ETH zürich

UQLab User Manual
Polynomial Chaos Kriging

R. Schöbi, S. Marelli, B. Sudret

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

Risk, Safety & Uncertainty Quantification

How to cite UQLab

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

How to cite this manual

R. Schöbi, S. Marelli, B. Sudret, UQLab user manual – Polynomial chaos Kriging, Report UQLab-V2.0-109, Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland, 2022

BibTeX entry

```
@TechReport{UQdoc_20_109,
author = {Sch\"obi, R. and Marelli, S. and Sudret, B.},
title = {{UQLab user manual -- Polynomial chaos Kriging}},
institution = {Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich,
Switzerland},
year = {2022},
note = {Report UQLab-V2.0-109}
}
```

Document Data Sheet

Document Ref.	UQLAB-V2.0-109
Title:	UQLAB user manual – Polynomial chaos Kriging
Authors:	R. Schöbi, S. Marelli, B. Sudret Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland
Date:	01/02/2022

Doc. Version	Date	Comments
V2.0	01/02/2022	UQLAB V2.0 release
V1.4	01/02/2021	UQLAB V1.4 release
V1.3	19/09/2019	UQLAB V1.3 release
V1.2	22/02/2019	UQLAB V1.2 release • added automatic calculation of validation error if a validation set is provided
V1.1	05/07/2018	UQLAB V1.1 release
V1.0	01/05/2017	Initial release

Abstract

Polynomial chaos Kriging (PC-Kriging) is a novel metamodelling technique which combines the advantages of Kriging (Gaussian process modelling) and polynomial chaos expansions (PCE). More specifically, PC-Kriging consists of a universal Kriging model, the trend of which is modelled by a sparse set of orthogonal polynomials.
UQLAB metamodelling tools provide an efficient, flexible and easy-to-use PC-Kriging module that allows one to apply state-of-the-art algorithms for different variations of PC-Kriging on a variety of applications. This manual for the PC-Kriging metamodelling module is divided into three parts:

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

Keywords: UQLAB, metamodelling, Kriging, PC-Kriging, Polynomial Chaos Expansions, PCE, Sparse PCE

Contents

Chapter 1

Theory

1.1 Introduction

In modern engineering and applied sciences in general, computational simulations become more and more complex and hence more time-consuming. In fact, the evaluation of a state-of-the-art finite element model may take hours to days. In this context, metamodelling (a.k.a surrogate modelling) attempts to reduce the computational costs and hence to allow for more sophisticated analyses, such as reliability analysis and design optimizations.
Polynomial-Chaos-Kriging (PC-Kriging) is a state-of-the-art metamodelling algorithm which was developed by the authors and which is based on the well-established metamodelling techniques Polynomial Chaos Expansions (PCE, see also UQLAB User Manual – Polynomial Chaos Expansions) and Kriging (see also UQLAB User Manual – Kriging (Gaussian process modelling)). PC-Kriging makes use of the regression-type PCE to capture the global behaviour of the computational model as well as the interpolation-type Kriging to capture local variations. This combination results in a metamodelling technique that is more efficient than PCE and Kriging separately.
The recent developments of PC-Kriging are implemented in UQLAB. Hence, this part is intended as an overview of the relevant theory and associated literature.

1.2 Basics

1.2.1 Universal Kriging

Kriging is a stochastic interpolation algorithm which assumes that the model output $y = \mathcal{M}(\boldsymbol{x})$ is a realization of a Gaussian process indexed by $\boldsymbol{x} \in \mathcal{D}_X \subset \mathbb{R}^M$. A Kriging model is described by the following equation (Santner et al., 2003; Lataniotis et al., 2021):

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

where $\boldsymbol{\beta}^\mathsf{T} \boldsymbol{f}(\boldsymbol{x})$ is the mean value of the Gaussian process (a.k.a. trend), σ^2 is the variance of Gaussian process, and $Z(\boldsymbol{x}, \omega)$ is a zero mean, unit variance, stationary Gaussian process which is characterized by a correlation function R and its hyperparameters $\boldsymbol{\theta}$.

For more details, the reader is referred to UQLAB User Manual – Kriging (Gaussian process modelling).

1.2.2 Polynomial chaos expansions

Consider a random vector with independent components $\boldsymbol{X} \in \mathbb{R}^M$ described by the joint probability density function (PDF) $f_{\boldsymbol{X}}$. Polynomial Chaos Expansions (PCE) approximates the computational model output $Y = \mathcal{M}(\boldsymbol{X})$ by a sum of orthonormal polynomials (Xiu and Karniadakis, 2002; Sudret, 2007):

$$Y \approx \mathcal{M}^{PC} = \sum_{\boldsymbol{\alpha} \in \mathcal{A}} y_{\boldsymbol{\alpha}} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{X}), \tag{1.2}$$

where $\Psi_{\boldsymbol{\alpha}}(\boldsymbol{X})$ are multivariate polynomials orthonormal with respect to the input distributions $f_{\boldsymbol{X}}$, $\boldsymbol{\alpha} \in \mathcal{A} \subset \mathbb{N}^M$ are multi-indices, and $y_{\boldsymbol{\alpha}}$ are the corresponding coefficients. For more details, the reader is referred to UQLAB User Manual – Polynomial Chaos Expansions.

1.3 Polynomial-Chaos-Kriging

1.3.1 Framework

Kriging interpolates the local variations of Y as a function of the neighbouring experimental design points, whereas PCE approximates well the global behaviour of Y. By combining the global and local approximation of these techniques, a more accurate metamodel is achieved.

Polynomial-Chaos-Kriging (PC-Kriging) is defined as a universal Kriging model the trend of which consists of a set of orthonormal polynomials (Schöbi et al., 2015, 2016; Kersaudy et al., 2015):

$$y \approx \mathcal{M}^{(\mathrm{PCK})}(\boldsymbol{x}) = \sum_{\boldsymbol{\alpha} \in \mathcal{A}} y_{\boldsymbol{\alpha}} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{X}) + \sigma^2 Z(\boldsymbol{x}, \omega), \tag{1.3}$$

where $\sum_{\boldsymbol{\alpha} \in \mathcal{A}} y_{\boldsymbol{\alpha}} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{X})$ is a weighted sum of orthonormal polynomials describing the trend of the PC-Kriging model, σ^2 and $Z(\boldsymbol{x}, \omega)$ denote the variance and the zero mean, unit variance, stationary Gaussian process, respectively, as introduced in Section 1.2.1. Hence, PC-Kriging can be interpreted as a universal Kriging model with a specific trend. In other words, the Kriging equations described in UQLAB User Manual – Kriging (Gaussian process modelling) are valid.

Constructing a PC-Kriging model consists of two parts: the determination of the optimal set of polynomials contained in the trend and the calibration of the Kriging model (*i.e.*determining the parameters $\{\boldsymbol{\theta}, \sigma^2, y_{\boldsymbol{\alpha}}\}$). The two parts can be combined in various ways. In UQLAB, *Sequential PC-Kriging* and *Optimal PC-Kriging* are implemented and presented in the following sections (Schöbi et al., 2015, 2016).

1.3.2 Sequential PC-Kriging

In Sequential PC-Kriging (SPCK), the set of polynomials and the Kriging metamodel are determined sequentially. In a first step the optimal set of polynomials $\mathcal{A}$ is determined by sparse PCE based on *least angle regression selection* (LARS) (Blatman and Sudret, 2011). Then, then entire set $\mathcal{A}$ is embedded into the PC-Kriging equation Eq. (1.3) as a trend, consisting of $P = |\mathcal{A}|$ regressors. Finally, the PC-Kriging metamodel is calibrated as a usual Kriging model, including the computation of the coefficients $y_{\boldsymbol{\alpha}}$.

1.3.3 Optimal PC-Kriging

In Optimal PC-Kriging (OPCK), the PC-Kriging model is obtained iteratively. As in SPCK, the optimal set of polynomials is determined by LARS. The LARS algorithm results in a sparse set of polynomials which are ranked according to their correlation to the current residual at each LARS iterations (in decreasing order). Each polynomial is then added individually to the trend of a PC-Kriging model. In each iteration, a new PC-Kriging model is calibrated. In the end, $P = |\mathcal{A}|$ PC-Kriging models are available, which are compared by means of their leave-one-out (LOO) error estimator (see UQLAB User Manual – Kriging (Gaussian process modelling)for more details). The Optimal PC-Kriging metamodel is then chosen as the PC-Kriging model that minimizes the LOO error.

1.3.4 *A posteriori* error estimation

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

$$\epsilon_{Val} = \frac{N-1}{N} \left[\frac{\sum_{i=1}^{N} \left(\mathcal{M}(\boldsymbol{x}_{Val}^{(i)}) - \mathcal{M}^{PCK}(\boldsymbol{x}_{Val}^{(i)}) \right)^2}{\sum_{i=1}^{N} \left(\mathcal{M}(\boldsymbol{x}_{Val}^{(i)}) - \hat{\mu}_{Y_{Val}} \right)^2} \right] \tag{1.4}$$

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

Chapter 2

Usage

In this section a reference problem will be set up to showcase how each of the techniques described in Part 1 can be deployed in UQLAB.

2.1 Reference problem: one-dimensional function

In the context of this manual, the following one-dimensional function is used as the reference computational model:

$$f(x) = x\,\sin(x). \tag{2.1}$$

The input random vector consists of a uniform random variable $X \sim \mathcal{U}(0, 15)$. An example UQLAB script that showcases several of the currently available PC-Kriging techniques on this function can be found in the example file:

`Examples/MetaModelling/PCK/uq_Example_PCK_01_XsinX.m`

2.2 Problem setup

2.2.1 Full model and probabilistic input model

The UQLAB framework is first initialized with the following command:

```
uqlab
```

To surrogate it using UQLAB, we need to first configure a basic MODEL object:

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

For more details about the configuration options available for a model, please refer to the UQLAB User Manual – the MODEL module.

Compared to plain Kriging, PC-Kriging *always* requires an input model due to the existence of PCE, even when using a given experimental design. The input variable can be defined as:

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

For more details about the configuration options available for an INPUT object, please refer to the UQLAB User Manual – the INPUT module.

2.3 Setup of a PC-Kriging model

The PC-Kriging module creates a MODEL object that can be used as any other model. Its configuration options, however, are generally more complex than for a basic Kriging or PCE metamodel.
The basic options common to any PC-Kriging metamodelling MODEL read:

```
MetaOpts.Type = 'Metamodel';
MetaOpts.MetaType = 'PCK';
```

The additional configuration options needed to properly create a PCK object in UQLAB are given in the following subsections.

2.3.1 Sequential PC-Kriging model

Computing a Sequential PC-Kriging (SPCK) requires then the specification of an experimental design. Here, we specify an experimental design consisting of 10 samples from a Sobol' sequence:

```
MetaOpts.ExpDesign.Sampling = 'Sobol';
MetaOpts.ExpDesign.NSamples = 10;
```

Further, we specify the type of PC-Kriging model and compute the metamodel:

```
MetaOpts.Mode = 'sequential';
mySPCK = uq_createModel(MetaOpts);
```

Once the model is created, a report with basic information about the PC-Kriging model can be printed as follows:

```
uq_print(mySPCK)
```

which produces the following output in the MATLAB workspace:

```
%--------------- PC-Kriging metamodel ---------------%
Object Name: Model 2
Input Dimension: 1

Experimental Design
Sampling: Sobol
X size: [10x1]
Y size: [10x1]

Combination
Mode: sequential
```

```
Trend
Type: orthogonal polynomials
No. polys: 4

Gaussian Process
Corr. Handle: uq_eval_Kernel

Hyperparameters
theta: [ 0.09651 ]
Optim. method: Hybrid Genetic Algorithm

Leave-one-out error: 7.0028372e-02

%--------------------------------------------------%
```

Similarly, a visual representation of the PC-Kriging model can be obtained as follows:

```
uq_display(mySPCK)
```

which produces the image in Figure 1. Please note that such a representation is only available for 1D models.

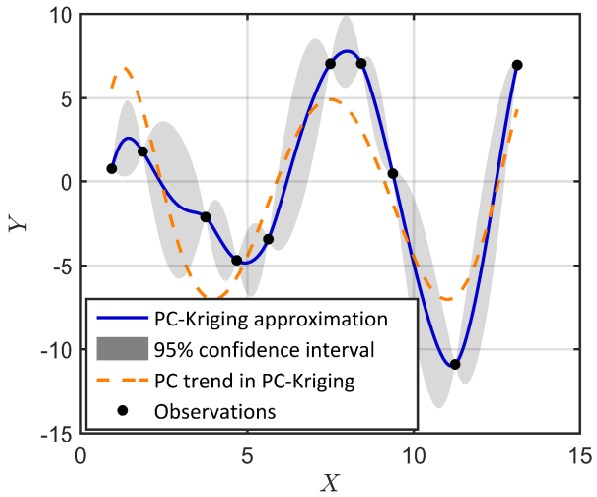

Figure 1: The figure created by `uq_display` of a SPCK-Kriging MODEL object which has a one-dimensional input.

Further results are stored in the object `mySPCK` that was created by UQLAB and the default configurations for SPCK. We can directly access this object in order to obtain various results and information on the metamodel. The content of PC-Kriging objects is summarized in Section 3.2.

2.3.2 Using PC-Kriging as a model (predictor)

Regardless of the specifications of the metamodel, a PC-Kriging model can be used to predict an output of new points in the input domain. Indeed, after a PC-Kriging MODEL object is created in UQLAB, it can be used like an ordinary model (for details, see the UQLAB User Manual – the MODEL module).
Consider the point $x = 3$, which is part of the input domain $\mathcal{D}_X = [0, 15]$. To evaluate the SPCK metamodel on the point x, one can write:

```
x = 3;
[ySPCK, ySPCKs2] = uq_evalModel(mySPCK, x);
```

where `ySPCK` and `ySPCKs2` are the prediction mean value and variance of the fitted Gaussian process at point $x = 3$, respectively. In this example, we obtain:

```
ySPCK = 0.1104
ySPCKs2 = 1.0576
```

2.4 Assessing the results

Apart from `uq_display` and `uq_print`, all results are accessible through the structure `mySPCK`:

```
mySPCK
  uq_model with properties:
          Name: 'Model 2'
      Internal: [1x1 struct]
           PCK: [1x1 struct]
    ExpDesign: [1x1 struct]
         Error: [1x1 struct]
       Options: [1x1 struct]
```

2.4.1 Experimental design

The experimental design used to calculate the PC-Kriging model can be accessed in `mySPCK.ExpDesign`:

```
mySPCK.ExpDesign
ans =
    Sampling: 'Sobol'
    NSamples: 10
    ED_Input: [1x1 uq_input]
        X: [10x1 double]
            Y: [10x1 double]
            U: [10x1 double]
```

Note that `mySPCK.ExpDesign.Sampling` field has the value `'Sobol'` due to the Sobol sequence used to generate the experimental design.

2.4.2 PCE-trend

As the trend of PC-Kriging is based on orthogonal polynomials, the associated PCE model can be accessed in the returned structure:

```
mySPCK.Internal.PCE
ans =
  uq_model with properties

         Name: 'Model 2'
     Internal: [1x1 struct]
         Type: 'uq_metamodel'
          PCE: [1x1 struct]
    ExpDesign: [1x1 struct]
        Error: [1x1 struct]
      Options: [1x1 struct]
```

Note that the experimental design field `mySPCK.Internal.PCE.ExpDesign` is identical to the one of the PC-Kriging model in field `mySPCK.ExpDesign`.

2.4.3 Gaussian process

The properties of the Gaussian process can be accessed via the field `mySPCK.PCK`. As an example, `mySPCK.PCK` includes the following fields:

```
mySPCK.PCK
ans =
       beta: [4x1 double]
    sigmaSQ: 1.3425
      theta: 0.0965
```

Note that `mySPCK.Internal.Kriging` is of type `uq_model` and can hence be used as such in prediction, too.

2.5 Advanced options

2.5.1 Optimal PC-Kriging model

An Optimal PC-Kriging model can be obtained by specifying the following option:

```
MetaOpts.Mode = 'optimal';
mySPCK = uq_createModel(MetaOpts);
```

The resulting metamodel can be summarized by `uq_print(myOPCK)` and visualized by `uq_display(myOPCK)`. Its usage is identical to that of the Sequential PCK.

2.5.2 PCE options

The trend of a PC-Kriging model is obtained by default by LARS and a polynomial degree $p = 3$. However, it is possible to change the options for LARS by specifying them in

`MetaOpts.PCE`. Every option related to LARS (described in UQLAB User Manual – Polynomial Chaos Expansions) may be set to a non-default value in the corresponding field.

In the above example, we considered a degree-adaptive LARS of degrees one to ten as follows:

```
MetaOpts.PCE.Degree = 1:10;
```

2.5.3 Custom set $\mathcal{A}$

By default, the trend of the PC-Kriging is obtained by LARS. However, if a suitable set of polynomials is available, it can be specified in `MetaOpts` by giving the polynomial types and the corresponding index set. Then, no LARS will be computed.

As an example, the polynomial types are set to Legendre polynomials and index set is $\alpha = (1, 4, 6)^{\mathsf{T}}$ (see also Section 2 in UQLAB User Manual – Polynomial Chaos Expansions):

```
MetaOpts.PolyTypes = {'Legendre'};
MetaOpts.PolyIndices = [1; 4; 6];
```

In case of multidimensional inputs `MetaOpts.PolyIndices` is a matrix of size $P \times M$ where P is the number of $\boldsymbol{\alpha}$-vectors and M is the size of the input vector.

2.5.4 Kriging options

By default, the Gaussian process of a PC-Kriging model is computed with the default Kriging settings. However, it is possible to change the options by specifying them in `MetaOpts.Kriging` with the same syntax as in a normal Kriging model (see also UQLAB User Manual – Kriging (Gaussian process modelling)). Every option related to the Kriging model (other than those associated to the trend) may the set manually.

As an example, we consider a separable Gaussian correlation function as follows:

```
MetaOpts.Kriging.Corr.Type = 'separable';
MetaOpts.Kriging.Corr.Family = 'Gaussian';
```

2.5.5 Use of a validation set

If a validation set is provided (see Table 3),UQLAB automatically computes the validation error given in Eq. (1.4). To provide a validation set, the following command shall be used:

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

The value of the validation error is stored in `myPCK.Error.Val` next to the other error measures (see Table 6) and will also be displayed when typing `uq_print(myPCK)`.

2.6 Vector-valued models

The examples presented so far in this chapter dealt with scalar-valued models. In case the model (or the experimental design, if manually specified) has multi-component outputs (denoted by N_{out}), UQLAB performs an independent PC-Kriging for each output component on the shared experimental design. No additional configuration is needed to enable this behaviour.

A PC-Kriging with multi-component outputs can be found in the UQLAB examples in: `Examples/PCK/uq_Example_03_MultipleOutputs.m`

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

```
myPCK.PCK
ans =
1x4 struct array with fields:
  beta
  sigmaSQ
  theta
```

Each element of the `myPCK` structure is functionally identical to its scalar counterpart in Section 2.4. Similarly, the `myPCE.Error` structure becomes a multi-element structure:

```
myPCK.Error
ans =
1x4 struct array with fields:
  LOO
```

2.7 Using PC-Kriging with constant parameters

In some analyses, one may need to assign a constant value to one or to a set of parameters. When this is the case, the PC-Kriging metamodel is built by internally removing the constant parameters from the inputs. This process is transparent to the users as they shall still evaluate the model using the full set of parameters (including those which were set constant). UQLAB will automatically and appropriately account for the set of input parameters which were declared constant.

To set a parameter to constant, the following command can be used (See UQLAB User Manual – the INPUT module):

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

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

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

Chapter 3

Reference List

How to read the reference list

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

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

which corresponds to the following syntax:

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

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

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

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

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

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

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

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

This is illustrated as follows:

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

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

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

Note: In the sequel, `double` and `doubles` mean a real number represented in double precision and a set of such real numbers, respectively.

3.1 Create a PC-Kriging metamodel

Syntax

```
myPCK = uq_createModel(MetaOpts)
```

Input

The struct variable `MetaOpts` contains the following fields:

Table 1: `MetaOpts`

●	`.Type`	`'Metamodel'`	Select the metamodelling tool
●	`.MetaType`	`'PCK'`	Select PC-Kriging
☐	`.Input`	INPUT object	Probabilistic input model
☐	`.Name`	String	Unique identifier for the metamodel
☐	`.Display`	String default: `'standard'`	Level of information displayed by the methods.
		`'quiet'`	Minimum display level, displays nothing or very few information.
		`'standard'`	Normal display level, shows the most important information.
		`'verbose'`	Maximum display level, shows all the information on runtime, like updates on iterations, etc.
☐	`.Mode`	String default: `'sequential'`	Combination strategy
		`'sequential'`	Computation of Sequential PC-Kriging model
		`'optimal'`	Computation of Optimal PC-Kriging model
☐	`.PCE`	Structure	Options related to the polynomial trend of PC-Kriging (see also UQLAB User Manual – Polynomial Chaos Expansions). By default `MetaOpts.PCE.Degree = 3`.
☐	`.PolyTypes`	$1 \times M$ Cell array of strings	List of polynomial families to be used to build the PCE basis and hence $\mathcal{A}$. If used, then also `.PolyIndices` is required.
☐	`.PolyIndices`	$P \times M$ Double	Set of multi-indices α to define the polynomial set $\mathcal{A}$. If used, then also `.PolyTypes` is required.

☐	`.Kriging`	Structure	Options related to the polynomial trend of PC-Kriging (see also UQLAB User Manual – Kriging (Gaussian process modelling))
☐	`.FullModel`	MODEL object	UQLab model used to create an experimental design (Section 2.3)
☐	`.ExpDesign`	Table 2	Experimental design-specific options (Section 2.3)
☐	`.ValidationSet`	Table 3	Validation set components (Section 2.5.5)

If a model is specified, UQLAB can automatically create an experimental design for PC-Kriging. The available options are listed in Table 2.

Table 2: `MetaOpts.ExpDesign`

⊕	`.Sampling`	String default: `'MC'`	Sampling type
		`'MC'`	Monte Carlo sampling
		`'LHS'`	Latin Hypercube sampling
		`'Sobol'`	Sobol sequence sampling
		`'Halton'`	Halton sequence sampling
☐	`.Nsamples`	Integer	The number of samples to draw. It is required when `.Sampling` is specified.
⊕	`.X`	$N \times M$ Double	User-defined experimental design X. If specified, `.Sampling` is ignored.
⊕	`.Y`	$N \times N_{Out}$ Double	User-defined model response Y. If specified, `.Sampling` is ignored.
⊕	`.DataFile`	String	`mat`-file containing the experimental design. If specified, `.Sampling` is ignored.

3.1.1 Validation Set

If a validation set is provided, UQLAB automatically calculates the validation error of the created PCK. The required information is listed in Table 3.

Table 3: `MetaOpts.ValidationSet`

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

3.2 Accessing the results

Syntax

```
myPCK = uq_createModel(MetaOpts) ;
```

Output

Regardless on the configuration options given at creation time in the `MetaOpts` structure, all PC-Kriging metamodels share the same output structure, given in Table 4. Note that because of the similarities of PC-Kriging and Kriging, many entries are the same as in the reference list of the UQLAB User Manual – Kriging (Gaussian process modelling) and hence some results are not explicitly elaborated here. For further details, the reader is referred to the UQLAB User Manual – Kriging (Gaussian process modelling), Chapter 3.

Table 4: `myPCK`

`.Name`	String	Unique name of the PC-Kriging metamodel
`.Options`	Table 1	Copy of the `MetaOpts` structure used to create the metamodel
`.PCK`	Table 5	Information about the final Kriging model
`.ExpDesign`	Table 7	Experimental design used for calculating the coefficients
`.Error`	Table 6	Error estimates of the metamodels's accuracy
`.Internal`	Table 8	Internal state of the MODEL object (useful for debug/diagnostics)

Table 5: `myPCK.PCK`

`.beta`	$P \times 1$ Double	The value of $\boldsymbol{\beta}(\boldsymbol{\theta})$ in Eq. (1.1)
`.sigmaSQ`	Double	The value of $\sigma^2(\boldsymbol{\theta})$ in Eq. (1.1)
`.theta`	Variable size double	The value of $\boldsymbol{\theta}$ in Eq. (1.1)

Table 6: `myPCK.Error`

`.LOO`	Double	The Leave-One-Out error
`.Val`	Double	Validation error (see Eq. (1.4) and Section 2.5.5). Only available if a validation set is provided (see Table 3).

Note: In general the fields `myPCK.PCK` and `myPCK.Error` are structure arrays with length equal to the number of outputs N_{out} of the metamodel.

Table 7: myPCK.ExpDesign		
.NSamples	Double	The number of samples
.Sampling	String	The sampling method
.ED_Input	INPUT object	The input module that was used to generate the experimental design (X)
.X	$N \times M$ Double	The experimental design values
.U	$N \times M$ Double	The experimental design values in the reduced space
.Y	$N \times N_{out}$ Double	The output Y that corresponds to the input X

Table 8: myPCK.Internal		
.Runtime	Structure	Variables that are used during the calculation of the PC-Kriging metamodel
.Error	Structure	Internal fields related to the error of the Kriging part of PC-Kriging (see also UQLAB User Manual – Kriging (Gaussian process modelling))
.Kriging	MODEL object	Kriging model generated with a PCE trend (see also UQLAB User Manual – Kriging (Gaussian process modelling))
.PCE	MODEL object	The PCE model generated to model the trend of the PC-Kriging model. More details can be found in UQLAB User Manual – Polynomial Chaos Expansions
.NumberOfPoly	Integer	Number of polynomials in the trend of the PC-Kriging model
.AuxSpace	INPUT object	Auxiliary space where PC-Kriging is performed
.TrendMethod	String	Method that defines the trend, either 'pce' or 'user'

Note: The internal fields of the PC-Kriging module are not intended to be accessed or changed for most typical usage scenarios of the module.

References

Blatman, G. and B. Sudret (2011). Adaptive sparse polynomial chaos expansion based on Least Angle Regression. *Journal of Computational Physics 230*, 2345–2367. 3

Kersaudy, P., B. Sudret, N. Varsier, O. Picon, and J. Wiart (2015). A new surrogate modeling technique combining Kriging and polynomial chaos expansions – Application to uncertainty analysis in computational dosimetry. *Journal of Computational Physics 286*, 103–117. 2

Lataniotis, C., D. Wicaksono, S. Marelli, and B. Sudret (2021). UQLab user manual – Kriging. Technical report, Chair of Risk, Safety & Uncertainty Quantification, ETH Zurich. Report # UQLab-V1.4-105. 1

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

Schöbi, R., B. Sudret, and S. Marelli (2016). Rare event estimation using Polynomial-Chaos-Kriging. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering*, D4016002. 2

Schöbi, R., B. Sudret, and J. Wiart (2015). Polynomial-chaos-based Kriging. *International Journal for Uncertainty Quantification 5*(2), 171–193. 2

Sudret, B. (2007). Uncertainty propagation and sensitivity analysis in mechanical models - Contributions to structural reliability and stochastic spectral methods. Habilitation thesis, Université Blaise Pascal, Clermont-Ferrand, France. 2

Xiu, D. and G. E. Karniadakis (2002). The Wiener-Askey polynomial chaos for stochastic differential equations. *SIAM Journal of Scientific Computing 24*(2), 619–644. 2