ETH zürich

UQLab User Manual
Stochastic Polynomial Chaos Expansions (SPCE)

N. Lüthen, X. Zhu, 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

N. Lüthen, X. Zhu, S. Marelli, B. Sudret, UQLab user manual – Stochastic polynomial chaos expansions (SPCE), Report UQLab-V2.1-121, Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland, 2024

BibTeX entry

```
@TechReport{UQdoc_21_121,
author = {Lüthen, N. and Zhu, X. and Marelli, S. and Sudret, B.},
title = {{UQLab user manual -- Stochastic PCE}},
institution = {Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich,
Switzerland},
year = {2024},
note = {Report UQLab-V2.1-121}
}
```

Document Data Sheet

Document Ref.	UQLAB-V2.1-121
Title:	UQLAB user manual – Stochastic polynomial chaos expansions (SPCE)
Authors:	N. Lüthen, X. Zhu, S. Marelli, B. Sudret Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland
Date:	15/04/2024

Doc. Version	Date	Comments
V2.1	15/04/2024	UQLAB V2.1 release (first version)

Abstract

Stochastic polynomial chaos expansions are a novel metamodelling technique for stochastic simulators, *i.e.* for computational models whose response to a given set of input parameters is not a deterministic value, but a realization from a random variable with unknown probability distribution. SPCE is a surrogate designed to efficiently represent the probability distribution of the model output for any given input values. The stochasticity is reproduced by introducing a latent variable and an additional noise variable on top of the well-defined input variables. The surrogate consists of a PCE built on this augmented input space.
It can be built from experimental designs comprising replicated model evaluations as well as from replication-free experimental designs. SPCE can accurately represent general response distributions, *i.e.* not only normal or unimodal ones.
The UQLAB SPCE module offers an easy way to construct SPCE surrogate models based on replicated and replication-free experimental designs. It makes use of other UQLAB modules (UQLAB User Manual – the INPUT module, UQLAB User Manual – the MODEL module) to define the input distribution and the stochastic model. It also relies on the UQLAB PCE-module (UQLAB User Manual – Polynomial Chaos Expansions) for computing the metamodel.
The manual for the SPCE metamodelling module is divided into three parts:

- A introduction to the main ideas and theoretical foundations of stochastic PCE;
- An example-based guide to SPCE with an explanation of available options and methods;
- A comprehensive reference list detailing all available functionalities of the SPCE module.

Keywords: UQLAB, metamodelling, stochastic model, polynomial chaos expansion

Contents

Chapter 1

Theory

1.1 Introduction

In uncertainty quantification, two classes of models can be distinguished. For *deterministic models*, the uncertainty in the output originates exclusively from the uncertainty in the input variables. If the values of the inputs are held fixed, the output is fixed, too. On the other hand, there are models whose output is random, even if all input variables are set to fixed values. In other words, for each vector of input parameters, a model evaluation is a realization from an associated random variable. Such models are called *stochastic simulators*.
Stochastic polynomial chaos expansions (SPCE) are a surrogate modelling technique developed for the case of stochastic simulators (Zhu and Sudret, 2023). Its aim is to represent the probability distribution of the random variable response for any vector of input parameters. It uses the powerful methodology of polynomial chaos expansion (PCE) to build the metamodel.
In Section 1.2 we review the components of the method, before we describe SPCE in detail in Section 1.3.

1.2 Basics

1.2.1 Stochastic simulators

A computational model can be seen as a black box, *i.e.* as a map from the space of input parameters to that of output quantities:

$$\boldsymbol{y} = \mathcal{M}(\boldsymbol{x}) \tag{1.1}$$

where $\boldsymbol{x}$ is a realization of the random vector $\boldsymbol{X}$ that parametrizes the variability of the input parameters (typically through a joint probability density function (PDF)), and $\boldsymbol{y}$ is the vector of model responses.
We identify two classes of models:

- *deterministic* models, which will always give the same result $\boldsymbol{y}_0 = \mathcal{M}(\boldsymbol{x}_0)$ when evaluated repeatedly for a given input realization $\boldsymbol{x}_0$.

- *stochastic* models, which for a given input realization $\boldsymbol{x}_0$ return an output random variable $\boldsymbol{Y}_0 = \mathcal{M}(\boldsymbol{x}_0)$ with distribution $f_{\boldsymbol{Y}_0}$. Every time a stochastic model is evaluated, a realization $\boldsymbol{y}_0 \sim \boldsymbol{Y}_0$ is generated. A model from the this class is also called *stochastic simulator*.

Considering the randomness in the input variables, the output of a stochastic simulator is a random variable. The randomness of the latter comes from both the intrinsic stochasticity and the uncertain inputs. When fixing the input parameters, the model response remains random. For the purpose of clarity, we denote by $Y_{\boldsymbol{x}}$ the random model response for the input parameters $\boldsymbol{x}$ and by Y the model output containing all the uncertainties:

$$Y_{\boldsymbol{x}} \overset{\text{def}}{=} \mathcal{M}(\boldsymbol{x}, \omega), \quad Y \overset{\text{def}}{=} \mathcal{M}(\boldsymbol{X}(\omega), \omega). \tag{1.2}$$

Here, we have made the dependence of the stochastic simulator on the random event ω explicit.

For a detailed description of how stochastic simulators are treated and can be implemented in UQLAB, the reader is referred to the UQLAB User Manual – the MODEL module.

Replications

For stochastic simulators, the output $\boldsymbol{Y}_0 = \mathcal{M}(\boldsymbol{x}_0)$ is a random variable. Evaluating the model several times for the same input vector $\boldsymbol{x}_0$ generates several independent realizations $\boldsymbol{y}_0^{(1)}, \ldots, \boldsymbol{y}_0^{(n)} \sim \boldsymbol{Y}_0$, called *replications*.

Replications can be used to characterize the distribution $f_{\boldsymbol{Y}_0}$ of the response random variable at $\boldsymbol{x}_0$. A set of model evaluations that contains replications for all points of the experimental design is called *replication-based*.

Conversely, we call a dataset that contains only a single realization of the response random for each input realization *replication-free*.

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):

$$Y \approx \mathcal{M}^{PC} = \sum_{\boldsymbol{\alpha} \in \mathcal{A}} c_{\boldsymbol{\alpha}} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{X}), \tag{1.3}$$

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

1.3 Stochastic PCE (SPCE)

1.3.1 Formulation

From a probabilistic perspective, $Y_{\boldsymbol{x}}$ can be seen as a conditional random variable $Y \mid \boldsymbol{X} = \boldsymbol{x}$. Let $F_{Y|\boldsymbol{X}}\left(y \,|\, \boldsymbol{x}\right)$ denote the associated cumulative distribution function (CDF). By using the probability integral transform, we can transform *any* continuous random variable Z to the desired distribution, that is

$$Y_{\boldsymbol{x}} \overset{\mathrm{d}}{=} F^{-1}_{Y|\boldsymbol{X}}\left(F_Z\left(Z\right) | \, \boldsymbol{x}\right) \tag{1.4}$$

where F_Z is the CDF of Z. The equality in Eq. (1.4) is to be understood *in distribution*, meaning that the two random variables on the left- and right-hand side follow the same distribution. In Eq. (1.4), the right-hand side is a deterministic function of both $\boldsymbol{x}$ and z. As a result, assuming that Y has a finite variance, we can achieve this equality in distribution thanks to a PCE in the $(\boldsymbol{X}, Z)$ space, that is,

$$F^{-1}_{Y|\boldsymbol{X}}\left(F_Z\left(Z\right) | \, \boldsymbol{X}\right) \overset{\mathrm{d}}{=} \sum_{\boldsymbol{\alpha} \in \mathbb{N}^{M+1}} c_{\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}\left(\boldsymbol{X}, Z\right). \tag{1.5}$$

To improve numerical stability in the training steps (see Sections 1.3.2 and 1.3.3), we also introduce an additive noise variable ϵ, and define the stochastic surrogate (with truncation set $\mathcal{A} \subset \mathbb{N}^{M+1}$, see also UQLAB User Manual – Polynomial Chaos Expansions, Section 1.3.3.2) as follows:

$$Y_{\boldsymbol{x}} \overset{\mathrm{d}}{\approx} \tilde{Y}_{\boldsymbol{x}} = \sum_{\boldsymbol{\alpha} \in \mathcal{A}} c_{\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}\left(\boldsymbol{x}, Z\right) + \epsilon, \tag{1.6}$$

where ϵ is a centered Gaussian random variable with standard deviation σ, i.e. $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

To construct the stochastic PCE defined in Eq. (1.6), one needs to estimate both the coefficients $\boldsymbol{c}$ and the standard deviation σ of the noise variable.

1.3.2 Likelihood function

Based on the available data $(\mathcal{X}, \boldsymbol{y})$, the PCE coefficients $\boldsymbol{c}$ can be fitted using the maximum likelihood estimation (MLE)

$$\hat{\boldsymbol{c}} = \arg\max_{\boldsymbol{c}} \sum_{i}^{N} \log\left(\tilde{l}\left(\boldsymbol{c}; \boldsymbol{x}^{(i)}, y^{(i)}, \sigma\right)\right) \tag{1.7}$$

where the likelihood function is given by

$$l(\boldsymbol{c}; \boldsymbol{x}, y, \sigma) = \int_{\mathcal{D}_Z} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(y - \sum_{\boldsymbol{\alpha} \in \mathcal{A}} c_{\boldsymbol{\alpha}} \psi_{\boldsymbol{\alpha}}(\boldsymbol{x}, z)\right)^2}{2\sigma^2}\right) f_Z(z) \mathrm{d}z, \tag{1.8}$$

and $\tilde{l}$ is the numerical approximation to this one-dimensional integral.

1.3.3 Estimation of the SPCE coefficients

The gradient of the likelihood in Eq. (1.7) can be calculated analytically. Therefore, the optimization problem is solved using the derivative-based BFGS quasi-Newton method.

Starting point. The objective function to optimize in Eq. (1.7) is highly nonlinear. As a result, a good starting point is necessary to ensure convergence. The mean function $\tilde{m}(\boldsymbol{x}) \overset{\text{def}}{=} \mathbb{E}\left[\tilde{Y}_{\boldsymbol{x}}\right]$ of the stochastic simulator is approximated by sparse regression (LARS), which yields a starting point for all coefficients *not* associated to the latent variable Z. The remaining coefficients are initialized randomly.
The mean fitting by sparse regression has two more functions. First, it results in an estimate ε_{LOO} for the mean-squared approximation error. This value is then used to determine a meaningful range for σ (see below). Second, the sparse expansion of the mean is used to exclude basis functions from the SPCE, thereby reducing the dimensionality of $\boldsymbol{c}$: among the basis functions *not* associated to the latent variable, only those with a non-zero coefficient are retained for further calculations.

Iterative estimation of $\boldsymbol{c}$. Because of the form of the likelihood in Eq. (1.8), the gradient at the starting point can take extremely large values if σ is small. Therefore, the coefficients $\boldsymbol{c}$ are computed using an iterative algorithm ("warm-start strategy"): a decreasing sequence $\boldsymbol{\sigma} = \{\sigma_1, \ldots, \sigma_{N_s}\}$ is generated, where $\sigma_1 = \sqrt{\varepsilon_{\text{LOO}}}$, $\sigma_{N_s} = \sigma$ (the target value), and the remaining values are generated equally-spaced in the log-space between σ_1 and σ_{N_s}.
Starting with σ_1, a stochastic PCE is built by solving Eq. (1.7) with the starting point as defined above. Then, the results are used as a starting point for the construction of the surrogate for σ_2. This procedure is repeated for each element in $\boldsymbol{\sigma}$ with each new starting point being the results of the previous optimization.

1.3.4 Determining the optimal σ

To avoid numerical issues that may lead to an infinite likelihood in Eq (1.8), N_{cv}-fold cross-validation (CV) is applied to select the optimal value of σ, *i.e.* the one which maximizes the CV score of (1.7). This maximizer is computed using derivative-free Bayesian optimization. The search interval for σ is $[0.1, 1] \times \sqrt{\varepsilon_{\text{LOO}}}$. The optimal $\hat{\sigma}$ is used in (1.7) with all the available data to build the final surrogate.
For more details, the reader is referred to Zhu and Sudret (2023).

1.3.5 Adaptivity

The methodology described above allows to build a stochastic PCE for a given distribution of the latent variable Z and truncated set $\mathcal{A}$ of polynomial chaos basis. For choosing the best latent distribution type and truncation set from a number of possible options, the cross-validation (CV) score is used. For the distribution type, the choice is typically between standard uniform and standard Gaussian. The best truncation set is determined by degree- and

q-norm adaptivity with Early-Stop criterion (see UQLAB User Manual – Polynomial Chaos Expansions). For each combination of distribution type and truncation set, the optimal $\hat{\sigma}$ is determined as described in Section 1.3.4.

1.3.6 Post-processing

Due to the form of SPCE, mean and variance of the model response $\tilde{Y}_{\boldsymbol{x}}$ can be computed analytically from the PCE. Furthermore, several types of Sobol' indices can be obtained by analytical postprocessing of the PCE coefficients. For more details, the reader is referred to Zhu and Sudret (2023).

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: Geometric Brownian motion

Some models are not deterministic but *stochastic*: they generate a different output value every time they are run. An example is the Geometric Brownian Motion (GBM), which is defined by the stochastic differential equation

$$dS_t(\boldsymbol{x}) = x_1 S_t + x_2 \, dW_t, \tag{2.1}$$

with the boundary condition $S_0(\boldsymbol{x}) = 1$. Here, W_t is a Wiener process, and x_1 and x_2 are parameters of the equation. The model we consider simulates the value of a GBM at time 1, *i.e.* $Y_{\boldsymbol{x}} = S_1(\boldsymbol{x})$. Analytical calculations show that for every $\boldsymbol{x}$, the model response $Y_{\boldsymbol{x}}$ follows a lognormal distribution

$$Y_{\boldsymbol{x}} \sim \mathcal{LN}(x_1 - x_2^2/2, x_2). \tag{2.2}$$

This model is available in UQLAB under the name `Y = uq_GBM(X)`.
In this example, the two parameters x_1 and x_2 are considered random variables $X_1 \sim \mathcal{U}(0, 0.1), X_2 \sim \mathcal{U}(0.1, 0.4)$.

2.2 Problem setup

The UQLAB framework is first initialized with the following command:

```
uqlab
```

2.2.1 Full model and probabilistic input model

To surrogate the GBM model using UQLAB, we need to first define a basic stochastic MODEL object:

```
ModelOpts.mFile = 'uq_GBM';
ModelOpts.isStochastic = true;
ModelOpts.isVectorized = true;
myModel = uq_createModel(modelopts);
```

Setting the flag `isStochastic` to `true` causes UQLAB treat this model as a stochastic simulator.

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

Due to the internal use of PCE, a SPCE model *always* requires an input model, even when using a given experimental design. The INPUT object is defined by:

```
InputOpts.Marginals(1).Type = 'Uniform';
InputOpts.Marginals(1).Parameters = [0, 0.1];
InputOpts.Marginals(2).Type = 'Uniform';
InputOpts.Marginals(2).Parameters = [0.1, 0.4];

myInput = uq_createInput(InputOpt);
```

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 SPCE metamodel

The SPCE module creates a MODEL object that can be used as any other stochastic model.

A SPCE with default options is created as follows:

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

The experimental design is sampled with Latin hypercube sampling:

```
MetaOpts.ExpDesign.NSamples = 800;
MetaOpts.ExpDesign.Sampling = 'LHS';
```

The surrogate is created by

```
mySPCE = uq_createModel(MetaOpts);
```

There is a variety of optional configuration options, which are explained in Section 2.5. For them to be considered, they must be set before the SPCE metamodel is created.

2.4 Accessing the results

Once the model is created, a report with basic information about the SPCE metamodel can be printed as follows:

```
>> uq_print(mySPCE)
```

which produces the following output in the MATLAB command window:

```
%------------ Stochastic polynomial chaos expansion output ---------------%
   Number of input variables:            2
   Latent variable: Gaussian distribution with parameters [ 0.00 1.00 ]
   Maximal degree:                       3
   Size of basis:                        12
   Full model evaluations:               800 ED points with 1 replications
   k-fold CV negative loglikelihood:     -7.8310676e+02
   AIC:                                  -3.0529900e+01
   BIC:                                  2.5685441e+01
   Mean value:                           1.0590
   Standard deviation:                   0.2752
   Coef. of variation:                   25.988%
%-----------------------------------------------------------------------%
```

Similarly, a visual representation of the SPCE can be obtained as follows:

```
uq_display(mySPCE)
```

which plots the the mean and the the area plus/minus 2 standard deviations (for 1-dimensional input) or the mean and variance (for 2-dimensional input) of the SPCE predictor as shown in Figure 1.

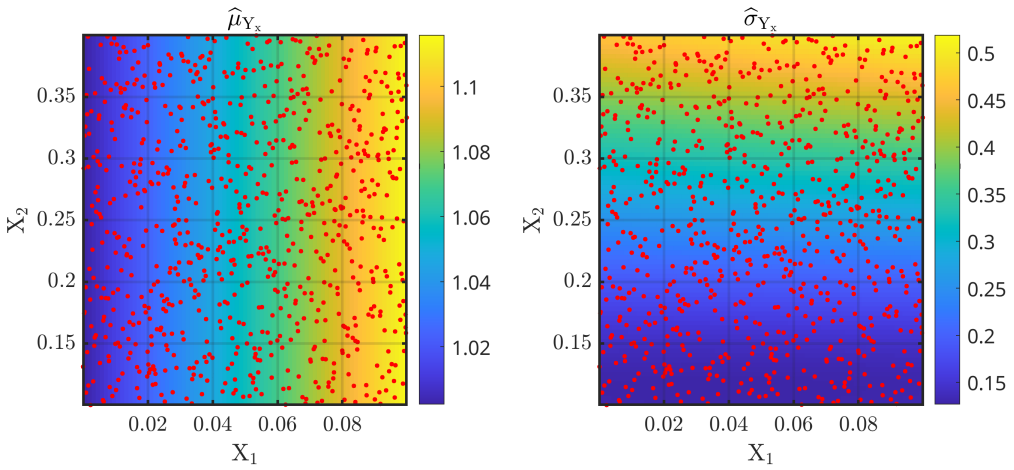

Figure 1: The figure created by `uq_display` of an SPCE MODEL object which has a two-dimensional input. The left panel reports the evolution of the mean value of the stochastic emulator, while the right panel shows its standard deviation. The red dots represent the experimental design points.

More options for `uq_display` as well as three additional useful postprocessing and display functions are listed in Section 3.3.

Further results are stored in the object `mySPCE` that was created by UQLAB. We can directly access this object in order to obtain various results and information on the metamodel:

```
>> mySPCE
  uq_model with properties:

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

```
            SPCE: [1x4 struct]
    isStochastic: 1
         Options: [1x1 struct]
           Error: [1x1 struct]
```

A complete list of results is provided in Section 3.2.

2.4.1 Error

To assess the goodness-of-fit of the calculated metamodel, UQLAB provides several error measures. These include the negative log-likelihood, *i.e.* the value of (1.7) after optimization, as well as various model selection criteria (namely AIC and BIC) that also take the number of free parameters into account. For a review of these model selection criteria, the reader is referred to Ye et al. (2008).
If a validation set is provided (2.5.5), the field `.Val` contains several validation error metrics, as detailed in Section 3.3.2.

2.4.2 Experimental design

The field `mySPCE.ExpDesign` contains all information about the experimental design, whether it was provided by the user or sampled by UQLAB. This includes information such as the number of samples, the number of replications, and the input samples as well as the corresponding model evaluations.

2.5 Advanced options

2.5.1 Latent variables

The types of latent variables (see Section 1.3.5) to choose from can be specified as follows:

```
MetaOpts.Latent(1).Type = 'Gaussian';
MetaOpts.Latent(2).Type = 'Uniform';
MetaOpts.Latent(3).Type = 'Beta';
MetaOpts.Latent(3).Parameters = [4,6];
```

Here, three possible distributions are specified: standard Gaussian $\mathcal{N}(0,1)$, standard uniform $\mathcal{U}(-1,1)$, and Beta $\mathcal{B}(4,6)$ (on $[0,1]$). If the field `.Parameters` is not specified for Gaussian or uniform distributions, the standard parameters are assumed. For all other distribution types, the parameters must be specified.

2.5.2 PCE options

As introduced in Section 1.3.3, constructing an SPCE model consists of two steps: the mean function is fitted by sparse regression. The coefficients of the latent variable are determined in a separate step by maximization of Eq. (1.7).

2.5.2.1 Basis adaptivity for the overall coefficient estimation

Basis adaptivity for the latter step can be enabled as follows:

```
MetaOpts.Degree = 1:3;
MetaOpts.TruncOptions.qNorm = [0.5,0.75,1];
```

An SPCE model is computed for every combination of degree and q-norm, and the SPCE with the largest CV score of the associated likelihood (Eq. (1.7)) is chosen as the final metamodel. For both degree and q-norm adaptivity, UQLAB implements a so-called early stop criterion: if the CV score decreases twice in a row (for q-norm adaptivity) or once (for degree adaptivity), respectively, the adaptivity iteration is stopped.
Albeit it is not recommended, as it can cause overfitting, as well as greatly increase training times, the early stop criterion for the coefficient calculation can be turned off by setting the following flags to `false`:

```
MetaOpts.DegreeEarlyStop = false; % default: true
MetaOpts.qNormEarlyStop = false; % default: true
```

2.5.2.2 PCE options for the mean estimation

By default, the mean value is fitted independently of the rest of the parameters in an initial step (see Section 2.5.2.2). Its options are the same as those available for PCE, documented in the UQLAB User Manual – Polynomial Chaos Expansions.
In particular, the type of sparse solver and the settings for degree and q-norm adaptivity can be set as follows:

```
MetaOpts.MeanReg.Method = 'LARS';
MetaOpts.MeanReg.Degree = 0:5;
MetaOpts.MeanReg.TruncOptions.qNorm = 0.4:0.2:1;
```

Basis adaptivity is performed based on the leave-one-out (LOO) cross-validation error with an early-stop criterion as described in UQLAB User Manual – Polynomial Chaos Expansions. If no values are provided for degree and q-norm, the global values from `MetaOpts.Degree` and `MetaOpts.TruncOptions.qNorm` are used.

2.5.3 Integration options

To calculate the likelihood in Eq. (1.8), a 1-dimensional integral needs to be solved efficiently. This is needed both for the overall coefficient estimation as well as for the evaluation of the cross-validation score (see Section 1.3.4). The integration strategies for the two purposes can be set separately. The latter typically needs a more precise estimate of Eq. (1.8) than the former.
For the coefficient estimation, the integration strategy can be configured as:

```
MetaOpts.IntMethod = 'Quadrature';
MetaOpts.Quadrature.Level = 50; % default: 100
```

More integration options can be found in Table 4.
For the evaluation of the CV score, which is used to determine the optimal σ, integration options can be set similarly as:

```
MetaOpts.Sigma.IntMethod = 'Quadrature';
MetaOpts.Sigma.Quadrature.Level = 200; % default: 1000
```

More options for the estimation of σ can be found in Table 6.

2.5.4 User-defined experimental design

Instead of letting UQLAB create an experimental design, it is also possible to provide a data set using the following syntax:

```
MetaOpts.ExpDesign.X = X_data;
MetaOpts.ExpDesign.Y = Y_data; % corresponding model evaluations
```

If an experimental design is provided in this way, the fields `.Sampling`, `.Nsamples`, and `Replications` are ignored. Note that it is still necessary to specify an INPUT object, because PCE requires it.
The format of `Y_data` follows the convention of stochastic simulators introduced in the UQLAB User Manual – the MODEL module, *i.e.* a collection of N model responses with N_{Out} model outputs and N_R replications must be provided as a 3-dimensional matrix of dimensions $N \times N_{\text{Out}} \times N_R$.
Note that it is still necessary to specify an INPUT object, because it is needed co construct the PCE basis.

2.5.5 Use of a validation set

A validation set can be provided as follows:

```
MetaOpts.ValidationSet.X = Xval;
MetaOpts.ValidationSet.Y = Yval;
```

UQLAB calculates various validation error metrics based on this data set, which are stored in `mySPCE.Error.Val`.

2.5.6 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 computes an independent SPCE for each output component on the shared experimental design. Please note that SPCE cannot capture the covariance/interaction between different outputs, and they are treated entirely independently. No additional configuration is needed to enable this behaviour.
A SPCE with multi-component outputs can be found in the UQLAB examples in:
`Examples/SPCE/uq_Example_SPCE_04_MultiOutputs.m`

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

```
>> mySPCE.SPCE

ans =
  1x2 struct array with fields:
    Latent
    Basis
    Coefficients
    Sigma
    Moments
```

Its fields are described in Table 11.

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

```
>> mySPCE.Error

ans =
  1x2 struct array with fields:
  CV
  Nloglikelihood
  BIC
  AIC
```

2.5.7 Using SPCE as a model (predictor)

Every SPCE metamodel computed by UQLAB is a MODEL object. It can be evaluated on points from the input domain just like any stochastic simulator implemented in UQLAB.

For example, we obtain an evaluation for a new sample point `X` by

```
X = uq_getSample(1);
Y_SPCE = uq_evalModel(mySPCE, X);
```

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 SPCE metamodel

Syntax

```
mySPCE = uq_createModel(MetaOpts)
```

Input

The struct variable `MetaOpts` contains the following fields:

Table 1: `MetaOpts`

●	`.Type`	`'Metamodel'`	Select the metamodelling tool
●	`.MetaType`	`'SPCE'`	Select the generalized lambda model
☐	`.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.
☐	`.FullModel`	MODEL object	UQLab model to be surrogated
☐	`.Latent`	Struct array, see Table 3	Properties of the latent variable
☐	`.IntMethod`	String default: `'Quadrature'`	Integration method for fitting
		`'Quadrature'`	Gaussian quadrature
		`'Sampling'`	Sampling-based integration
		`'MatlabInt'`	MATLAB built-in integrator
⊞	`.Quadrature`	Table 4	Options for the integration method `Quadrature`
⊞	`.Sampling`	Table 5	Options for the quadrature method `Sampling`
☐	`.Sigma`	Table 6	Properties of sigma (noise level)
☐	`.MeanReg`	Table 7	Options for the mean estimation by sparse regression
●	`.ExpDesign`	Table 8	Experimental design-specific options
☐	`.ValidationSet`	Table 9	Validation set components

☐	`.Degree`	Integer array default: `1:5`	Degree of the PCE (adaptive if array is specified)
		Integer scalar	Maximum polynomial degree
		Integer array	Set of polynomial degrees for degree-adaptive polynomial chaos (see UQLAB User Manual – Polynomial Chaos Expansions)
☐	`.TruncOptions`	Struct, see Table 2	Truncation options for the PCE
☐	`.DegreeEarlyStop`	Logical default: true	Whether or not to use the option "DegreeEarlyStop" (see UQLAB User Manual – Polynomial Chaos Expansions)
☐	`.qNormEarlyStop`	Logical default: true	Whether or not to use the option "qNormEarlyStop" (see UQLAB User Manual – Polynomial Chaos Expansions)
☐	`.SelectCrit`	String default: `'CV'`	Model selection criterion (for degree and qNorm adaptivity)
☐	`.Standardize`	logical default: true	Whether the data should be standardized

Note: If `.FullModel` or `.Input` are not specified, the currently active model or input is used. By default, the *last created* model or surrogate model is the currently active model.

3.1.1 Truncation options

Table 2: `MetaOpts.TruncOptions`

☐	`.qNorm`	Double default: 1	Set of hyperbolic norms for q-norm-adaptive polynomial chaos (see UQLAB User Manual – Polynomial Chaos Expansions).
☐	`.MaxInteraction`	Integer default: M	Maximum rank truncation
☐	`.Custom`	$P \times M$ integer array	Manual specification of the multi-index set $\mathcal{A}$

3.1.2 Latent options

The field `.Latent` is a struct array containing different choices for the distribution of latent variables. The syntax is the same as when specifying the marginals for the INPUT module (see UQLAB User Manual – the INPUT module).

Table 3: `MetaOpts.Latent`

●	`.Type`	String	Distribution family
☐	`.Parameters`	Double array	Parameters of the distribution. If not specified for type `'Uniform'` or `'Gaussian'`, the arrays `[-1,1]` or `[0,1]` are used, respectively.

By default, UQLAB chooses between the following two distributions:

```
Latent(1).Type = 'Gaussian';
Latent(1).Parameters = [0,1];
Latent(2).Type = 'Uniform';
Latent(2).Parameters = [-1,1];
```

3.1.3 Integration options

To numerically evaluate Eq. (1.8), the three methods `'Quadrature'`, `'Sampling'`, and `'MatlabInt'` are available. For the two former methods, additional options can be specified.

Table 4: `MetaOpts.Quadrature`

☐	`.Level`	Integer default: 100	Number of quadrature nodes

Table 5: `MetaOpts.Sampling`

☐	`.N`	Integer default: 1,000	Number of function calls
☐	`.Method`	String default: 'LHS'	Sampling method

3.1.4 Sigma options

Table 6: `MetaOpts.Sigma`

☐	`.IntMethod`	String default: `'Quadrature'`	Integration method for fitting
		`'Quadrature'`	Gauss quadrature
		`'Sampling'`	Monte Carlo integration
		`'MatlabInt'`	MATLAB's built-in integration routine
⊞	`.Quadrature`	Table 4	Options for the integration method `Quadrature` (Gaussian quadrature)
⊞	`.Sampling`	Table 5	Options for the quadrature method `Sampling`

☐	`.Value`	double or double array	Manually specify a value, or a set of candidate values for σ

3.1.5 MeanReg options

Table 7: `MetaOpts.MeanReg`			
☐	`.separateFit`	logical default: true	If set to true (the default behavior), the degree and q-norm adaptivity for the estimation of the mean value (see Section 1.3.3) are performed separately from the rest of the coefficients for the full SPCE.
☐	`.Method`	String default: `'LARS'`	Regression method for the expansion of the mean
☐	`.Degree`		Degree of the adaptive expansion of the mean
☐	`.TruncOptions`	See Table 2	Truncation options for the mean expansion

3.1.6 Experimental design

Table 8: `MetaOpts.ExpDesign`			
⊕	`.Sampling`	String default: `'LHS'`	Sampling type for points in the input space
		`'MC'`	Monte Carlo sampling
		`'LHS'`	Latin Hypercube sampling
☐	`.NSamples`	Integer	The number of samples to draw. It is required when `.Sampling` is specified.
☐	`.Replications`	Integer	The number of replications.
⊕	`.X`	$N \times M$ Double	User-defined experimental design X. If specified, `.Sampling` is ignored.
⊕	`.Y`	$N \times N_{\text{out}} \times R$ Double	User-defined model response Y with R replications. If specified, `.Sampling` is ignored.

3.1.7 Validation Set

Table 9: `MetaOpts.ValidationSet`			
●	`.X`	$N \times M$ double	User-specified validation set $\boldsymbol{\mathcal{X}}_{\text{Val}}$
●	`.Y`	$N \times N_{\text{out}} \times R$ double	User-specified validation set response $\mathcal{Y}_{\text{Val}}$ with replications

3.2 Accessing the results

Syntax

```
mySPCE = uq_createModel(MetaOpts);
```

Output

Regardless on the configuration options given at creation time in the `MetaOpts` structure, all SPCE metamodels share the same output structure, given in Table 10.

Table 10: `mySPCE`		
`.Name`	String	Unique name of the SPCE metamodel
`.Options`	Table 1	Copy of the `MetaOpts` structure used to create the metamodel
`.SPCE`	$1 \times N_{\text{out}}$ struct array with fields detailed in Table 11	Information about the final SPCE model
`.ExpDesign`	Table 13	Experimental design used for calculating the coefficients
`.Error`	$1 \times N_{\text{out}}$ struct array with fields detailed in Table 12	Error estimates of the metamodels's accuracy
`.Internal`	Table 14	Internal state of the MODEL object (useful for debug/diagnostics)

The field `mySPCE.SPCE` contains a $1 \times N_{\text{out}}$ struct array defining the SPCEs for each of the N_{out} outputs of the computational model.

Table 11: `mySPCE.SPCE`		
`.Latent`	Struct	Information about the latent variable of the SPCE
`.Basis`	Struct array	Basis information for the SPCE, see UQLAB User Manual – Polynomial Chaos Expansions
`.Coefficients`	Double array	Coefficient vector of the SPCE
`.Sigma`	Double	
`.Moments`	Struct	Mean and variance of the SPCE

Table 12: `mySPCE.Error`		
`.CV`	Double	Cross-validation score
`.NLogLikelihood`	Double	Negative log-likelihood

.AIC	Double	Akaike information criterion
.BIC	Double	Bayesian information criterion
.Val	Struct	Error metrics between the SPCE surrogate and the provided validation set, computed by uq_evalStochSimMetrics (see Section 3.3.2)

Table 13: mySPCE.ExpDesign

.NSamples	Integer	The number of samples
.Sampling	String	The sampling method
.ED_Input	INPUT object	The input module that was used to generate the experimental design
.Replications	Integer	Number of replications
.isTrajectory	Logical	Whether or not the experimental design consists of trajectories
.X	$N \times M$ double	The experimental design values
.U	$N \times M$ double	The experimental design values in the reduced space
.Y	$N \times N_{out} \times R$ double	The output Y that corresponds to the input X, with R replications

The following table contains a selection of fields from mySPCE.Internal that might be of interest for the user.

Table 14: mySPCE.Internal

.Input	INPUT object	The input module that was used to generate the experimental design
.FullModel	MODEL object	The model object
.Method	String	Method used for fitting the coefficients
.SPCE	Struct	Further options for the SPCE calculation

3.3 Additional functions

3.3.1 uq_display

Syntax

```
uq_display(SPCEModel)
uq_display(SPCEModel, OUTARRAY)
```

Description

Note: This display function only works for SPCE models with 1- and 2-dimensional inputs.

`uq_display(SPCEodel)` plots the median and the 2.5%-97.5% quantile (for 1-dimensional input) or the mean and variance (for 2-dimensional input) of a SPCE predictor specified by `SPCEodel`. If there is more than one output, only the first output component is plotted.

`uq_display(SPCEodel, OUTARRAY)` creates the plots of a SPCE predictor with multiple outputs for the selected output components given in OUTARRAY.

3.3.2 uq_evalStochSimMetrics

Syntax

```
uq_evalStochSimMetrics(SPCEModel, X, Y)
uq_evalStochSimMetrics(SPCEModel, X, Y, Name, Value)
Metrics = uq_evalStochSimMetrics(...)
```

Description

`uq_evalStochSimMetrics(SPCEModel, X, Y)` computes error metrics between a SPCE model and a data set (X, Y). Here, X is an experimental design sampled from the input space, and Y contains evaluations of a stochastic model with replications.

`uq_evalStochSimMetrics(SPCEModel, X, Y, Name, Value)` allows for fine-tuning the metrics computation by specifying `Name`, and `Value` pairs of options. The available options are summarized in Table 15.

`Metrics = uq_evalStochSimMetrics(...)` returns a struct with fields corresponding to different error metrics. These include the negative log-likelihood, the normalized Wasserstein distance, and the Wasserstein distances for every sample point. For the definitions of Wasserstein distances, the reader is referred to Zhu and Sudret (2023).

Table 15: Available options of `uq_evalStochSimMetrics`

`Name`	`Value` (default)	Description
`'Nllh'`	Logical default: `true`	Whether to compute the negative log-likelihood
`'Quadrature'`	Integer	Number of nodes for computation of likelihood integral by Gaussian quadrature
`'MCS'`	Integer	Number of function evaluations for computation of likelihood integral by sampling
`'MatlabInt'`	(no value)	Computation of likelihood integral by Matlab built-in integration. No value necessary.

`'WSD'`	Logical default: `true`	Whether to compute the Wasserstein distance (only possible if the data set contains replications)
`'WSDorder'`	Double (p) default: 2	Order of the Wasserstein distance
`'WSDRep'`	Integer default: 1e4	the number of replications/samples of SPCE to evaluate WSD
`'WSDcalcMethod'`	String default: `'ot'`	Method to compute the Wasserstein distance
	`'quantile'`	Calculation based on the empirical quantile
	`'mixed'`	Calculation based on mean, standard deviation and quantile correlations (Schefzik et al., 2021)
	`'ot'`	Exact semi-discrete Wasserstein distance based on optimal transport (Solomon, 2018)

Usage

```
% Create a validation set:
Nval = 1e3;
Xval = uq_getSample(myInput, Nval, 'LHS');
% use 1e4 replications to represent the response distribution
Yval = uq_evalModel(myModel, Xval, 20);

% Evaluate some metrics of the metamodel at the validation set:
valMetrics = uq_evalStochSimMetrics(mySPCE, Xval, Yval);
fprintf('The validation normalized Wasserstein distance is %1.3e. \n', ...
    valMetrics.NormalizedWSD);
```

References

Schefzik, R., Flesch, J., and Goncalves, A. (2021). Fast identification of differential distributions in single-cell RNA-sequencing data with waddR. *Bioinformatics*, 37(19):3204–3211. 24

Solomon, J. (2018). Optimal transport on discrete domains. *AMS Short Course on Discrete Differential Geometry*. 24

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

Ye, M., Meyer, P. D., and Neuman, S. P. (2008). On model selection criteria in multimodel analysis. *Water Resources Research*, 44(3). 10

Zhu, X. and Sudret, B. (2023). Stochastic polynomial chaos expansions to emulate stochastic simulators. *International Journal for Uncertainty Quantification*, 13(2):31–52. 1, 4, 5, 23