using the full set of parameters (including those which are set constant). UQ[PY]LAB will automatically and appropriately account for the set of input parameters which are declared constant.

To set a parameter to constant, *e.g.*, with `const_value` as its value, the following options are specified (for details, see UQ[PY]LAB User Manual – the INPUT module):

```
InputOpts = {
    'Marginals': [{
        'Type': 'Constant',
        'Parameters': [const_value]
    }]
}
```

Furthermore, when the standard deviation of an input parameter is set to zero, UQ[PY]LAB automatically sets the marginal of this parameter to the type `Constant`. For example, the following uniformly distributed variable whose upper and lower bounds are identical (therefore its standard deviation is zero) is automatically set to a constant with value 1:

```
InputOpts = {
    'Marginals': [{
        'Type': 'Constant',
        'Parameters': [1, 1]
    }]
}
```

Chapter 3

Reference List

How to read the reference list

Python dictionaries play an important role throughout the UQLAB syntax. They offer a natural way to semantically group configuration options and output quantities. Due to the complexity of the algorithms implemented, it is not uncommon to employ nested dictionaries to fine-tune the inputs and outputs. Throughout this reference guide, a table-based description of the configuration dictionaries is adopted.

The simplest case is given when a value of a dictionary key is a simple value or a list:

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, from left to right, correspond to the name, the data type and a brief description of each key-value pair. At the beginning of each row a symbol is given to inform as to whether the corresponding key 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 keys 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 the value of one of the keys of a dictionary is a dictionary itself, a link to a table that describes the structure of that nested dictionary is provided, as in the case of the `Options` key in the following example:

Table X: `Input`			
●	`Name`	String	Description
☐	`Options`	Table Y	Description of the `Options` dictionary

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',
  'VALUE1' : {
    'Val1Opt1' : ... ,
    '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`	Float	Description

Table Z: `Input.VALUE2`

☐	`Val2Opt1`	String	Description
☐	`Val2Opt2`	Float	Description

3.1 Create a Kriging metamodel

Syntax

```
myKriging = uq.createModel(MetaOpts)
```

Input

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

Table 4: `MetaOpts`

●	`Type`	`'Metamodel'`	Select the metamodeling tool
●	`MetaType`	`'Kriging'`	Select Kriging
☐	`Name`	String	Unique identifier for the metamodel
☐	`ExpDesign`	Table 5	Option to specify an experimental design (Section 2.4.1)
☐	`Input`	String	Option to specify an INPUT object that describes the inputs of the metamodel. For example, you specify the INPUT `myInput` by supplying `myInput['Name']`
☐	`FullModel`	String	Option to specify the MODEL object that describes the full computational model of the metamodel. The object is used to compute the model responses. For example, you specify the MODEL `myModel` by supplying `myModel['Name']`
☐	`Trend`	Table 6	Options to specify the Kriging trend (Sections 1.3 and 2.4.2)
☐	`Corr`	Table 8	Options to specify the correlation function (Section 2.4.3)
☐	`EstimMethod`	String default: `'CV'`	Select the method to estimate Kriging parameters (Section 2.4.4)
		`'CV'`	Cross-validation estimation (Section 1.5.2)
		`'ML'`	Maximum-likelihood estimation (Section 1.5.1)
☐	`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	Options related to the optimization in the estimation of Kriging parameters (Sections 1.6 and 2.4.5)
☐	`Regression`	Table 16	Options to specify the noise in the model responses for a Gaussian process regression model (Sections 1.2.2 and 2.5)
☐	`Kriging`	Table 18	Field to specify the parameters of a custom Kriging metamodel (Section 2.9)
☐	`KeepCache`	Boolean default: `True`	Flag to keep (*cached*) some important matrices. Keeping the cached matrices may speed up the Kriging predictor calculation for some problems. If set to `False`, some cached matrices are removed after the metamodel has been created.
☐	`Scaling`	Boolean or String default: true	Conduct Kriging calculation in an auxiliary space, either by standardizing the experimental design or using an INPUT object. This affects the value of various Kriging parameters, notably the correlation function hyperparameters (Section 2.10).
		Boolean	Flag to scale the experimental design. If set to `True`, the experimental design will be scaled (specifically, standardized) before calculating the Kriging metamodel.
		String	The name of the INPUT object that defines the auxiliary probabilistic space
☐	`ValidationSet`	Table 19	Independent validation data set (Section 2.7)

3.1.1 Experimental design options

Table 5: `MetaOpts['ExpDesign']`			
●	`Sampling`	String default: `'LHS'`	Type of sampling
		`'MC'`	Monte Carlo sampling
		`'LHS'`	Latin Hypercube sampling
		`'Sobol'`	Sobol' sequence sampling
		`'Halton'`	Halton sequence sampling

		`'User'`	User-defined experimental design, defined in `MetaOpts['ExpDesign']['X']` and `MetaOpts['ExpDesign']['Y']`.
☐	`NSamples`	Integer	Number of sample points to draw. It is required for all types of sampling.
☐	`X`	$N \times M$ Numpy Array of Floats	User-defined experimental design (model inputs) $\mathcal{X}$. It only applies, and is required, if the type of sampling is `'User'`.
☐	`Y`	$N \times M$ Numpy Array of Floats	User-defined model responses $\mathcal{Y}$. It only applies, and is required, if the type of sampling is `'User'`.

3.1.2 Trend options

Table 6: `MetaOpts['Trend']`

☐	`Type`	String default: `'ordinary'`	Type of the Kriging trend (Sections 1.3 and 2.4.2)
		`'simple'`	Known trend function (*i.e.*, simple Kriging), defined in `.Trend.CustomF`. defined in `metaopts['Trend']['CustomF']`. No estimation of the coefficients takes place.
		`'ordinary'`	Unknown constant trend (*i.e.*, ordinary Kriging)
		`'linear'`	Linear polynomial trend
		`'quadratic'`	Quadratic polynomial trend
		`'polynomial'`	Polynomial trend of an arbitrary degree, defined in `MetaOpts['Trend']['Degree']`.
		`'custom'`	User-defined trend (Section 2.4.2.2)
☐	`Degree`	Integer default: 0	Degree of the polynomial trend. It only applies if the Kriging trend type is `'polynomial'`.
☐	`CustomF`	Float , String, or List	Field to set up custom trend function
		Float	Constant trend in simple Kriging
		String	Function name that returns $f(\boldsymbol{x})$, an arbitrary trend function

		List of strings	Respective names of functions $f_i(\boldsymbol{x}), i = 1, \ldots, P$, such that each function evaluated on $\mathcal{X}$ returns the i'th column of $\boldsymbol{F}$
☐	`TruncOptions`	Table 7	Polynomial basis truncation options

Table 7: `MetaOpts.Trend['TruncOptions']`

☐	`qNorm`	Float default: 1	Parameter for the *q-norm* (or hyperbolic) truncation scheme. For details, see the UQ[PY]LAB User Manual – Polynomial Chaos Expansions.
☐	`MaxInteraction`	Integer	Maximum interaction of the input variables. This is used to limit the basis terms to up to the given `MaxInteraction`. For details, see the UQ[PY]LAB User Manual – Polynomial Chaos Expansions.
☐	`Custom`	List of Floats	User-defined basis specified using multi-indices

3.1.3 Correlation function options

Table 8: `MetaOpts['Corr']`

☐	`Family`	String default: `'Matern-5_2'`	One-dimensional correlation family (Section 1.4.1)
		`'Linear'`	Linear correlation (Eq. (1.28))
		`'Exponential'`	Exponential correlation (Eq. (1.29))
		`'Gaussian'`	Gaussian correlation (Eq. (1.30))
		`'Matern-3_2'`	Matérn-3/2 correlation (Eq. (1.32))
		`'Matern-5_2'`	Matérn-5/2 correlation (Eq. (1.33))
☐	`Type`	String default: `'Ellipsoidal'`	Type of the correlation function
		`'Ellipsoidal'`	Ellipsoidal correlation (Eq. (1.34))
		`'Separable'`	Separable correlation (Eq. (1.35))
☐	`Isotropic`	Boolean default: false	Flag to determine whether the correlation function is isotropic or anisotropic (Section 1.4.3)

☐	`Nugget`	Float scalar or List default: 10^{-10}	Set of values that are added to the main diagonal of the correlation matrix $\boldsymbol{R}$ (Section 1.4.4) • If scalar, add this quantity to the diagonal elements of $\boldsymbol{R}$. • If List , add each element to the corresponding diagonal element of $\boldsymbol{R}$.

3.1.4 Estimation method options

Table 9: `MetaOpts['CV']`

☐	`LeaveKOut`	Integer ($< N$) default: 1	Number of sample points left out in cross-validation (CV) (Section 2.4.4) • The number of sample points in $\mathcal{D}_k$ (Eq. (1.53)). It can be any integer between 1 and N (the size of experimental design). • If the total number of sample points is not divisible by the number of sample points left out, the last subset contains the remainder. • The default is 1, also known as *leave-one-out* (LOO) CV.

3.1.5 Hyperparameters optimization options

Table 10: `MetaOpts['Optim']`

☐	`InitialValue`	Scalar or $M \times 1$ Float default: 1.0	Initial estimate of the correlation parameters
		Scalar Float	Initial estimate of the correlation parameter. If $M > 1$, the same value is assigned for all input dimensions.
		$M \times 1$ Float	Initial estimates of the correlation parameters for each of the M input dimensions
☐	`Bounds`	$2 \times M$ or 2×1 Float default: $[10^{-3}, 10]^T$	Bound of admissible values of the correlation parameters, in the form `[lower_bound; upper_bound]`.

		2×1 Float	Bound of the correlation parameter. If $M > 1$, the same value is assigned for all input dimensions.
		$2 \times M$ Float	Bounds of the correlation parameters for each of the M input dimensions
☐	`Display`	String default: `'none'`	Option to determine the verbosity of the optimization process
		`'none'`	Nothing will be printed to the command window
		`'final'`	Only the final result of the optimization process will be printed
		`'iter'`	State of the optimization process will be printed after each iteration
☐	`MaxIter`	Integer default: 20	Maximum number of iterations or generations
☐	`Tol`	Float default: 10^{-4}	Covergence tolerance of the optimization
☐	`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 the `fmincon` function of MATLAB
		`'GA'`	Genetic Algorithm (GA) optimization using the `ga` function of MATLAB
		`'HGA'`	Hybrid Genetic Algorithm (HGA) optimization using the functions `ga` and `fmincon` of MATLAB
		`'CMAES'`	Covariance Matrix Adaptation-Evolution Strategy (CMA-ES) optimization using the function `uq_cmaes` of UQLIB
		`'HCMAES'`	Hybrid Covariance Matrix Adaptation-Evolution Strategy (HCMA-ES) optimization using the functions `uq_cmaes` of UQLIB and `fmincon` of MATLAB
⊞	`BFGS`	Table 11	Options relevant to the L-BFGS 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

Note: The convergence tolerance defined under `Metaopts.Optim.Tol` may slightly differ depending on the optimization method. For MATLAB optimization functions (`fmincon` and `ga`), check the definition of the option `'TolFun'` of the respective function.

Table 11: `MetaOpts['Optim']['BFGS']`

☐	nLM	Integer (≥ 1) default: 5	Limited memory size of the L-BFGS method (nLM ≥ 1)

Table 12: `MetaOpts['Optim']['GA']`

☐	nPop	Integer default: 30	Population size of each generation
☐	nStall	Integer default: 5	Maximum number of stall generations

Table 13: `MetaOpts['Optim']['HGA']`

☐	nPop	Integer default: 30	Population size of each generation
☐	nStall	Integer default: 5	Maximum number of stall generations
☐	nLM	Integer (≥ 1) default: 5	Limited memory size of the L-BFGS method (nLM ≥ 1). The method is executed starting, as the initial value, with the optimal value obtained from the GA optimization.

Table 14: `MetaOpts['Optim']['CMAES']`

☐	nPop	Integer default: 30	The population size of each generation
☐	nStall	Integer default: 5	The maximum number of stall generations

Table 15: `MetaOpts['Optim']['HCMAES']`

☐	`nPop`	Integer default: 30	Population size of each generation
☐	`nStall`	Integer default: 5	Maximum number of stall generations
☐	`nLM`	Integer default: 5	Limited memory size of the L-BFGS method (`nLM` ≥ 1). The method is executed starting, as the initial value, with the optimal value obtained from the CMA-ES optimization.

3.1.6 Regression options

Table 16: `MetaOpts['Regression']`

☐	`SigmaNSQ`	`'none'`, `'auto'`, Boolean, or Float default: `'none'`	Specification of the output noise variance in a regression model (Section 2.5)
		`'none'` or `False`	Flag to ignore the noise variance. The resulting Kriging model will be in interpolation mode.
		`'auto'` or `True`	Flag to estimate the homogeneous (homoscedastic) noise variance
		Scalar Float	Homoscedastic noise variance σ_{n}^2
		$N \times 1$ Float	Independent heterogeneous (heteroscedastic) noise variance $\boldsymbol{\sigma}_{\mathrm{n}}^2$
		$N \times N$ Float	Noise covariance matrix $\boldsymbol{\Sigma}_{\mathrm{n}}$
☐	`SigmaSQ`	Table 17	Initial value and bound for the optimization of the Gaussian process variance σ^2 in the case of a regression model with known noise variance

Note: If the metamodel has more than one outputs, then `MetaOpts['Regression']` is expected to be a *list* with length equal to the number of outputs and that the parameters in Table 16 are defined for each `MetaOpts['Regression'][idx]` (where `idx` ranges from 0 to the total number of outputs−1). If only one regression option is specified, then the option applies to all output by default.

Table 17: `MetaOpts['Optim']['Regression']['SigmaSQ']`

☐	`InitialValue`	Float default: $0.5\mathrm{Var}\left[\mathcal{Y}\right]$	Initial value of the Gaussian process variance

☐	`Bound`	2×1 Float default: $[0.1 \times \mathrm{Var}[\mathcal{Y}]\,; 10 \times \mathrm{Var}[\mathcal{Y}]]$	Lower and upper bounds for the Gaussian process variance
where $\mathrm{Var}[\mathcal{Y}] = \frac{1}{N}\sum_i (y_i - \bar{y})^2$ and $\bar{y} = \frac{1}{N}\sum_i y_i$ are the sample variance and mean of the observed responses, respectively.			

3.1.7 Custom Kriging options

To create a user-defined Kriging model as desribed in Section 2.9, a `MetaOpts['Kriging']` options must be set according to Table 18.

Table 18: `MetaOpts['Kriging']`

●	`Trend`	Table 6	Kriging trend definition
●	`beta`	Float	Values of the trend coefficients $\boldsymbol{\beta}$ in Eq. (2.2)
●	`sigmaSQ`	Float	Value of σ^2 in Eq. (2.2)
●	`theta`	Float	Values of $\boldsymbol{\theta}$ in Eq. (2.2)
●	`Corr`	Table 8	Correlation function definition (See below **Note**)
☐	`SigmaNSQ`	Scalar, List , or List of lists with Float entries	Values of the output noise variance in a regression model (Sections 1.2.2 and 1.2.2)
		Scalar Float	Homoscedastic noise variance σ_{n}^2
		$N \times 1$ Float	Independent heteroscedastic noise variance $\boldsymbol{\sigma}_{\mathrm{n}}^2$
		$N \times N$ Float	Noise covariance matrix $\boldsymbol{\Sigma}_{\mathrm{n}}$

Note: In the current version of UQ[PY]LAB, only the default correlation function handle `uq_eval_Kernel` is supported. All other options follow Table 8.

Note: If the metamodel has more than one outputs, then `MetaOpts['Kriging']` is expected to be a *list* with length equal to the number of outputs and that the parameters in Table 18 are defined for each `MetaOpts['Kriging'][idx]` (where `idx` ranges from 0 to the total number of outputs−1).

Note: To create a custom Kriging model, the fields `MetaOpts['ExpDesign']['X']` and `MetaOpts['ExpDesign']['Y']` are also required (see Table 5).

3.1.8 Validation Set

If an independent validation set is provided, UQ[PY]LAB automatically calculates the validation error of the created Kriging model (Section 1.7.2). The options are set according to Table 19.

Table 19: `MetaOpts['ValidationSet']`

●	`X`	$N \times M$ Float	User-specified validation set experimental design $\mathcal{X}_{\text{val}}$
●	`Y`	$N \times N_{\text{out}}$ Float	User-specified validation set response $\mathcal{Y}_{\text{val}}$

3.2 Accessing the Results

Table 20: `myKriging = uq.createModel(...)`		
`Name`	String	Name of the Kriging metamodel
`Kriging`	Table 21	Kriging results
`Error`	Table 22	Error metrics of the metamodeling result
`ExpDesign`	Table 23	Options and values related to experimental design
`Options`	Table 4	Options that were defined in `MetaOpts` variable (Section 3.1)
`Internal`	Table 24	Internal fields

Table 21: `myKriging['Kriging']`		
`beta`	$P \times 1$ Float	Values of $\boldsymbol{\beta}(\boldsymbol{\theta})$ in Eq. (2.2)
`sigmaSQ`	Float	Value of $\sigma^2(\boldsymbol{\theta})$ in Eq. (2.2)
`theta`	Scalar or $1 \times M$ Float	Values of $\boldsymbol{\theta}$ in Eq. (2.2)
`sigmaNSQ`	Scalar, List , or List of lists with Float entries	Value of noise variance in Gaussian process regression (Sections 1.2.2 and 1.2.2)
	Scalar Float	Homoscedastic noise variance σ_{n}^2
	$N \times 1$ Float	Independent heteroscedastic noise variance $\boldsymbol{\sigma}_{\mathrm{n}}^2$
	$N \times N$ Float	Noise covariance matrix $\boldsymbol{\Sigma}_{\mathrm{n}}$

Table 22: `myKriging['Error']`		
`LOO`	Float	Leave-One-Out (LOO) error, calculated using Eq. (1.61)
`Val`	Float	Validation error (see Eq. (1.62) and Section 2.7). Only available if a validation set is provided (see Table 19).

Note: In general, the fields `myKriging['Kriging']` and `myKriging['Error']` are *lists* with length equal to the number of outputs of the Kriging model. To access the results that correspond to the respective output, one should use, for example, `myKriging['Internal']['Kriging'][k][field]` where `k` $= 0, \ldots, N_{\mathrm{out}} - 1$.

Table 23: `myKriging['ExpDesign']`		
`NSamples`	Float	Number of sample points
`Sampling`	String	Sampling method

`X`	$N \times M$ Float	Values in the experimental design
`U`	$N \times M$ Float	Experimental design values in the auxiliary (scaled) space. For details, see Section 2.10
`Y`	$N \times N_{\text{out}}$ Float	Observed model responses (output) $\mathcal{Y}$ that corresponds to the experimental design $\mathcal{X}$

3.2.1 Internal fields (advanced)

Note: The internal fields of the Kriging metamodel object are not intended to be accessed or changed in most typical usage scenarios of the module. Note that some internal fields are not documented in this user manual.

Table 24: `myKriging['Internal']`

`Kriging`	Table 25	Internal fields with data related to the Kriging model
`Runtime`	Dictionary	Internal fields with variables that are used during the calculation of the kriging metamodel
`Error`	Table 26	Internal fields related to the metamodeling error
`ExpDesign`	Table 27	Internal fields related to the experimental design
`Scaling`	Boolean or INPUT object	Scaling of the experimental design (see Section 2.10)
`Regression`	Table 28	Internal fields related to a Gaussian process regression model (*e.g.*, noise of the responses). It is only available in the case of a regression model

Table 25: `myKriging.Internal['Kriging']`

`Trend`	Table 29	Internal fields related to the trend
`GP`	Table 30	Internal fields related to the Gaussian process
`Optim`	Table 31	Internal fields related to the optimization of the hyperparameters
`sigmaNSQ`	Float	Internal field that contains the value of noise variance in Gaussian process regression (refer to the entry `sigmaNSQ` in Table 21)
`Cached`	Dictionary	Internal fields related to cached variables

Note: The fields `myKriging['Internal']['Kriging']` and `myKriging['Internal']['Error']` are, in general, *list objects* with length equal to the number of outputs of the metamodel. To access the results that correspond to the respective output, one should use, for example, `myKriging['Internal']['Kriging'][k][field]` where $k = 0, ..., N_{\text{out}} - 1$.

Table 26: `myKriging.Internal['Error']`

`LOOmean`	$N \times 1$ Float	Error between the observed data and the mean of the Kriging predictor from leave-one-out (LOO) cross-validation (CV) (*i.e.*, from the prediction made on a point conditioned on all the other points in the observed data)
`LOOsd`	$N \times 1$ Float	Standard deviation of the Kriging predictor from LOO CV
`varY`	$1 \times N_{\text{out}}$ Float	The variance of the observed model responses (output) $\mathcal{Y}$. This quantity can be multiplied with the relative LOO error in `myKriging['Error']['LOO']` to get the absolute value of the LOO error (Eq. (1.61)).

Table 27: `myKriging.Internal['ExpDesign']`

`muX`	$1 \times M$ Float	Mean of the inputs used in the scaling of the experimental design (Section 2.10). • If an experimental design is specified in `MetaOpts['ExpDesign']`, then this corresponds to the empirical mean of each input dimension. • If INPUT object is specified in `MetaOpts['Input']`, then this corresponds to the first moment of the corresponding marginal of each input dimension.
`stdX`	$1 \times M$ Float	Standard deviation of the inputs used in the scaling of the experimental design (Section 2.10). • If an experimental design is specified in `MetaOpts['ExpDesign']`, then this corresponds to the empirical standard deviation of each input dimension. • If INPUT object is specified in `MetaOpts['Input']`, then this corresponds to the second moment of the corresponding marginal of each input dimension.

`varY`	$1 \times N_{\text{out}}$ Float	Variance of the observed model responses (output) $\mathcal{Y}$. This quantity can be multiplied with the relative LOO error in `myKriging['Error']['LOO']` to get the absolute value of the LOO error (Eq. (1.61)).

Table 28: `myKriging.Internal['Regression']`

`IsRegression`	Boolean	Flag that indicates the current Kriging model is a Gaussian process regression model
`EstimNoise`	Boolean	Flag that indicates that the noise variance is estimated
`Tau`	Dictionary	Dictionary that contains the initial value and bounds used in the optimization of the τ parameter (Eq. (1.21))
`SigmaSQ`	Dictionary	Dictionary that contains the initial value and bounds used in the optimization of the σ^2 (Gaussian process variance)
`SigmaNSQ`	Scalar, List , or List of lists with Float entries	Estimated or specified noise variance.
`IsHomoscedastic`	Boolean	Flag that indicates the Gaussian process regression model is homoscedastic, with a constant noise variance.

Table 29: `myKriging.Internal.Kriging['Trend']`

`F`	$N \times P$ Float	Observation matrix, *i.e.*, the basis functions of the Kriging trend f_j's evaluated on the experimental design (Eq. (2.2)).
`beta`	$p \times 1$ Float	Values of $\boldsymbol{\beta}(\widehat{\boldsymbol{\theta}})$ in Eq. (2.2)

Table 30: `myKriging.Internal.Kriging['GP']`

`R`	$N \times N$ Float	Correlation matrix of the Gaussian process on the experimental design points (*i.e.*, $\boldsymbol{R} = R(\mathcal{X}, \mathcal{X})$)
`sigmaSQ`	Float	Value of σ^2 in Eq. (2.2)

Table 31: `myKriging.Internal.Kriging['Optim']`

`Theta`	Scalar or $1 \times M$ Float	Optimum value of $\boldsymbol{\theta}$
`Tau`	Float	Ratio of σ_{n}^2 to $\sigma^2 + \sigma_{\text{n}}^2$ as defined in Eq. (1.21). Only available in a regression with unknown homoscedastic noise variance.

`SigmaSQ`	Float	Gaussian process variance, directly optimized, in the case of known noise variance. Only available in a regression with known noise variance (Sections 1.5.2.3 and 1.5.1.3)
`ObjFun`	Float	Objective function value at the optimum $\boldsymbol{\theta}$
`InitialObjFun`	Float	Objective function value at the initial estimate of $\boldsymbol{\theta}$
`nEval`	Float	Number of objective function evaluations during the optimization process
`nIter`	Float	Number of iterations (or generations) during the optimization process. For hybrid methods (*i.e.*, `'HGA'`, `'HCMAES'`), only the number of generations is taken into account.

3.3 Kriging predictor

Syntax

```
Y_mu = uq.evalModel([Model=None, ]X=None[, nargout=1])
Y_mu, Y_sigma2 = uq.evalModel(..., nargout=2)
Y_mu, Y_sigma2, Y_cov = uq.evalModel(..., nargout=3)
```

Description

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

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

`[Y_mu,Y_sigma2] = uq.evalModel(..., nargout=2)` additionally returns the variance of the Kriging predictor ($N \times N_{\text{out}}$).

`[Y_mu,Y_sigma2,Y_cov] = uq.evalModel(..., nargout=3)` additionally returns the covariance matrices of the Kriging predictor ($N \times N \times N_{\text{out}}$), *i.e.*, one covariance matrix per output dimension. 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)`.

Note: By default, the *most recently created* model or metamodel is the current active model.

3.4 Printing and visualizing a Kriging metamodel

UQ[PY]LAB offers two commands to conveniently print reports containing contextually relevant information for a given Kriging metamodel object.

3.4.1 Printing the results: `uq.print`

Syntax

```
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 report on the information about the first output dimension are printed.

`uq.print( myKriging , outIdx)` prints a report on the configuration options and results of the Kriging metamodel object `myKriging` for the output dimensions specified in the array `outIdx`.

`uq.print( myKriging , outIdx, 'Option1', 'Option2', ...)` prints a report only on selected configuration options or results, the selection of which, are specified by the string options `'Option1'`, `'Option2'`, *etc.* See Table 32 for the available options.

Table 32: `uq.print` options

`'beta'`	Prints out the regression coefficients $\boldsymbol{\beta}$
`'theta'`	Prints out the final hyperparameters $\boldsymbol{\theta}$ (either the optimal value or the user-specified value if they are manually specified)
`'F'`	Prints out the observation matrix $\boldsymbol{F}$
`'optim'`	Prints out hyperparameters optimization information (*i.e.*, optimization methods and optimized hyperparameters)
`'trend'`	Prints out trend-related information (*e.g.*, the type, coefficients)
`'GP'`	Prints out the Gaussian process-related information (*e.g.*, the correlation family, type)
`'R'`	Prints out the correlation matrix $\boldsymbol{R}$
`'regression'`	Prints out the Gaussian process regression-related information

Examples

`uq.print( myKriging ,[1 3])` prints the Kriging metamodeling results for output dimensions 1 and 3.

`uq.print( myKriging , 3, 'theta', 'R')` only prints the hyperparameters $\boldsymbol{\theta}$ and correlation matrix $\boldsymbol{R}$ for output dimension 3.

3.4.2 Visualize the results: `uq.display`

Syntax

```
uq.display(myKriging[, outIdx[, 'R']])
```

Description

`uq.display(myKriging)` creates a visualization of the Kriging metamodel object `myKriging`. It plots the Kriging model predictions (*i.e.*, the mean and standard deviation) against the input. If the model has multiple outputs, only the prediction of the first output dimension is plotted.

`uq.display(myKriging, outIdx)` plots the Kriging model predictions against the input for the model output dimensions specified in the array `outIdx`.

`uq.display(myKriging, outIdx, 'R')` creates a display of the correlation matrix $\boldsymbol{R}$ of the metamodel object `myKriging` for the model output dimensions specified in the array `outIdx`.

Note: Creating plots of Kriging predictions against the model inputs using `uq.display` is only available for Kriging model in 1- and 2-dimension. In the case of 1-dimensional model, the function creates a line plot; while in the case of 2-dimensional, the function creates a contour plot. Visualizing the correlation matrix, however, can be used for arbitrary input dimensions.

Examples

`uq.display(myKriging, [1 3])` creates two plots, one plot for each selected output dimension, of Kriging predictions against the input for the Kriging metamodel object `myKriging`.

`uq.display(myKriging, 1:3, 'R')` displays the correlation matrices of the metamodel object `myKriging` for output dimensions 1, 2, and 3 in three separate figures.

References

Abramovitz, M. and I. A. Stegun (1965). *Handbook of mathematical functions*. New York: Dover Publications Inc. 9

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. 15

Byrd, R. H., M. E. Hribar, and J. Nocedal (1999). An interior point algorithm for large scale nonlinear programming. *SIAM Journal on Optimization 9*(4), 877–900. 16

Cressie, N. A. C. (1993). *Statistics for spatial data*. John Wiley & Sons Inc. 15

Dubourg, V. (2011). *Adaptive surrogate models for reliability analysis and reliability-based design optimization*. Ph. D. thesis, Université Blaise Pascal, Clermont-Ferrand, France. 3, 6, 11, 13

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional. 16

Hansen, N. and A. Ostermeier (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation 9*(2), 159–195. 17

Krige, D. G. (1951). A statistical approach to some mine valuation and allied problems on the Witwatersrand. Master's thesis, University of the Witwatersrand, South Africa. 1

Matheron, G. (1963). Principles of geostatistics. *Economic Geology 58*(2), 1246–1266. 1

Nocedal, J. (1980). Updating Quasi-Newton Matrices with Limited Storage. *Mathematics of Computation 35*(151), 773–782. 16

Rasmussen, C. and C. Williams (2006). *Gaussian processes for machine learning*. Adaptive computation and machine learning. Cambridge, Massachusetts: MIT Press. 1, 5, 10, 11, 14

Sacks, J., W. J. Welch, T. J. Mitchell, and H. P. Wynn (1989). Design and analysis of computer experiments. *Statistical Science 4*, 409–435. 1, 11

Santner, T., B. Williams, and W. Notz (2003). *The design and analysis of computer experiments*. Springer series in Statistics. Springer. 1, 2, 3, 13, 15