ETH zürich

UQLab User Manual
Sensitivity Analysis

S. Marelli, C. Lamas, K. Konakli, C. Mylonas, P. Wiederkehr, 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

S. Marelli, C. Lamas, K. Konakli, C. Mylonas, P. Wiederkehr, B. Sudret, UQLab user manual – Sensitivity analysis, Report UQLab-V2.1-106, Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland, 2024

BibTeX entry

```
@TechReport{UQdoc_21_106,
author = {Marelli, S. and Lamas, C. and Konakli, K. and Mylonas, C. and Wiederkehr,
P. and Sudret, B.},
title = {{UQLab user manual -- Sensitivity analysis}},
institution = {Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich,
Switzerland},
year = {2024},
note = {Report UQLab-V2.1-106}
}
```

Document Data Sheet

Document Ref.	UQLAB-V2.1-106
Title:	UQLAB user manual – Sensitivity analysis
Authors:	S. Marelli, C. Lamas, K. Konakli, C. Mylonas, P. Wiederkehr, 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
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 • Sample-based estimator for Kucherenko indices
V1.2	22/02/2019	UQLAB V1.2 release • ANCOVA indices • Kucherenko indices • Manual restructuring
V1.1	05/07/2018	UQLAB V1.1 release • Borgonovo moment-independent indices
V1.0	01/05/2017	UQLAB V1.0 release • Analytical sensitivity indices from low-rank tensor approximation (LRA) models
V0.9	01/07/2015	Initial release

Abstract

The UQLAB Sensitivity Analysis module is a powerful and flexible tool for performing sensitivity analysis with a number of different techniques. This user manual includes a review of several advanced sensitivity analysis methods and algorithms available in UQLAB, also serving as an overview of the relevant up-to-date literature. It also provides an in-depth, example-driven user guide to help new users to properly set up and solve sensitivity analysis problems with the techniques described. Finally, a comprehensive reference list of the methods and functions available in the UQLAB sensitivity module is given at the end of the manual.

Keywords: Global sensitivity analysis - Sobol' indices - Morris method - Cotter measure - SRC/SRRC coefficients - Polynomial Chaos Expansions - Low-Rank Approximations - Borgonovo Indices - ANCOVA indices - Kucherenko indices

Contents

Chapter 1

Theory

1.1 Introduction

Let $\mathcal{M}(\boldsymbol{x})$ be a mathematical model, depending on a set of input variables $(x_1, ..., x_M)$ described by a random vector $\boldsymbol{X}$. In general, the aim of sensitivity analysis is to describe how the variability of the model response, $y = \mathcal{M}(\boldsymbol{x})$ is affected by the variability of each input variable or combinations thereof.
Apart from the interesting information that such an analysis provides to the modeller in a theoretical sense, sensitivity analysis is useful to spot unimportant input variables and help reduce the dimension of the problem (*model reduction*), among other usages. This kind of analyses are performed with a *black-box* approach, i.e. only based on the model response evaluations for a certain sample of inputs, selected in such a way that they will maximize the output information about the model structure. Furthermore, it is often the case that each run of the model is expensive in terms of computer time and, therefore, sensitivity methods generally aim at reducing the number of model evaluations as much as possible.

1.2 Problem statement

Numerous approaches are available nowadays to perform sensitivity analysis. Broadly speaking, we can either look at the correlation of samples of the input parameters with samples of the output (correlation-based measures), at the values of partial derivatives of the model or at a given point (linearisation methods) or directly at the distribution of the model output. More specifically, global sensitivity analysis (many of which are variance decomposition techniques) aims at decomposing the variance of the model output in terms of contributions of each single input parameter, or combinations thereof.
Here, the following approaches will be discussed (for an up-to-date review and classification of sensitivity analysis techniques, see Iooss and Lemaître (2015)):

- **Sample-based methods** - They are based on the post-processing of available Monte Carlo samples of the model. They are rather simple as they do not require special sampling strategies. The following methods are available in UQLAB:

  - Input/output correlation
  - Standard regression coefficients

- **Linearisation methods** - They are based on the assumption that the model is linear or can be linearised around a central value, usually referred to as the *nominal value* in the literature. They are simple and usually require few model evaluations. However, they fail to identify non-linear behaviours and higher-order interactions between input variables. Within this category, we will review the following methods:

  - Perturbation method
  - Cotter method

- **Global methods** - These methods take into account the whole input domain. They may refer to different features of the model output, such as variance or distribution. In the current version of UQLAB the following methods are implemented:

  - Morris' elementary effects
  - Borgonovo sensitivity indices
  - Sobol' sensitivity indices
  - ANCOVA sensitivity indices
  - Kucherenko sensitivity indices

Note that other authors use a different classification of the aforementioned methods, by identifying the class of so-called *screening methods*. Screening methods aim at providing a qualitative ranking of the importance of the input variables w.r.t. the model response, with relatively low computational costs. Cotter and Morris' methods are commonly grouped in this class. In this context, we could include PCE- and LRA-based Sobol' indices (Section 1.5.3.4 and Section 1.5.3.5) and the ANCOVA indices (Section 1.5.4) in the class of screening methods.

Note: Some methods are specifically developed for the case of independent input variables and are only to be applied in case this premise is fulfilled. While it is also remarked in the respective sections, the methods *only applicable in the case of independent input variables* are:

  - Standard regression coefficients
  - Perturbation method
  - Sobol' sensitivity indices

1.3 Sample-based methods

1.3.1 Input/Output Correlation

Perhaps the most intuitive measure of sensitivity of model response Y to the components of the input random vector $\boldsymbol{X}$ is the component-wise correlation coefficients between the two. Due to its simplicity this method is applicable regardless of dependence in the input variables. Consider a sampling of the input random vector $\mathcal{X} = \left\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, ..., \boldsymbol{x}^{(N)}\right\}$ and the corresponding model responses $\mathcal{Y} = \left\{y^{(1)}, y^{(2)}, ..., y^{(N)}\right\}$. The linear correlation coefficient ρ_i between the i^{th} input and the output is defined as

$$\rho_i \overset{\text{def}}{=} \rho(X_i, Y) = \frac{\mathbb{E}\left[(X_i - \mu_i)(Y - \mu_Y)\right]}{\sigma_i \, \sigma_Y}, \tag{1.1}$$

where $\mu_i \overset{\text{def}}{=} \mathbb{E}\left[X_i\right]$ and $\mu_Y \overset{\text{def}}{=} \mathbb{E}\left[Y\right]$ and σ_i and σ_Y are the corresponding standard deviations. The unbiased Monte Carlo estimator of the sample correlation reads:

$$\widehat{\rho}_i = \frac{\sum_{j=1}^{N}(x_i^{(j)} - \widehat{\mu}_i)(Y^{(j)} - \widehat{\mu}_Y)}{\widehat{\sigma}_i \, \widehat{\sigma}_Y}, \tag{1.2}$$

where the hat notation $\widehat{\cdot}$ identifies the sample estimates of the corresponding quantities in Eq. (1.1).

Linear correlation, however, is known to be inaccurate in the presence of strongly non-linear dependence between variables. A more stable estimator that relies on the monotonicity instead of the linearity of the dependence of two variables, is the Spearman's rank correlation index ρ_S.

To calculate Spearman's ρ_S, each component of the input sample and its response are transformed into their rank-equivalents:

$$R_i = \left\{ r_i^{(j)} \in \{1, ..., N\} : r_i^{(j)} > r_i^{(k)} \iff x_i^{(j)} > x_i^{(k)} \; \forall j, k \in \{1, ..., N\} \right\}. \tag{1.3}$$

The rank-transformed model response then reads $R_Y = \left\{ r_Y^{(1)}, r_Y^{(2)}, ..., r_Y^{(N)} \right\}$. The Spearman correlation coefficient is defined as the linear correlation coefficients between the ranks:

$$\rho_i^S \overset{\text{def}}{=} \rho_S\left(X_i, Y\right) = \rho(R_i, R_Y). \tag{1.4}$$

1.3.2 Standard Regression Coefficients

Another simple measure of sensitivity of the model response to each of the input variables is given by the Standard Regression Coefficients (SRC). Since it consists of a simplified variance decomposition, it only leads to useful results for uncorrelated input variables. For this method, the model $\mathcal{M}(X)$ is approximated by a linear regression:

$$Y = \mathcal{M}(\boldsymbol{X}) \approx \beta_0 + \sum_{i=1}^{M} \beta_i X_i \,, \tag{1.5}$$

The advantage of this formulation is that, in case of independent variables X_i, the total variance of the model can be estimated by:

$$\widehat{\sigma}_Y^2 = \sum_{i=1}^{M} \beta_i^2 \sigma_{X_i}^2 \,, \tag{1.6}$$

where $\widehat{\sigma}_Y^2$ and $\sigma_{X_i}^2$ are the variances of the response and each input, respectively.
Given a sample of the input $\mathcal{X} = \left\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, ..., \boldsymbol{x}^{(N)}\right\}$ and the corresponding sample of model responses $\mathcal{Y} = \left\{y^{(1)}, y^{(2)}, ..., y^{(N)}\right\}$, the least-square estimator of the $\boldsymbol{\beta} = \{\beta_1, \beta_2, ..., \beta_M\}$ is given by:

$$\widehat{\boldsymbol{\beta}} = \left(\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{A}\right)^{-1}\boldsymbol{A}^{\boldsymbol{T}}\mathcal{Y} \,, \tag{1.7}$$

where the matrix $\boldsymbol{A}$ has for columns the M components of the input sampling $\boldsymbol{\mathcal{X}}$.
The SRC indices are then defined as:

$$SRC_i = \frac{\widehat{\beta}_i \sigma_i}{\sigma_Y} \tag{1.8}$$

A straightforward extension to this method consists in substituting both the experimental design samples and their response with the corresponding ranks so as to obtain the standard rank regression coefficients (SRRC). Applying a rank-transform to both Y and $\boldsymbol{X}$ in Eq. (1.5) yields:

$$r_Y \approx \gamma_0 + \sum_{i=1}^{M} \gamma_i r_i \,, \tag{1.9}$$

where the r_Y are the ranks of the model responses, r_i the ranks of the input coordinates and γ_i the rank-regression coefficients. Eq. (1.7) then becomes:

$$\widehat{\boldsymbol{\gamma}} = \left(\boldsymbol{R}^{\boldsymbol{T}}\boldsymbol{R}\right)^{-1}\boldsymbol{R}^{\boldsymbol{T}}R_Y \,, \tag{1.10}$$

where the matrix $\boldsymbol{R}$ has for columns the M components of the rank-transformed experimental design R_i (Eq. (1.4)). The corresponding indices read:

$$SRRC_i = \widehat{\gamma}_i. \tag{1.11}$$

Note that the standard deviations in Eq. (1.8) cancel out after the rank transform because each rank-transformed variable is a permutation of the vector of integer ranks $\{1, 2, \ldots, N\}$.

1.4 Linearization methods

1.4.1 Perturbation method

The basis of the perturbation method is to substitute the model by its Taylor expansion around a value of interest, usually the mean value of the input distribution. Once the first-order expansion is found, it can be used to analytically find the mean and variance of the model, and

to develop an importance measure of each of the input variables, by looking at their contribution to the model variance in terms of their own initial variance. The decomposition leading to the importance measures is based on the assumption of independent input variables.
Let $\mu_{\boldsymbol{X}} = \{\mu_1, ..., \mu_M\}^{\mathsf{T}}$ denote the vector containing the means of the marginal distributions of the input variables and $\boldsymbol{C}$ be the covariance matrix. The first-order expansion of the model $\mathcal{M}(\boldsymbol{x})$ around the mean value is as follows:

$$\mathcal{M}(\boldsymbol{x}) = \mathcal{M}(\mu_{\boldsymbol{X}}) + \sum_{i=1}^{M} \frac{\partial M}{\partial x_i}|_{x=\mu_{\boldsymbol{X}}}(x_i - \mu_i) + o(\|x - \mu_{\boldsymbol{X}}\|^2)\,.$$

The mean and variance can be estimated from this expression as:

$$\widehat{\mathbb{E}}[\mathcal{M}(\boldsymbol{X})] = \mathcal{M}(\mu_{\boldsymbol{X}})\,, \tag{1.12}$$

$$\widehat{\mathrm{Var}}\left[\mathcal{M}(\boldsymbol{X})\right] = \nabla\mathcal{M}^T(\mu_{\boldsymbol{X}}) \cdot \boldsymbol{C} \cdot \nabla\mathcal{M}^T(\mu_{\boldsymbol{X}})\,. \tag{1.13}$$

If the variables are independent, $\boldsymbol{C}$ contains only the variances of the input variables on the diagonal, so the estimate of the total variance in (1.13) simplifies to:

$$\widehat{\mathrm{Var}}\left[\mathcal{M}(\boldsymbol{X})\right] = \sum_{i=1}^{M} \left(\frac{\partial M}{\partial x_i}(\mu_{\boldsymbol{X}})\right)^2 \sigma_i^2\,. \tag{1.14}$$

As this is a sum of positive terms over the number of variables, each of the terms represents its contribution to the estimate of the total variance. Finally, to construct the perturbation method sensitivity estimates η_i, these terms are divided by the total variance, so that they sum up to one:

$$\eta_i = \frac{\left(\frac{\partial M}{\partial x_i}(\mu_{\boldsymbol{X}})\right)^2 \sigma_i^2}{\widehat{\mathrm{Var}}\left[\mathcal{M}(\boldsymbol{X})\right]}, \qquad i = 1, ..., M. \tag{1.15}$$

Note that these estimates are based on the *assumption of independent input variables* and are thus only applicable in case this is assumption verified.

1.4.2 Cotter method

The Cotter method (Cotter, 1979) is a simple and low-cost method that allows for ranking input parameters. It can be applied regardless of dependence between the input variables. Let us consider an input vector $\boldsymbol{x} = \{x_1, ..., x_M\}^T \in \mathcal{D}_x$. Each input variable X_i is assumed to vary in a known interval described by its *low* and *high* levels $X_i \in [x_i^-, x_i^+]$. Then, a systematic fractional replicate design is carried out, in which the following $2M + 2$ model evaluations are performed:

- One run with all variables at their low levels, say $y_0 = \mathcal{M}(x_1^-, ..., x_M^-)$
- M runs at low levels, switching one variable at a time to its high level: $y_i = \mathcal{M}(x_1^-, ..., x_i^+, ..., x_M^-),\ i = 1, ..., M$

- M runs at high levels, switching one variable at a time to its low level: $y_i = \mathcal{M}(x_1^+, ..., x_i^-, ..., x_M^+),\ i = M+1, ..., 2M$
- One run with all variables at their high levels, $y_{2M+1} = \mathcal{M}(x_1^+, ..., x_M^+)$

The importance of the variables is then measured by:

$$I_{Cotter}(i) = |C_o(i)| + |C_e(i)|\,, \tag{1.16}$$

where $C_o(i)$ and $C_e(i)$ denote the expectation of the importance of the odd and even order effects involving the input variable X_i respectively, and are given by:

$$C_o(i) = \frac{1}{4}\left[(y_{2M+1} - y_{M+i}) + (y_i - y_0)\right], \tag{1.17}$$

$$C_e(i) = \frac{1}{4}\left[(y_{2M+1} - y_{M+i}) - (y_i - y_0)\right]. \tag{1.18}$$

The great advantage of this method is that it only requires $2M + 2$ runs of the model, yet it has the drawback that, if the odd or even orders cancel each other out among dimensions, this can lead to an incorrect importance measure.

1.5 Global methods

1.5.1 Morris method

The Morris method (Morris, 1991) is somehow similar to the Cotter method, but tries to cover more exhaustively the input space. Instead of looking at a single global perturbation, it creates a grid where multiple different perturbations are possible. The sensitivity measure is then built based on the mean and standard deviation of that set of perturbations. Like the Cotter method, it is applicable for any kind of dependence in the input random vector.
With no loss of generality, let us assume that the input parameters $\boldsymbol{X}$ vary in an M-dimensional unit hypercube, i.e. $X_i \in [0, 1],\ i = 1, ..., M$ (in the practical case, these sets can be rescaled to the real input space).
A grid of p values is created for each parameter. It consists of the set:

$$\mathcal{G}_i = \left\{0, \frac{1}{p-1}, \frac{2}{p-1}, ..., 1\right\}. \tag{1.19}$$

The whole input set is then defined by:

$$\boldsymbol{\mathcal{G}} = \mathcal{G}_1 \times ... \times \mathcal{G}_M \tag{1.20}$$

Let us now define a perturbation parameter Δ such that it is a multiple of $\frac{1}{p-1}$,

$$\Delta = \frac{\bar{c}}{p-1},\ \text{for a fixed } \bar{c} \in \{1, ..., p-1\}. \tag{1.21}$$

For a given value of $\boldsymbol{x} = \{x_1, ..., x_M\}^\mathsf{T} \in \boldsymbol{\mathcal{G}}$, such that $x_i + \Delta \leq 1$, the *elementary effect* for the

variable x_i is defined by:

$$d_i(\boldsymbol{x}) = \frac{\mathcal{M}(x_1, ..., x_i + \Delta, ..., x_M) - \mathcal{M}(\boldsymbol{x})}{\Delta} \tag{1.22}$$

For each input variable, there exist a finite number of $p^{M-1}(p - \bar{c})$ elementary effects that can be computed. Morris' original approach suggests to estimate the distribution of the elementary input variables by randomly sampling r values in $\boldsymbol{\mathcal{G}}$ and computing (1.22) for each of them. Once these values are found, the estimators of the moments of the distribution can be analysed as follows:

- **Mean**, $\widehat{\mu}_i$: A high value for the mean indicates that the input variable X_i is important and, thus, its input values cause variability on the outputs of the model. Low mean would indicate an unimportant input variable.
- **Standard deviation**, $\widehat{\sigma}_i$: A high value of the standard deviation indicates that the effect of the variable X_i is significant, either because of a nonlinear behaviour with respect to this variable or because of interactions with other input variables.

Note that using this method with samples of r values would require a total of $2Mr$ runs ($2r$ runs per elementary effect).

1.5.1.1 Trajectory-based sampling strategy

Using a *trajectory-based* sampling strategy, the required number of model evaluations can be reduced to $(M + 1)r$. The key idea is as follows: since two model evaluations are needed for computing the elementary effect of each variable, one of them can be shared between two different elementary effects by perturbing the current point properly. This means that only one new model evaluation is needed for each variable after the starting point, leading to $M + 1$ model evaluations in total. Let us introduce the following vectors and matrices:

- **Random starting point**, $\boldsymbol{x}^*$
  It is a random point (dimension $M \times 1$) of the grid, constructed in such a way that increasing its coordinates in any dimension will lead to a point that still lies in the grid.
- **Base matrix**, $\boldsymbol{B}$
  It is a $M \times (M + 1)$ matrix containing ones in its strict lower triangular part and zeros on the rest.
- **Matrix $\boldsymbol{J_{i,j}}$**
  It denotes a matrix made of ones, with dimension $i \times j$.
- **Random displacements matrix $\boldsymbol{D}$**
  It consists of a diagonal $M \times M$ matrix that contains -1 or 1 on its diagonal, randomly chosen with equal probability.

- **Identity permutation $\boldsymbol{P}$**
  It is a random permutation of the identity matrix of dimension $M \times M$

The trajectory can be represented by a $M \times (M+1)$ matrix $\boldsymbol{B}^*$ that contains in each column a point, starting from the random point $\boldsymbol{x}^*$. Each successive trajectory point is obtained from the previous by sequentially increasing one of its coordinate by Δ. Each coordinate can be increased only once.
A matrix $\boldsymbol{B}'$ satisfying such properties is given by:

$$\boldsymbol{B}' = \left(\boldsymbol{J}_{M+1,1}(\boldsymbol{x}^*)^T + \Delta \boldsymbol{B}\right)^T . \tag{1.23}$$

However, such trajectory is still deterministic except for the starting point. The following formula provides a properly randomized version (Morris, 1991):

$$\boldsymbol{B}^* = \left[\left(\boldsymbol{J}_{M+1,1}(\boldsymbol{x}^*)^T + \frac{\Delta}{2}\left[(2\boldsymbol{B} - \boldsymbol{J}_{M+1,M})\boldsymbol{D}^* + \boldsymbol{J}_{M+1,1}\right]\right)\boldsymbol{P}\right]^T \tag{1.24}$$

1.5.1.2 Modified estimator

As it is based on the mean, the estimator $\widehat{\mu}$ fails to identify the effects of those variables that can cancel each other out (i.e., alternate signs). To avoid this drawback, a new estimator $\widehat{\mu}_*$ was introduced in Campolongo et al. (2007) which, instead of computing the mean of the elementary effects of (1.22), computes the mean of its absolute values:

$$d_i^*(\boldsymbol{x}) = \left|\frac{\mathcal{M}(x_1, ..., x_i + \Delta, ..., x_M) - \mathcal{M}(\boldsymbol{x})}{\Delta}\right| \tag{1.25}$$

This estimator, however, lacks the information about the sign of the effect introduced by the input variable; yet combined with the previous $\widehat{\mu}_i$ and $\widehat{\sigma}_i$ it can provide more complete information about the actual importance of each input variable, the sign of its effect and the nature of this effect (linear, non-linear or interactions).

1.5.1.3 Graphical representation

Morris sensitivity analysis provides two estimators for each variable $\hat{\mu}_i$ and $\hat{\sigma}_i$. It is customary to represent them in a scatter-plot, often referred to as a Morris plot. Figure 1 shows a Morris plot for an 8-dimensional problem (see Section 2.1.6 for details on the model).
Each point on the plot is the short name of one of input variables. Its position on the $\mu\sigma$-plane is given by the corresponding $\hat{\mu}_i$ and $\hat{\sigma}_i$. Points close to the origin correspond to unimportant variables. Points far away from the origin in the μ direction are generally important. Points far away from the origin in the σ direction represent variables with significantly non-linear contributions, hence sometimes difficult to properly assess.

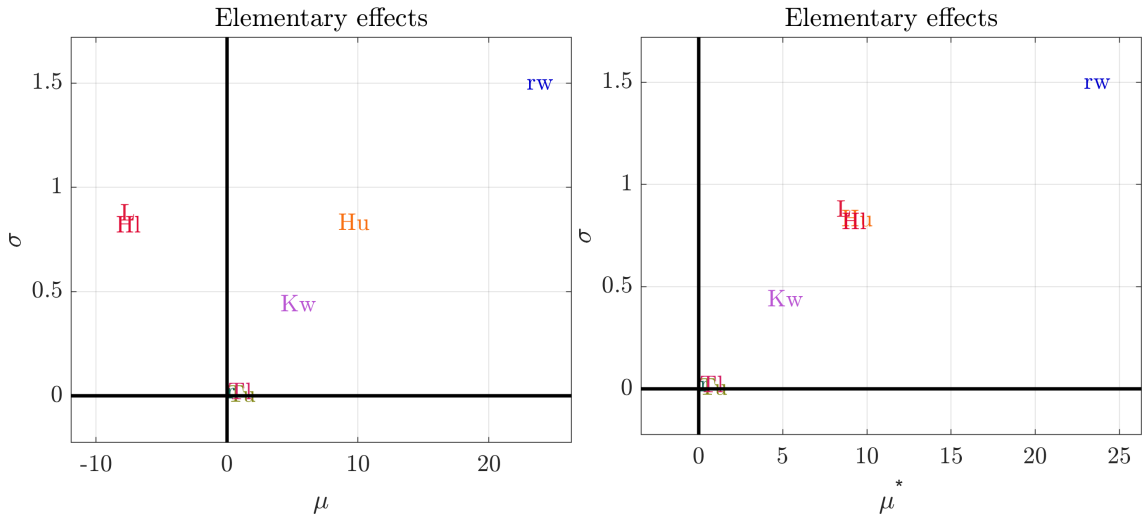

Figure 1: Morris plots for an 8-dimensional problem (see Section 2.1.6 for details). The μ of each index is given by its position on the $x-$axis, while the corresponding σ by that on the $y-$axis.

1.5.2 Borgonovo indices

The Borgonovo index δ_i (Borgonovo, 2007) of a random input variable X_i is a measure of the expected shift in the probability distribution of the model output when X_i is set to a fixed value. If the variable is not important, the expected shift is close to zero, otherwise it takes a larger value. Its derivation is not based on any assumptions on the dependence structure of the input variables $\boldsymbol{X}$, which makes it applicable in either case, dependent or independent input variables. The definition of the (first-order[1]) Borgonovo index reads:

$$\delta_i = \frac{1}{2}\mathbb{E}_{X_i}\left[\int_{\mathcal{D}_y} |f_Y(y) - f_{Y|X_i}(y)|dy\right], \tag{1.26}$$

where f_Y is the probability distribution function (PDF) of the model output and $f_{Y|X_i}$ is the conditional distribution on X_i. The expectation value in Eq. (1.26) can be written as:

$$\delta_i = \frac{1}{2}\int_{\mathcal{D}_{x_i}} f_{X_i}(x_i)\int_{\mathcal{D}_y} |f_Y(y) - f_{Y|X_i}(y|x_i)|dydx_i\,. \tag{1.27}$$

1.5.2.1 Estimation of Borgonovo indices

Accurate estimation of the Borgonovo index δ_i for an input variable X_i relies on the precise approximation of both the unconditional PDF f_Y and the conditional PDF $f_{Y|X_i}$ of the model output. UQLAB provides two methods to estimate the inner integral in Eq. (1.27), *Histogram-* and *CDF-based*. The approximation of the conditional distribution of $Y|X_i$ is similar for both methods. N samples are drawn from the input random vector $\boldsymbol{X}$ and binned into so-called *classes* on variable X_i. Each of the P classes is defined by its lower and upper limits $\{a_p, b_p\}$

[1] At this point, higher-order Borgonovo indices are not defined yet.

with $a_p < b_p$, $p \in \{1, \ldots, P\}$. Class C_p includes all samples $\boldsymbol{x}^{(j)}$, $j \in \{1, \ldots, N\}$, whose realizations of X_i, $x_i^{(j)}$, lie between between a_p and b_p. In short, this can be written as $C_p = \left\{\boldsymbol{x}^{(j)} : a_p \leq x_i^{(j)} < b_p\right\}$. The distribution of Y in each class is then:

$$f_{Y|X_i=x_i}(y) \approx \hat{f}_{Y|C_p \supset x_i}(y)\,, \tag{1.28}$$

where $C_p \supset x_i$ means that all the points in the same class C_p as x_i are considered to estimate the conditional PDF.

Class building

The classes can either be disjoint (if $b_p = a_{p+1}$, $p = 1, \ldots, P-1$) or overlapping (if $b_p > a_{p+1}$, $p = 1, \ldots, P-1$). Overlapping classes can improve the estimation in two ways. First, for a given input sample size N, one can define more classes each containing a sufficient amount of samples, thus leading to a better estimate. Second, samples in the overlapping regions are included in both classes and correlate the estimates in neighbouring classes and smooth the change. Another possibility to improve the estimation is to allow the class widths $(b_p - a_p)$ to be dynamically set by equally-spaced quantiles of X_i, ensuring the same number of samples in each class.

Histogram-based estimation

In the original approach of Borgonovo (2007), the distributions for the inner integral in Eq. (1.27) are estimated from a sample of the input random vector $\mathcal{X} = \left\{\boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(N)}\right\}$ and the corresponding model evaluations $\mathcal{Y} = \mathcal{M}(\boldsymbol{x}^{(1)}, ..., \mathcal{M}(\boldsymbol{x}^{(N)}))$. The unconditional PDF values are directly calculated through a histogram of $\mathcal{Y}$. In order to estimate the PDF values of the conditional distributions, a histogram is set up in each class. In this setting, the sum of the absolute differences between the conditional and unconditional PDF values yields an estimation of the inner integral in Eq. (1.27). A graphical representation of this approach is given in Figure 2.

Since this can be expected to be a rough approximation in the presence of small samples, Plischke et al. (2013) and Caniou (2012) proposed the use of kernel density estimators to improve the estimate in Eq. (1.28). After extensive testing, a choice has been made in UQLAB to avoid the usage of kernel density estimation for the conditional PDF estimation due to an observed tendency of such estimators to produce biased estimates in the inner integral in Eq. (1.27). Consequently, computations are based on histograms only and a reasonable sample size (*e.g.* $N \geq 10^3$) is recommended. For the histograms to accurately estimate the conditional PDF, it is important to have enough samples in each class and to choose a reasonable number of histogram bins. While the number of samples per class is ensured by quantile-based classes, the number of histogram bins and the bin widths should be chosen for each class separately.

Once the absolute difference of the histograms is calculated for each class, it is then averaged by taking the mean of the estimates. This averaging corresponds to the outer integral in

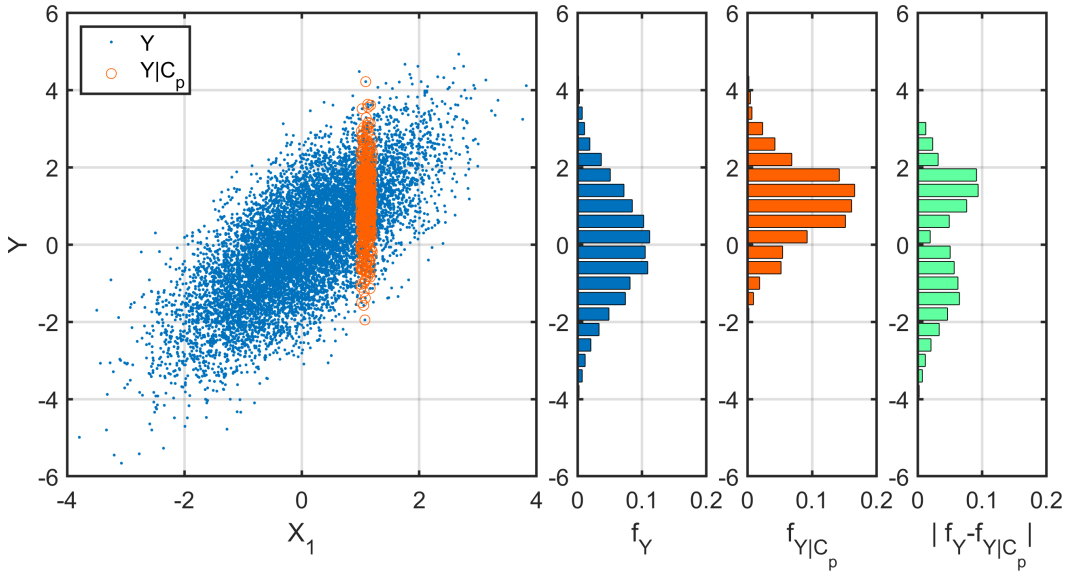

Figure 2: Approximating conditional distribution $f_{Y|X_1=1.1}$ with conditional distribution $\hat{f}_{Y|C_p}$ where $Y|C_p$ denotes the model responses where x_1 lies between $a_p = 1.0$ and $b_p = 1.2$.

Eq. (1.27).

CDF-based estimation

An alternative method to estimate the inner integral in Eq. (1.27) is based on the computation of cumulative distribution functions and was introduced in Liu and Homma (2009). It can be shown that:

$$\int S(y)dy = \int |f_Y(y) - f_{Y|X_i}(y|x_i)|dy \tag{1.29}$$

can be replaced with a summation of the difference of the unconditioned and conditional cumulative density function (CDF) at the crossing points $\{a_1, \cdots, a_l\}$ of f_Y and $f_{Y|X_i}$. The summation formula reads:

$$\int S(y)dy = \begin{cases} 2 \cdot \sum_{k=1}^{l} (-1)^{k-1}[F_Y(a_k) - F_{Y|X_i}(a_k)] & \text{if} \quad F_Y(a_1) > F_{Y|X_i}(a_1) \\ \\ 2 \cdot \sum_{k=1}^{l} (-1)^{k-1}[-F_Y(a_k) + F_{Y|X_i}(a_k)] & \text{if} \quad F_Y(a_1) < F_{Y|X_i}(a_1) \end{cases} \tag{1.30}$$

To evaluate Eq. (1.30), kernel smoothing estimators of the CDFs F_Y and $F_{Y|X_i}$ are used as in Liu and Homma (2009) and Caniou (2012). Again, this is done for each class $C_p \supset x_i$ and averaged by taking the mean.

1.5.2.2 Comparison of histogram- and CDF-based estimation methods

Considering a simple model $M(X_1, X_2) = X_1 + X_2$ with marginal distributions $X_1 \sim \mathcal{N}(0, 1)$ and $X_2 \sim \mathcal{N}(1, 2)$ the Borgonovo indices were estimated 50 times with the number of classes

equal to 10.

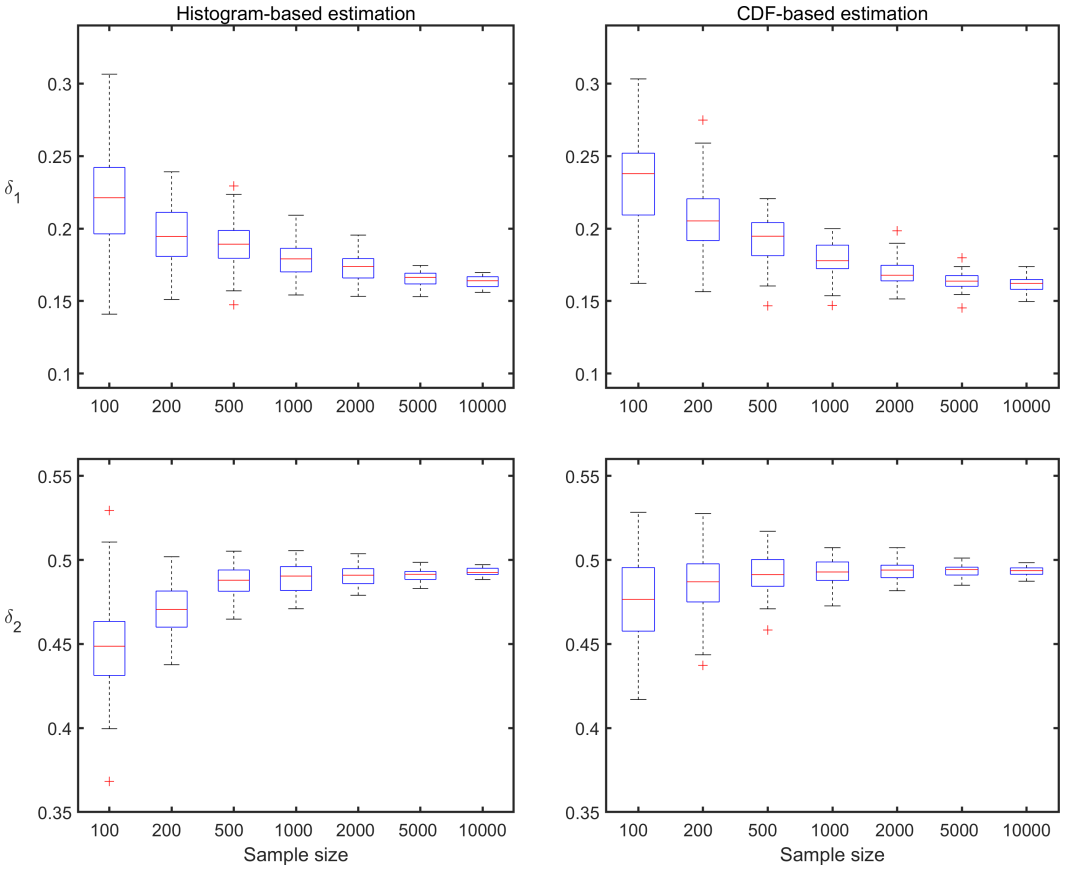

Figure 3: Boxplots of the Borgonovo histogram- and CDF-based estimate for indices of $\mathcal{M}(X_1, X_2) = X_1 + X_2$ based on 50 re-samplings for each sample size.

The histogram-based implementation yields as robust results as the CDF-based, as is demonstrated in Figure 3. The two methods yield roughly the same results. Since for the CDF-based approach the crossing points of the CDF's have to be found, which seems to unnecessarily complicate the estimation, the histogram-based approach is the default in UQLAB, but both methods are available.

1.5.3 Sobol' indices (ANOVA)

Sobol' indices (Sobol', 1993) are based on the idea of defining the expansion of the computational model into summands of increasing dimension. Analogously, the total variance of the model is described in terms of the sum of the variances of the summands. This decomposition only holds for independent input variables and this method is often referred to as ANOVA (ANalysis Of VAriance).

To simplify the notation and without loss of generality, all the equations in the following will assume that the input variables are uniform variables in $[0, 1]$ and, therefore, that the support of the input set is $\mathcal{D}_{\boldsymbol{X}} = [0, 1]^M$. (In the general case, the integration domains would need to be properly defined and all of the expectation value integrals to be weighted by the

appropriate marginal PDFs.)

1.5.3.1 Sobol' decomposition

The Sobol' decomposition is defined as:

$$f(x_1,...,x_M) = f_0 + \sum_{i=1}^{M} f_i(x_i) + \sum_{1\le i<j\le M} f_{ij}(x_i,x_j) + ... + f_{1,2,...,M}(x_1,...,x_M), \tag{1.31}$$

where the following conditions hold:

i) The term f_0 is constant and equal to the expected value of $f(\boldsymbol{X})$.

ii) The integrals of the summands with respect to their own variables $f_{i_1,...,i_s}$ vanish:

$$\int_0^1 f_{i_1,...,i_s}(x_{i_1},...,x_{i_s})\, dx_{i_k} = 0 \quad \text{if } 1 \le k \le s\,. \tag{1.32}$$

For integrable functions in $\mathcal{D}_{\boldsymbol{X}}$ and independent input variables $\boldsymbol{X}$, this expansion is shown to exist and to be unique in Sobol' (1993). It is also shown that all the summands of the expansion can be computed by integrals in a recursive manner as follows:

$$\begin{aligned} f_0 &= \int_{\mathcal{D}_{\boldsymbol{X}}} f(\boldsymbol{x}) d\boldsymbol{x}, \\ f_i(x_i) &= \int_0^1 ... \int_0^1 f(\boldsymbol{x}) d\boldsymbol{x}_{\sim i} - f_0, \\ f_{ij}(x_i,x_j) &= \int_0^1 ... \int_0^1 f(\boldsymbol{x}) d\boldsymbol{x}_{\sim (ij)} - f_0 - f_i(x_i) - f_j(x_j), \end{aligned} \tag{1.33}$$

where $\boldsymbol{x} = \{x_1,...,x_M\}$ and the notation $\sim$ indicates that variables are excluded, *e.g.*

$$\boldsymbol{x}_{\sim i} = (x_1,...,x_{i-1},x_{i+1},...,x_M)\,. \tag{1.34}$$

The higher-order summands can be constructed in an analogous way. However, the use of Eq. (1.31) is only for computing the variance decomposition, which is presented next. Following the previous results, the total variance of $f(\boldsymbol{X})$ is defined as:

$$D = \int_{\mathcal{D}_{\boldsymbol{X}}} f^2(\boldsymbol{x}) d\boldsymbol{x} - f_0^2, \tag{1.35}$$

and the partial variances are computed as:

$$D_{i_1,...,i_s} = \int_0^1 ... \int_0^1 f^2_{i_1,...,i_s}(x_{i_1},...,x_{i_s}) dx_{i_1}...dx_{i_s} \quad 1 \le i_1 < ... < i_s \le M;\ s = 1,...,M \tag{1.36}$$

since the summands have zero mean due to Eq. (1.32). The partial variances have the property that they sum up to the total variance.

1.5.3.2 Sensitivity indices

First- and higher-order indices

The variance decomposition leads to a natural definition of the sensitivity measures:

$$S_{i_1,\ldots,i_s} = \frac{D_{i_1,\ldots,i_s}}{D}, \tag{1.37}$$

which represent the relative contribution of each group of variables $\{X_{i_1}, \ldots, X_{i_s}\}$ to the total variance. The index with respect to one input variable X_i is called the *first-order Sobol' index* and represents the effect of X_i alone. Multiple-term indices, *e.g.* $S_{ij},\ i \neq j$, are referred to as *higher-order* Sobol' indices and are interaction indices that account for the effects of the interactions of the variables X_i and X_j that cannot be decomposed into the contributions of those variables separately. Further higher-order indices can be interpreted analogously.

Total indices

The total Sobol' index of input variable X_i, denoted S_i^T, is the sum of all the Sobol' indices involving this variable:

$$S_i^T = \sum_{\{i_1,\ldots,i_s\} \supset i} S_{i_1,\ldots,i_s}. \tag{1.38}$$

This definition is not practical for actual computations since it would result in computing each index separately. Instead, by denoting $S_{\sim i}$ the sensitivity measure of all the variables excluding variable X_i, i.e. $S_{\sim i} = S_{\mathbf{v}}$, where $\mathbf{v} = \{1, ..., i-1, i+1, ..., M\}$, the total index can be written as:

$$S_i^T = 1 - S_{\sim i}, \tag{1.39}$$

From this, the idea of computing indices of groups of variables is developed.

Closed indices

If the components of the input vector $\boldsymbol{x} \in [0,1]^M$ are separated into two groups $\boldsymbol{x}_{\mathbf{v}}$ and $\boldsymbol{x}_{\mathbf{w}}$, where $\mathbf{v} = \{i_1, ..., i_s\} \subset \{1, ..., M\}$ and $\mathbf{w} = \sim\mathbf{v}$ such that $y = f(\boldsymbol{x}) = f(\boldsymbol{x}_{\mathbf{v}}, \boldsymbol{x}_{\mathbf{w}})$, the Sobol' index with respect to $\boldsymbol{X}_{\mathbf{v}}$ can be written as:

$$S_{\mathbf{v}} = \frac{\mathrm{Var}\left[\mathbb{E}\left[Y|\boldsymbol{X}_{\mathbf{v}} = \boldsymbol{x}_{\mathbf{v}}\right]\right]}{\mathrm{Var}\left[Y\right]}. \tag{1.40}$$

This index accounts for the sum of all the single effects, also referred to as structural effects, as well as interactions between the variables $\boldsymbol{X}_{\mathbf{v}}$. In the univariate case, *i.e.* $\mathbf{v} = \{i\}$, $|\mathbf{v}| = 1$, this index leads to the previously introduced first-order Sobol' index. In the general case with $|\mathbf{v}| > 1$, this measure is called the *closed* index of $\boldsymbol{X}_{\mathbf{v}}$, since it includes all univariate and interaction effects of $\boldsymbol{X}_{\mathbf{v}}$ but no interactions with $\boldsymbol{X}_{\mathbf{w}}$.

To compute higher-order indices, lower-order indices can be subtracted from the closed index. For example, the second order index of (X_i, X_j) is the closed index of these two variables

(denoted by $S_{\{ij\}}$) minus the single effect of each of them:

$$S_{ij} = S_{\{ij\}} - S_i - S_j. \tag{1.41}$$

1.5.3.3 Monte Carlo-based estimation

The variances described in Eqs. (1.35) and (1.36) may be computed by means of Monte Carlo simulation using the following estimators:

$$\widehat{f}_0 = \frac{1}{N}\sum_{n=1}^{N} f(\boldsymbol{x}^{(n)}), \tag{1.42}$$

$$\widehat{D} = \frac{1}{N}\sum_{n=1}^{N} f^2(\boldsymbol{x}^{(n)}) - \widehat{f}_0^2, \tag{1.43}$$

$$\widehat{D}_i = \frac{1}{N}\sum_{n=1}^{N} f(x_i^{(n)}, \boldsymbol{x}_{\sim i}^{(n)}) f(x_i^{(n)}, \boldsymbol{x}'^{(n)}_{\sim i}) - \widehat{f}_0^2, \tag{1.44}$$

where $\boldsymbol{x}'$ denotes a realization of $\boldsymbol{X}$ independent of $\boldsymbol{x} = \left\{x_i^{(n)}, \boldsymbol{x}_{\sim i}^{(n)}\right\}^{\mathsf{T}}$, and the subscript $\boldsymbol{x}_{j,\sim i}$ indicates the j-th realization of $\boldsymbol{x}$ which does not contain the input variable i.

To compute total indices it is easier to use group indices in combination with Eq. (1.39). Let us now denote by $y^{\mathbf{v}} = f(\boldsymbol{x}_{\mathbf{v}}, \boldsymbol{x}'_{\mathbf{w}})$, where $\boldsymbol{x}'_{\mathbf{w}}$ is a sample of $\boldsymbol{X}_{\mathbf{w}}$ independent of $\boldsymbol{x}_{\mathbf{w}}$. Analogously, $\boldsymbol{X}'_{\mathbf{w}}$ denotes an independent copy of $\boldsymbol{X}_{\mathbf{w}}$ and $Y_{\mathbf{v}} = f(X_{\mathbf{v}}, X'_{\mathbf{w}})$. Following Janon et al. (2014), we can rewrite the index in Eq. (1.40) in terms of sample covariance as follows:

$$S_{\mathbf{v}} = \frac{\mathrm{Cov}\left[Y, Y_{\mathbf{v}}\right]}{\mathrm{Var}\left[Y\right]}, \tag{1.45}$$

which leads to a first estimator, hereafter referred to as the *Homma* estimator, initially proposed by Saltelli and Homma (1996):

$$S_{\mathbf{v}} = \frac{\frac{1}{N}\sum y_i y_i^{\mathbf{v}} - (\frac{1}{N}\sum y_i)(\frac{1}{N}\sum y_i^{\mathbf{v}})}{\frac{1}{N}\sum y_i^2 - (\frac{1}{N}\sum y_i)^2}, \tag{1.46}$$

where the sums without super- or subscript are considered to be $\sum \overset{\text{def}}{=} \sum_{i=1}^{N}$. A similar estimator shown to perform better in most cases, hereafter referred to as the *Janon* estimator, is described in Janon et al. (2014):

$$S_{\mathbf{v}} = \frac{\frac{1}{N}\sum y_i y_i^{\mathbf{v}} - \left(\frac{1}{N}\sum\left[\frac{y_i + y_i^{\mathbf{v}}}{2}\right]\right)^2}{\frac{1}{N}\sum\left[\frac{(y_i)^2 + (y_i^{\mathbf{v}})^2}{2}\right] - \left(\frac{1}{N}\sum\left[\frac{y_i + y_i^{\mathbf{v}}}{2}\right]\right)^2}. \tag{1.47}$$

NB: It is clear that for computing the effect of one group of variables, only one estimator is needed.

1.5.3.4 PCE-based Sobol' indices

Sobol' indices are traditionally evaluated by Monte Carlo simulation which make them difficult to use with computationally expensive models $\mathcal{M}$. In order to bypass this problem, Sudret (2008) proposed an original post-processing of polynomial chaos expansions (PCE) for sensitivity analysis. As it will be demonstrated below, the Sobol' decomposition of a truncated PCE:

$$\widehat{y} = \mathcal{M}^{PC}(\boldsymbol{x}) = \sum_{\boldsymbol{\alpha}\in\mathcal{A}} \widehat{y}_{\boldsymbol{\alpha}}\, \Psi_{\boldsymbol{\alpha}}(\boldsymbol{x})\,, \tag{1.48}$$

can be established analytically. For more information about PCE refer to the UQLAB User Manual – Polynomial Chaos Expansions.

The truncated expansion in Eq. (1.48) can be rearranged so as to reflect the decomposition into summands of increasing order. For any non-empty set $\mathbf{u} \subset \{1,\ldots,M\}$ and any finite truncation set $\mathcal{A} \subset \mathbb{N}^M$ we define:

$$\mathcal{A}_{\mathbf{u}} = \{\boldsymbol{\alpha} \in \mathcal{A} :\ \alpha_k \neq 0 \Leftrightarrow k \in \mathbf{u}, k = 1,\ldots,M\}\,. \tag{1.49}$$

In other words $\mathcal{A}_{\mathbf{u}}$ contains all the multi-indices within the truncation set $\mathcal{A}$ which have non-zero components $\alpha_k \neq 0$ if and only if $k \in \mathbf{u}$. The sum of the associated PCE terms forms a function which depends *only* on input variables $\boldsymbol{x}_{\mathbf{u}}$. The reordering of Eq. (1.48) reads:

$$\mathcal{M}_{\mathcal{A}}(\boldsymbol{x}) = y_0 + \sum_{\substack{\mathbf{u}\subset\{1,\ldots,M\}\\ \mathbf{u}\neq\varnothing}} \sum_{\boldsymbol{\alpha}\in\mathcal{A}_{\mathbf{u}}} \widehat{y}_{\boldsymbol{\alpha}}\, \Psi_{\boldsymbol{\alpha}}(\boldsymbol{x})\,. \tag{1.50}$$

The Sobol' decomposition described in Eq. (1.31) can also be written as follows, by introducing the generic index set $\mathbf{v} \overset{\text{def}}{=} \{i_1,\ldots,i_k\} \subset \{1,\ldots,M\}$ and denoting by $\boldsymbol{x}_{\mathbf{v}}$ the subvector of $\boldsymbol{x}$ obtained by extracting the components labelled by the indices in $\mathbf{v}$:

$$f(\boldsymbol{x}) = f_0 + \sum_{\substack{\mathbf{v}\subset\{1,\ldots,M\}\\ \mathbf{v}\neq\varnothing}} f_{\mathbf{v}}(\boldsymbol{x}_{\mathbf{v}})\,. \tag{1.51}$$

As mentioned in Section 1.5.3, the Sobol' decomposition is *unique*, thus the comparison of Eqs. (1.50) and (1.51) readily provides the expression of each summand for the PCE:

$$f_{\mathbf{v}}(\boldsymbol{x}_{\mathbf{v}}) = \sum_{\boldsymbol{\alpha}\in\mathcal{A}_{\mathbf{v}}} \widehat{y}_{\boldsymbol{\alpha}}\, \Psi_{\boldsymbol{\alpha}}(\boldsymbol{x})\,. \tag{1.52}$$

Due to the orthonormality of the polynomial chaos basis, the variance of the truncated PCE ($Y_{\mathcal{A}} = \mathcal{M}^{PC}(\boldsymbol{X})$) reads:

$$\mathrm{Var}\left[Y_{\mathcal{A}}\right] = \sum_{\substack{\boldsymbol{\alpha}\in\mathcal{A}\\ \boldsymbol{\alpha}\neq\mathbf{0}}} \widehat{y}_{\boldsymbol{\alpha}}^2\,, \tag{1.53}$$

$$\mathrm{Var}\left[f_{\mathbf{v}}(\boldsymbol{x}_{\mathbf{v}})\right] = \sum_{\substack{\boldsymbol{\alpha}\in\mathcal{A}_{\mathbf{v}}\\ \boldsymbol{\alpha}\neq\mathbf{0}}} \widehat{y}_{\boldsymbol{\alpha}}^2\,, \tag{1.54}$$

and the associated Sobol' indices in Eq. (1.40) is just the ratio of the above two quantities.

1.5.3.5 LRA-based Sobol' indices

The Sobol' sensitivity indices can also be computed by post-processing the coefficients of a canonical low-rank approximation (LRA) metamodel. A canonical LRA metamodel with polynomial basis has the form:

$$\widehat{y} = \mathcal{M}^{\mathrm{LRA}}(\boldsymbol{X}) = \sum_{l=1}^{R} b_l \left(\prod_{i=1}^{M}\left(\sum_{k=0}^{p_i} z_{k,l}^{(i)}\,\phi_k^{(i)}(X_i)\right)\right), \tag{1.55}$$

where $\phi_k^{(i)}$ is a univariate polynomial of degree k in the i-th input variable, and $z_{k,l}^{(i)}$ and b_l denote polynomial and weighing coefficients, respectively. For more information on canonical LRA metamodels, refer to the UQLAB User Manual – Canonical low-rank approximations.

The computation of the LRA-based Sobol' sensitivity indices relies on the following expressions for the first-order and the total indices, respectively:

$$S_i = \frac{\mathbb{E}\left[\mathbb{E}\left[\mathcal{M}(\boldsymbol{X})|\boldsymbol{X}_i\right]^2\right] - \mathbb{E}\left[\mathcal{M}(\boldsymbol{X})\right]^2}{\mathrm{Var}\left[\mathcal{M}(\boldsymbol{X})\right]} \tag{1.56}$$

and

$$S_i^T = 1 - \frac{\mathbb{E}\left[\mathbb{E}\left[\mathcal{M}(\boldsymbol{X})|\boldsymbol{X}_{\sim i}\right]^2\right] - \mathbb{E}\left[\mathcal{M}(\boldsymbol{X})\right]^2}{\mathrm{Var}\left[\mathcal{M}(\boldsymbol{X})\right]}, \tag{1.57}$$

where $\boldsymbol{X}_{\sim i}$ denotes the set of variables $\{X_j \in \boldsymbol{X} : X_j \neq X_i\}$ (see Konakli and Sudret (2016) for further details on the above equations). Note that each of Eq. (1.56) and Eq. (1.57) involves the mean and variance of the model response and additionally, the mean-square of a conditional expectation. Konakli and Sudret (2016) have shown that by substituting the model $\mathcal{M}$ with the LRA metamodel $\mathcal{M}^{\mathrm{LRA}}$, we can compute these quantities in terms of the LRA coefficients. In particular, the mean and variance of the model output are respectively approximated by:

$$\mathbb{E}\left[\mathcal{M}^{\mathrm{LRA}}(\boldsymbol{X})\right] = \sum_{l=1}^{R} b_l \left(\prod_{i=1}^{M} z_{0,l}^{(i)}\right) \tag{1.58}$$

and

$$\mathrm{Var}\left[\mathcal{M}^{\mathrm{LRA}}(\boldsymbol{X})\right] = \sum_{l=1}^{R}\sum_{l'=1}^{R} b_l\, b_{l'} \left(\left(\prod_{i=1}^{M}\left(\sum_{k=0}^{p_i} z_{k,l}^{(i)}\, z_{k,l'}^{(i)}\right)\right) - \left(\prod_{i=1}^{M} z_{0,l}^{(i)}\, z_{0,l'}^{(i)}\right)\right). \tag{1.59}$$

The quantity $\mathbb{E}\left[\mathbb{E}\left[\mathcal{M}(\boldsymbol{X})|\boldsymbol{X}_i\right]^2\right]$, appearing in the expression for S_i, is approximated by:

$$\mathbb{E}\left[\mathbb{E}\left[\mathcal{M}^{\mathrm{LRA}}(\boldsymbol{X})|\boldsymbol{X}_i\right]^2\right] = \sum_{l=1}^{R}\sum_{l'=1}^{R} b_l\, b_{l'} \left(\prod_{j\neq i}^{M} z_{0,l}^{(j)}\, z_{0,l'}^{(j)}\right)\left(\sum_{k=0}^{p_i} z_{k,l}^{(i)}\, z_{k,l'}^{(i)}\right), \tag{1.60}$$

while the quantity $\mathbb{E}\left[\mathbb{E}\left[\mathcal{M}(\boldsymbol{X})|\boldsymbol{X}_{\sim i}\right]^2\right]$, appearing in the expression for S_i^T, is approximated by:

$$\mathbb{E}\left[\mathbb{E}\left[\mathcal{M}^{\mathrm{LRA}}(\boldsymbol{X})|\boldsymbol{X}_{\sim i}\right]^2\right] = \sum_{l=1}^{R}\sum_{l'=1}^{R} b_l\, b_{l'}\, z_{0,l}^{(i)}\, z_{0,l'}^{(i)} \left(\prod_{j\neq i}\left(\sum_{k=0}^{p_j} z_{k,l}^{(j)}\, z_{k,l'}^{(j)}\right)\right). \tag{1.61}$$

Expressions for higher-order indices can also be obtained in terms of the LRA coefficients; details of the computation can be found in Konakli and Sudret (2016).

1.5.3.6 Confidence intervals

The bootstrap method (Efron, 1979) provides a simple and relatively inexpensive way of estimating confidence intervals on the estimations of the Sobol' indices. In its simplest form, bootstraping consists of resampling with replacement new points from the original sample set without additional sampling of the original model. The estimator of interest $\widehat{\theta}$ is then computed for each of the B bootstrap samples. From the resulting collection of estimators, namely $\widehat{\Theta} = \{\widehat{\theta}_1, \widehat{\theta}_2, \ldots, \widehat{\theta}_B\}$, empirical quantiles can be calculated to get confidence intervals on $\widehat{\theta}$.

1.5.4 ANCOVA indices

ANCOVA (ANalysis Of COVAriance), also called SCSA (Structural and Correlated Sensitivity Analysis), aims to produce helpful sensitivity indices for correlated input variables using a covariance decomposition method similar to the ANOVA introduced in Section 1.5.3. It is a way to generalize the formulation of the Sobol' indices for the case of dependent input variables.

The ANCOVA method was introduced by Xu and Gertner (2008). They proposed to decompose the variance contribution of X_i into two parts, one including the contributions due to independent variation and the other including the contributions due to dependence with other variables. Caniou (2012) and Sudret and Caniou (2013) proposed to use polynomial chaos expansion (PCE, see UQLAB User Manual – Polynomial Chaos Expansions) to derive a

high dimensional model representation (HDMR), originally proposed by Li et al. (2010):

$$\mathcal{M}^{PC}(\boldsymbol{x}) = \mathcal{M}_0^{PC} + \sum_{i=1}^{M} \mathcal{M}_i^{PC}(x_i) + \sum_{1\le i\le i\le M} \mathcal{M}_{ij}^{PC}(x_i, x_j) + \cdots + \mathcal{M}_{1\ldots M}^{PC}(\boldsymbol{x})\,. \tag{1.62}$$

where the $\mathcal{M}_{\boldsymbol{v}}^{PC}$ correspond to the definition provided in Eq. (1.52). By evaluating the terms of this decomposition on samples of the dependent input variables, the first-order sensitivity index can be defined as the covariance of the univariate PC component functions and the output normalized by the output variance:

$$S_i = \frac{\mathrm{Cov}\left[\mathcal{M}_i^{PC}(X_i), Y\right]}{\mathrm{Var}\left[Y\right]}\,, \tag{1.63}$$

where $\mathcal{M}_i^{PC}(X_i)$ represents the univariate component function(s) and $Y = \mathcal{M}^{PC}(\boldsymbol{X})$ represents the full PCE (see Eq. (1.62)). The covariance term in the numerator can be decomposed into:

$$S_i = \frac{\mathrm{Var}\left[\mathcal{M}_i^{PC}(X_i)\right]}{\mathrm{Var}\left[Y\right]} + \frac{\mathrm{Cov}\left[\mathcal{M}_i^{PC}(X_i), \sum_{\mathbf{v}\subset\{1,\ldots,M\}, \mathbf{v}\neq\{i\}} \mathcal{M}_{\mathbf{v}}^{PC}(\boldsymbol{X}_{\mathbf{v}})\right]}{\mathrm{Var}\left[Y\right]}\,. \tag{1.64}$$

This result provides the first-order sensitivity index of X_i as the sum of two partial variances, one including independent contributions and the other dependent ones, as was suggested by Xu and Gertner (2008). In the absence of correlation between the inputs, the second term vanishes due to the inherent orthogonality of the component functions and the first-order Sobol' index remains. In case of dependent variables, however, the second term remains.

A limitation of the decomposition in Eq. (1.31) is that it only exists in the case of independent input variables. On this account, Caniou (2012) proposed to set up a PCE assuming the input variables are independent $\boldsymbol{X}^{ind}$ and to use it as a surrogate of the full model. This PCE can be used as a predictor of the values of the HDMR in Eq. (1.62) but cannot be used to analytically derive sensitivity indices as in Section 1.5.3.5, since it will be evaluated on samples of the dependent input variables. Caniou (2012) further pointed out that in the case of dependent input variables the covariance between a univariate component function $\mathcal{M}_i^{PC}(x_i)$ and another function $\mathcal{M}_{\mathbf{v}}^{PC}(\boldsymbol{x}_{\mathbf{v}})$, where $i \in \mathbf{v}$, can originate not only from correlation but also from X_i acting in both terms. Thus, the covariance term in Eq. (1.64) consists of interactive and correlative contributions and can be further split up into two parts. The first-order index of X_i finally reads:

$$S_i = S_i^U + S_i^I + S_i^C\,, \tag{1.65}$$

where S_i^U, S_i^I and S_i^C represent the uncorrelative, interactive and correlative index of X_i, respectively. They are defined as:

$$S_i^U = \frac{\mathrm{Var}\left[\mathcal{M}_i^{PC}(X_i)\right]}{\mathrm{Var}\left[Y\right]}\,, \tag{1.66}$$

$$S_i^I = \frac{\mathrm{Cov}\left[\mathcal{M}_i^{PC}(X_i),\, \sum_{\substack{\mathbf{v}\subset\{1,\dots,M\}\\ \{i\}\in\mathbf{v}}} \mathcal{M}_{\mathbf{v}}^{PC}(\boldsymbol{X}_{\mathbf{v}})\right]}{\mathrm{Var}\left[Y\right]} \tag{1.67}$$

and:

$$S_i^C = \frac{\mathrm{Cov}\left[\mathcal{M}_i^{PC}(X_i), \sum_{\substack{\mathbf{w}\subset\{1,\dots,M\}\\ \{i\}\notin\mathbf{w}}} \mathcal{M}_{\mathbf{w}}^{PC}(\boldsymbol{X}_{\mathbf{w}})\right]}{\mathrm{Var}\left[Y\right]}\,. \tag{1.68}$$

The variances and covariances needed for the indices are computed using MC estimators. Note that the only computational cost involved in this method is the experimental design used to calculate the PCE metamodel $\mathcal{M}^{PC}$.

Splitting the first-order index into three parts aims to separate the effects of X_i in more detail than other methods. The interactive index S_i^I includes the structural influence of X_i in $\mathcal{M}_i^{PC}(X_i)$ and $\sum_{i\in\mathbf{v}} \mathcal{M}^{PC}_{\substack{\mathbf{v}\subset\{1,\dots,M\}\\ \{i\}\in\mathbf{v}}}(\boldsymbol{X}_{\mathbf{v}})$, as well as correlation effects between the component functions. Thus, some confusion between interactive and correlative influence of X_i remains. The same holds for total indices, because lower-order indices are summed up. For this reason, the definition of proper higher-order and total ANCOVA indices remains open.

Note: Since the interactive and correlative indices are defined as covariances, they can take on negative values. This is related to the fact that correlation between input variables can lead to a decrease of the output variance, which can also result in ANCOVA indices that exceed absolute values of 1. This is not an artefact of the ANCOVA decomposition, but rather one of the effects of dependence between input variables. Notably, the reduction of the variance in one input variable can result in an increase of the total variance.

1.5.5 Kucherenko indices

Kucherenko et al. (2012) proposed to define sensitivity indices using a direct decomposition of the output variance with the law of total variance. As in the case of ANCOVA, it is a way to generalize the Sobol' indices to the case of dependent input variables. If the input variables $\boldsymbol{X}$ are divided into two complementary subsets $\boldsymbol{X}_{\mathbf{v}}$ and $\boldsymbol{X}_{\mathbf{w}} = \boldsymbol{X}_{\sim\mathbf{v}}$, the total variance of $Y = \mathcal{M}(\boldsymbol{X})$ is the following sum:

$$\mathrm{Var}\left[Y\right] = \mathrm{Var}\left[\mathbb{E}\left[Y|\boldsymbol{X}_{\mathbf{v}}\right]\right] + \mathbb{E}\left[\mathrm{Var}\left[Y|\boldsymbol{X}_{\mathbf{v}}\right]\right]\,. \tag{1.69}$$

After normalization by $\mathrm{Var}\left[Y\right]$, the first summand is the *closed index* of $\boldsymbol{X}_{\mathbf{v}}$ (Eq. (1.40)), including all univariate and interaction effects between these variables:

$$S_{\mathbf{v}} = \frac{\mathrm{Var}\left[\mathbb{E}\left[Y|\boldsymbol{X}_{\mathbf{v}} = \boldsymbol{x}_{\mathbf{v}}\right]\right]}{\mathrm{Var}\left[Y\right]}\,. \tag{1.70}$$

Consequently, the second summand represents the *total effect* of the variables $\boldsymbol{X}_{\mathbf{w}}$, which

includes all univariate and interaction effects of the variables $\boldsymbol{X}_{\mathbf{w}}$ plus the interaction effects between $\boldsymbol{X}_{\mathbf{v}}$ and $\boldsymbol{X}_{\mathbf{w}}$:

$$S_{\mathbf{w}}^{T} = \frac{\mathbb{E}\left[\mathrm{Var}\left[Y|\boldsymbol{X}_{\mathbf{v}} = \boldsymbol{x}_{\mathbf{v}}\right]\right]}{\mathrm{Var}\left[Y\right]}, \quad \mathbf{w} = \sim \mathbf{v}\,. \tag{1.71}$$

This is the same definition of the sensitivity indices as in Section 1.5.3.2. However, in the presence of correlation between input variables, the indices cannot be estimated with Eq. (1.45) and the following. If $\boldsymbol{X}_{\mathbf{v}}$ is set equal to a certain value $\boldsymbol{x}_{\mathbf{v}}$, the remaining dependent variables have to be sampled conditionally to them and will be denoted as $\overline{\boldsymbol{X}}_{\mathbf{v}}$ in the following. Using this notation and the full expressions of variance and expectation, Eq. (1.70) can be rewritten as:

$$S_{\mathbf{v}} = \frac{1}{D}\left[\int_{R^{|\mathbf{v}|}} f_{\boldsymbol{X}_{\mathbf{v}}}(\boldsymbol{x}_{\mathbf{v}})\, d\boldsymbol{x}_{\mathbf{v}} \left[\int_{R^{M-|\mathbf{v}|}} \mathcal{M}(\boldsymbol{x}_{\mathbf{v}}, \overline{\boldsymbol{x}}_{\mathbf{w}}) f_{\overline{\boldsymbol{X}}_{\mathbf{w}}}(\boldsymbol{x}_{\mathbf{v}}, \overline{\boldsymbol{x}}_{\mathbf{w}})\, d\overline{\boldsymbol{x}}_{\mathbf{w}}\right]^2 - \mathcal{M}_0^2\right], \tag{1.72}$$

where $D = \mathrm{Var}\left[Y\right]$, $\mathcal{M}_0 = \mathbb{E}\left[Y\right]$, $f_{\boldsymbol{X}_{\mathbf{v}}}$ is the PDF of $\boldsymbol{X}_{\mathbf{v}}$, and $f_{\overline{\boldsymbol{X}}_{\mathbf{w}}}$ is the conditional PDF of $\boldsymbol{X}_{\mathbf{w}}$ given $\boldsymbol{X}_{\mathbf{v}} = \boldsymbol{x}_{\mathbf{v}}$.

1.5.5.1 Estimation of Kucherenko indices

Using the expression in Eq. (1.72), a Monte Carlo (MC) estimator of the first-order effect in Eq. (1.70) can be formulated:

$$S_{\mathbf{v}} = \frac{\frac{1}{N_{\mathbf{v}}}\sum_{j=1}^{N_{\mathbf{v}}}\left(\frac{1}{N_{\mathbf{w}}}\sum_{k=1}^{N_{\mathbf{w}}}\mathcal{M}(\boldsymbol{x}_{\mathbf{v},j}, \overline{\boldsymbol{x}}_{\mathbf{w},k})\right)^2 - \mathcal{M}_0^2}{D}. \tag{1.73}$$

This formula uses a *double-loop* MC estimation. It requires $N_{\mathbf{v}}$ sample points of $\boldsymbol{X}_{\mathbf{v}}$ and for each of them $N_{\mathbf{w}}$ sample points of $\overline{\boldsymbol{X}}_{\mathbf{w}}$ conditioned on $\boldsymbol{X}_{\mathbf{v}} = \boldsymbol{x}_{\mathbf{v},j}$. Additionally, it requires N samples of $\boldsymbol{X}$ to estimate the moments:

$$\mathcal{M}_0 = \frac{1}{N}\sum_{l=1}^{N}\mathcal{M}(\boldsymbol{x}_l)\,, \tag{1.74}$$

$$D = \frac{1}{N}\sum_{l=1}^{N}\mathcal{M}(\boldsymbol{x}_l)^2 - \mathcal{M}_0^2\,. \tag{1.75}$$

The estimator in Eq. (1.73) converges for large values of both $N_{\mathbf{v}}$ and $N_{\mathbf{w}}$. The final cost in terms of model evaluations is $N_{\mathrm{tot}} = N_{\mathbf{v}}N_{\mathbf{w}} + N$ and can get extremely large in realistic problems.

Kucherenko et al. (2012) estimators

Following Kucherenko et al. (2012), more efficient MC estimators of the first-order effect, which circumvents the expensive double-loop, can be formulated. To this end, a new set of notations is introduced. The samples $\boldsymbol{x} = (\boldsymbol{x}_{\mathbf{v}}, \boldsymbol{x}_{\mathbf{w}})$ and $\boldsymbol{x}' = (\boldsymbol{x}'_{\mathbf{v}}, \boldsymbol{x}'_{\mathbf{w}})$ are i.i.d. samples of $\boldsymbol{X}$. In $(\boldsymbol{x}_{\mathbf{v}}, \overline{\boldsymbol{x}}'_{\mathbf{w}})$, the overbar means that the marked sample $\overline{\boldsymbol{x}}'_{\mathbf{w}}$ is conditioned on $\boldsymbol{x}_{\mathbf{v}}$.

The first estimator, hereafter referred to as the *standard* estimator, has the following form:

$$S_{\mathbf{v}} = \frac{\frac{1}{N}\sum_{j=1}^{N}\left(\mathcal{M}(\boldsymbol{x}_{\mathbf{v},j}, \boldsymbol{x}_{\mathbf{w},j})\mathcal{M}(\boldsymbol{x}_{\mathbf{v},j}, \bar{\boldsymbol{x}}'_{\mathbf{w},j})\right) - \left(\frac{1}{N}\sum_{j=1}^{N}\mathcal{M}(\boldsymbol{x}_{\mathbf{v},j}, \boldsymbol{x}_{\mathbf{w},j})\right)^2}{D}, \tag{1.76}$$

where $\bar{\boldsymbol{x}}'_{\mathbf{w}}$ is conditioned on $\boldsymbol{x}_{\mathbf{v}}$. Using the fact that $S^T_{\mathbf{v}} = 1 - S_{\mathbf{w}}$, a more efficient estimator for the total effect can be formulated (Kucherenko et al., 2012):

$$S^T_{\mathbf{v}} = \frac{\frac{1}{N}\sum_{j=1}^{N}\left(\mathcal{M}(\boldsymbol{x}_{\mathbf{v},j}, \boldsymbol{x}_{\mathbf{w},j}) - \mathcal{M}(\bar{\boldsymbol{x}}'_{\mathbf{v},j}, \boldsymbol{x}_{\mathbf{w},j})\right)^2}{2D}. \tag{1.77}$$

In Eqs. (1.76) and (1.77), the estimators in Eqs. (1.74) and (1.75) are used. Per MC trial of $S_{\mathbf{v}}$ and $S^T_{\mathbf{v}}$, three model evaluations are required. Thus, the cost for the estimations of the first-order and total indices of M variables $\left(S_i, S^T_i\right)$, $i = 1, \ldots, M$ is $N_{\text{tot}} = N(2M+1)$. The samples $\bar{\boldsymbol{x}}'_{\mathbf{w}}$ and $\bar{\boldsymbol{x}}'_{\mathbf{v}}$ are sampled conditionally to $\boldsymbol{x}_{\mathbf{v}}$ and $\boldsymbol{x}_{\mathbf{w}}$, respectively.
Kucherenko et al. (2012) also proposed a *modified* MC estimator for the first-order effect:

$$S_{\mathbf{v}} = \frac{\frac{1}{N}\sum_{j=1}^{N}\mathcal{M}(\boldsymbol{x}_{\mathbf{v},j}, \boldsymbol{x}_{\mathbf{w},j})\left(\mathcal{M}(\boldsymbol{x}_{\mathbf{v},j}, \bar{\boldsymbol{x}}'_{\mathbf{w},j}) - \mathcal{M}(\boldsymbol{x}'_{\mathbf{v},j}, \boldsymbol{x}'_{\mathbf{w},j})\right)}{D}, \tag{1.78}$$

This modified estimator improves convergence for small indices but requires one additional model evaluation per MC trial. The total cost for the estimations of $\left(S_i, S^T_i\right)$, $i = 1, \ldots, M$ is therefore $N_{\text{tot}} = N(2M+2)$.

Sample-based estimator

The Kucherenko indices can also be estimated purely *based on an existing sample*. This is especially useful if conditional sampling, needed for the estimators in Eqs. (1.76)-(1.78), is cumbersome, or even unavailable. Sample-based estimation is performed with a similar approach to that of estimating Borgonovo indices (Section 1.5.2.1): exact conditioning on specific values $\boldsymbol{X}_{\mathbf{v}} = \boldsymbol{x}_{\mathbf{v}}$ is replaced by approximate conditioning on intervals $\boldsymbol{a}^l_{\mathbf{v}} < \boldsymbol{x}_{\mathbf{v}} < \boldsymbol{a}^u_{\mathbf{v}}$, where $\boldsymbol{a}^l_{\mathbf{v}}$ and $\boldsymbol{a}^u_{\mathbf{v}}$ are the multidimensional upper and lower bounds, respectively. The approximate conditioning is achieved by selecting subsets of the available sample $\mathcal{X}$ that belong to the relevant interval.
Given a sample of inputs $\boldsymbol{\mathcal{X}}$ and outputs $\mathcal{Y}$ of size N, the estimator of the total effect is the weighted mean of conditional variance estimates and takes the following form:

$$S^T_{\mathbf{v}} = \frac{\sum_{j=1}^{B}\frac{N_{b,j}}{N}\widehat{\text{Var}}\left[\mathcal{Y}|\boldsymbol{\mathcal{X}}_{\mathbf{w},j}\right]}{D}, \tag{1.79}$$

where B is the total number of bins, $N_{b,j}$ is the number of sample points in the j-th bin, and $\widehat{\text{Var}}\left[\mathcal{Y}|\boldsymbol{\mathcal{X}}_{\mathbf{w},j}\right]$ is the sample variance of the outputs $\mathcal{Y}_j$ that correspond to input sample points $\boldsymbol{\mathcal{X}}_{\mathbf{w}}$ that lie in the j-th bin. To compute this quantity, the input sample $\boldsymbol{\mathcal{X}}$ is binned along each conditioning set of variables $\boldsymbol{X}_{\mathbf{w}}$. In another words, $\boldsymbol{\mathcal{X}}$ is divided into B hyperrectangles (bins). Figure 4 provides a visualization of this process for the estimation of S^T_1 in the case of three input variables (X_1, X_2, X_3) in which the conditioning variables X_2 and X_3 are used to bin the sample points into B bins.

Analogously, the first-order index is the weighted variance of conditional mean estimates and can be formulated as:

$$S_{\mathbf{v}} = \frac{\sum_{j=1}^{B} \frac{N_{b,j}}{N} \left(\widehat{\mathbb{E}}\left[\mathcal{Y}|\boldsymbol{\mathcal{X}}_{\mathbf{v},j}\right] - \widehat{\mu}_{\widehat{\mathbb{E}}} \right)^2}{D}, \tag{1.80}$$

where $\widehat{\mathbb{E}}\left[\mathcal{Y}|\boldsymbol{\mathcal{X}}_{\mathbf{v},j}\right]$ is the sample mean of the outputs $\mathcal{Y}_j$ using the input sample points in which $\boldsymbol{\mathcal{X}}_{\mathbf{v}}$ lie in the j-th bin; while the term $\widehat{\mu}_{\widehat{\mathbb{E}}}$ is the weighted sample mean of the bin means $\widehat{\mathbb{E}}\left[\mathcal{Y}|\boldsymbol{\mathcal{X}}_{\mathbf{v},j}\right]$, that is

$$\widehat{\mu}_{\widehat{\mathbb{E}}} = \sum_{j=1}^{B} \frac{N_{b,j}}{N} \widehat{\mathbb{E}}\left[\mathcal{Y}|\boldsymbol{\mathcal{X}}_{\mathbf{v},j}\right]. \tag{1.81}$$

Contrary to the estimation of the total effect index, the input sample $\boldsymbol{\mathcal{X}}$ is now binned along $\boldsymbol{X}_{\mathbf{v}}$. Since all indices can be estimated based on one set of sample (by binning it in different ways), the total cost of estimation is $N_{\text{tot}} = N$ (*i.e.*, if no output sample is provided).

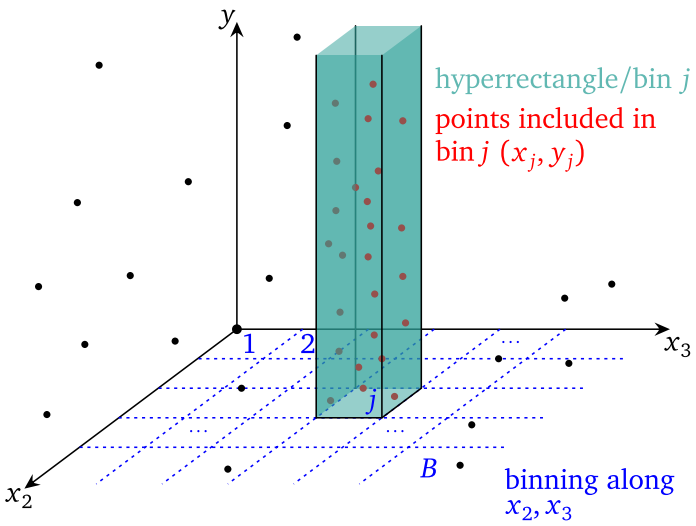

Figure 4: Approximate conditional sampling from a given sample used to calculate closed Kucherenko indices. The blue lines show the binning along $\boldsymbol{X}_{\mathbf{w}} = (X_2, X_3)$. The j-th bin / hyperrectangle is highlighted in green. The sample points lying in this bin are highlighted in red.

Chapter 2

Usage

In this section, a reference problem will be set up to showcase how each of the techniques in Chapter 1 can be deployed in UQLAB. This includes specification and running all the different sensitivity analyses, so that the corresponding results can be visualized. The following methods are showcased in this section:

- Input/output correlations: Section 2.1.2
- Standard regression coefficients (SRC/SRRC): Section 2.1.3
- Perturbation method: Section 2.1.4
- Cotter method: Section 2.1.5
- Morris elementary effects: Section 2.1.6
- Borgonovo sensitivity indices: Section 2.1.7
- Sobol' sensitivity indices: Section 2.1.8
- ANCOVA sensitivity indices: Section 2.2.2
- Kucherenko sensitivity indices: Section 2.2.3

Note: As mentioned in Section 1.2, the SRC, the Perturbation method and the Sobol' indices are only applicable in case the input variables are independent. All other methods can be used regardless of dependence between inputs.

The goal of sensitivity analysis is to identify which parameters contribute the most to the variability of the model output. To this end, we will make use of well known benchmarks for sensitivity analysis: the borehole function and the short column limit state function (see, e.g., `www.sfu.ca/~ssurjano/borehole.html` and `www.sfu.ca/~ssurjano/shortcol.html`). Methods specifically developed to deal with dependent input random vectors (*i.e.* ANCOVA and Kucherenko indices) are applied on the short column function, the other methods are applied on the borehole function.

2.1 Independent input variables: borehole function

2.1.1 Problem statement and set-up

The so-called borehole function (http://www.sfu.ca/~ssurjano/borehole.html) is an analytical 8-dimensional function that models the flow of water through a borehole with uncertain, independent parameters, given by the following equation:

$$f(\boldsymbol{x}) = \frac{2\pi T_u \left(H_u - H_l\right)}{\ln(r/r_w)\left(1 + \frac{2LT_u}{\ln(r/r_w) r_w^2 K_w} + \frac{T_u}{T_l}\right)} \tag{2.1}$$

where the description and ranges of each parameter are given in Table 1.

Table 1: Distributions and descriptions of the parameters of the borehole function in Eq. (2.1). The parameters refer to μ and σ for Gaussian distributions, to those of the underlying Gaussian for lognormal distributions and to the bounds for uniform distributions.

Name	Distributions	Parameters	Description
r_w	Gaussian	$[0.1, 0.0161812]$	radius of the borehole (m)
r	Lognormal	$[7.71, 1.0056]$	radius of influence (m)
T_u	Uniform	$[63070, 115600]$	transmissivity of upper aquifer (m^2/yr)
H_u	Uniform	$[990, 1110]$	potentiometric head of upper aquifer (m)
T_l	Uniform	$[63.1, 116]$	transmissivity of lower aquifer (m^2/yr)
H_l	Uniform	$[700, 820]$	potentiometric head of lower aquifer (m)
L	Uniform	$[1120, 1680]$	length of borehole (m)
K_w	Uniform	$[9855, 12045]$	hydraulic conductivity of borehole (m/yr)

The model in Eq. (2.1) is implemented as a Matlab `m-file` in:

`Examples/SimpleTestFunctions/uq_borehole.m`

To use it in a UQLAB analysis, we need to define a MODEL object:

```
ModelOpts.mFile = 'uq_borehole';
myModel = uq_createModel(ModelOpts);
```

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

Similarly, the distributions in Table 1 can be implemented in UQLAB as follows:

```
IOpts.Marginals(1).Name = 'rw';
IOpts.Marginals(1).Type = 'Gaussian';
IOpts.Marginals(1).Parameters = [0.10, 0.0161812];

IOpts.Marginals(2).Name = 'r';
IOpts.Marginals(2).Type = 'Lognormal';
IOpts.Marginals(2).Parameters = [7.71, 1.0056];

IOpts.Marginals(3).Name = 'Tu';
IOpts.Marginals(3).Type = 'Uniform';
IOpts.Marginals(3).Parameters = [63070, 115600];

IOpts.Marginals(4).Name = 'Hu';
IOpts.Marginals(4).Type = 'Uniform';
```

```
IOpts.Marginals(4).Parameters = [990, 1110];

IOpts.Marginals(5).Name = 'Tl';
IOpts.Marginals(5).Type = 'Uniform';
IOpts.Marginals(5).Parameters = [63.1, 116];

IOpts.Marginals(6).Name = 'Hl';
IOpts.Marginals(6).Type = 'Uniform';
IOpts.Marginals(6).Parameters = [700, 820];

IOpts.Marginals(7).Name = 'L';
IOpts.Marginals(7).Type = 'Uniform';
IOpts.Marginals(7).Parameters = [1120, 1680];

IOpts.Marginals(8).Name = 'Kw';
IOpts.Marginals(8).Type = 'Uniform';
IOpts.Marginals(8).Parameters = [9855, 12045];

myInput = uq_createInput(IOpts);
```

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

2.1.2 Input/output correlation

Specifying an input/output correlation analysis as described in Section 1.3.1 only requires the size of the sample from which to calculate the correlations. To create and run a correlation analysis based on 10,000 points, the following syntax is used:

```
CorrSensOpts.Type = 'Sensitivity';
CorrSensOpts.Method = 'Correlation';
CorrSensOpts.Correlation.SampleSize = 10000;
CorrAnalysis = uq_createAnalysis(CorrSensOpts);
```

Once the analysis is performed, a report with the sensitivity results can be printed on screen by:

```
uq_print(CorrAnalysis)
```

which produces the following:

```
-------------------------------------------
Correlation-based sensitivity indices:
-------------------------------------------

rw        r         Tu        Hu        Tl        Hl        L         Kw
0.807     -0.019    -0.007    0.311     0.003     -0.312    -0.303    0.157

-------------------------------------------
Rank-Correlation-based sensitivity indices:
-------------------------------------------

rw        r         Tu        Hu        Tl        Hl        L         Kw
0.815     -0.010    -0.005    0.312     0.006     -0.312    -0.292    0.150
```

```
Total cost (model evaluations): 10000
```

The results can be visualized graphically as follows:

```
uq_display(CorrAnalysis)
```

which produces the image in Figure 5.

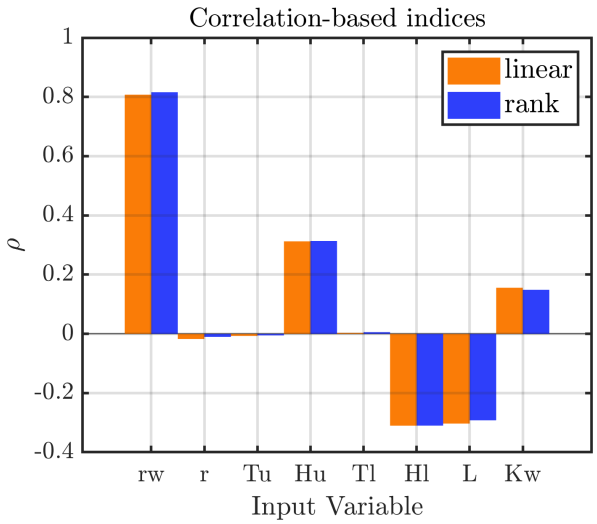

Figure 5: Input/output linear and rank correlation results from the analysis described in Section 2.1.2.

Note: In the preceding example only the sample size for the analysis is provided. If not further specified the input/output correlation analysis runs with the following default options:

- Sampling method: `'LHS'`.

2.1.2.1 Accessing the results

The analysis results can be accessed in the `CorrAnalysis.Results` structure:

```
CorrAnalysis.Results
ans =
  CorrIndices: [8x1 double]
  RankCorrIndices: [8x1 double]
  ExpDesign: [1x1 struct]
  Cost: 10000
  VariableNames: {'rw'  'r'  'Tu'  'Hu'  'Tl'  'Hl'  'L'  'Kw'}
```

In the structure, the `CorrIndices` and `RankCorrIndices` fields are arrays containing the indices in Eq. (1.1) and (1.4), respectively. The `Cost` field represents the number of model evaluations needed to calculate the indices, in this case equal to the sample size. A full description of the format of the output can be found in Table 22.

2.1.2.2 Advanced options

The input/output correlation method is extremely simple. Thus there are only a few options available for configuration. The most relevant are related to the generation of the sample from which the coefficients are computed. In the following, two typical advanced usage scenarios are presented:

- **Specify the sampling scheme**: by default, model realizations are sampled via simple Monte Carlo from the input distributions. It is possible, however, to specify a different sampling scheme (*e.g.* Latin Hypercube Sampling or Sobol' pseudo-random sequences), by adding the following option:

  ```
  CorrSensOpts.Correlation.Sampling = 'LHS';
  ```

  Any of the sampling schemes supported by the default input module can be specified (see Section 3.2, Page 37 of the UQLAB User Manual – the INPUT module).

- **Manually provide the samples:** if input samples are already available and no additional sampling is needed, it is possible to use them directly for the analysis. In case the samples are available in variables `X_ED` and the corresponding model evaluations in variable `Y_ED`, they can be added as:

  ```
  CorrSensOpts.Correlation.Sample.X = X_ED;
  CorrSensOpts.Correlation.Sample.Y = Y_ED;
  ```

  Note that in this case no `SampleSize` needs to be specified.

For a comprehensive list of the options available to the Correlation method, see Table 4.

2.1.3 Standard Regression Coefficients

Standard regression coefficients (SRC) and standard rank-regression coefficients (SRRC), described in Section 1.3.2 only require a single configuration option: the size of the sample on which to perform regression.
To create and run a SRC analysis based on a sample of 10,000 points, the following syntax is used:

```
SRCSensOpts.Type = 'Sensitivity';
SRCSensOpts.Method = 'SRC';
SRCSensOpts.SRC.SampleSize = 10000;
SRCAnalysis = uq_createAnalysis(SRCSensOpts);
```

Once the analysis is performed, a report with the sensitivity results can be printed on screen by:

```
uq_print(SRCAnalysis)
```

which produces the following:

```
-----------------------------------------
```

```
Standard Regression sensitivity indices:
-----------------------------------------

rw        r         Tu        Hu        Tl        Hl        L         Kw
0.804     -0.004    -0.001    0.306     0.002     -0.310    -0.295    0.148

-----------------------------------------
Standard Rank-Regression sensitivity indices:
-----------------------------------------

rw        r         Tu        Hu        Tl        Hl        L         Kw
0.810     -0.003    0.001     0.310     0.001     -0.306    -0.292    0.144
-----------------------------------------
```

The results can be visualized graphically as follows:

```
uq_display(SRCAnalysis)
```

which produces the image in Figure 6.

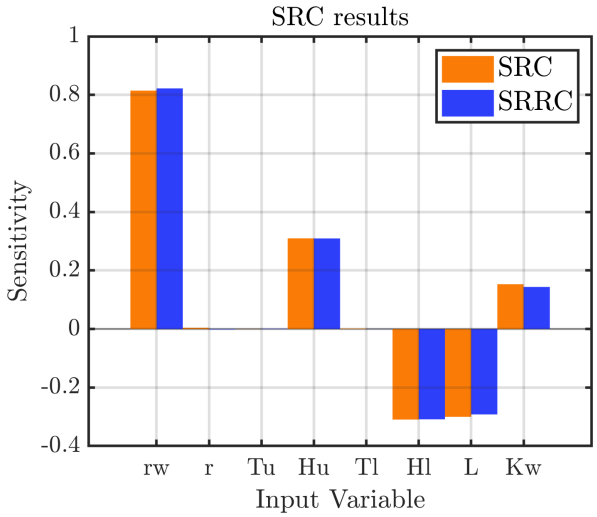

Figure 6: SRC and SRRC sensitivity results from the analysis described in Section 2.1.3.

Note: In the preceding example only the sample size for the analysis is provided. If not further specified the standard regression analysis runs with the following default options:

- Sampling method: `'LHS'`.

2.1.3.1 Accessing the results

The analysis results can be accessed in the `SRCAnalysis.Results` structure:

```
SRCAnalysis.Results
ans =
  SRCIndices: [1x8 double]
  SRRCIndices: [1x8 double]
  Cost: 1000
```

```
VariableNames: {'rw'  'r'  'Tu'  'Hu'  'Tl'  'Hl'  'L'  'Kw'}
```

In the structure, the `SRCIndices` and `SRRCIndices` fields are arrays containing the indices in Eq. (1.8) and (1.11), respectively. The `Cost` field represents the number of model evaluations needed to calculate the indices, in this case equal to the sample size. A full description of the format of the output can be found in Table 23.

2.1.3.2 Advanced options

In the following, two typical advanced usage scenarios of the SRC/SRRC method are presented:

- **Specify the sampling scheme**: by default, model realizations are sampled via simple Monte Carlo from the input distributions. It is possible, however, to specify a different sampling scheme (*e.g.* Latin Hypercube Sampling or Sobol' pseudorandom sequences), by adding the following option:

  ```
  SRCSensOpts.SRC.Sampling = 'Sobol';
  ```

  Any of the sampling schemes supported by the default input module can be specified (see Section 3.2, Page 37 of the UQLAB User Manual – the INPUT module).

- **Manually provide the samples:** if module samples are already available and no additional sampling is needed, it is possible to use them directly for the analysis. In case the samples are available in variables `X_ED` and the corresponding model evaluations in variable `Y_ED`, they can be added as:

  ```
  SRCSensOpts.SRC.Sample.X = X_ED;
  SRCSensOpts.SRC.Sample.Y = Y_ED;
  ```

  Note that in this case no `SampleSize` needs to be specified.

For a comprehensive list of options available to the SRC method, see Table 6.

2.1.4 Perturbation method

Perturbation-based sensitivity analysis (Section 1.4.1) does not require any configuration options since it is based on the numerical evaluation of the gradient of the model around the mean value. To create and run a perturbation analysis, the following syntax is used:

```
PerturbationSensOpts.Type = 'Sensitivity';
PerturbationSensOpts.Method = 'Perturbation';
PerturbationAnalysis = uq_createAnalysis(PerturbationSensOpts);
```

Once the analysis is performed, a report with the sensitivity results can be printed on screen by:

```
uq_print(PerturbationAnalysis)
```

which produces the following:

```
-----------------------------------------
Perturbation-based sensitivity indices:
-----------------------------------------

rw        r         Tu        Hu        Tl        Hl        L         Kw
0.699     0.000     0.000     0.095     0.000     0.095     0.088     0.022
-----------------------------------------
```

The results can be visualized graphically as follows:

```
uq_display(PerturbationAnalysis)
```

which produces the image in Figure 7.

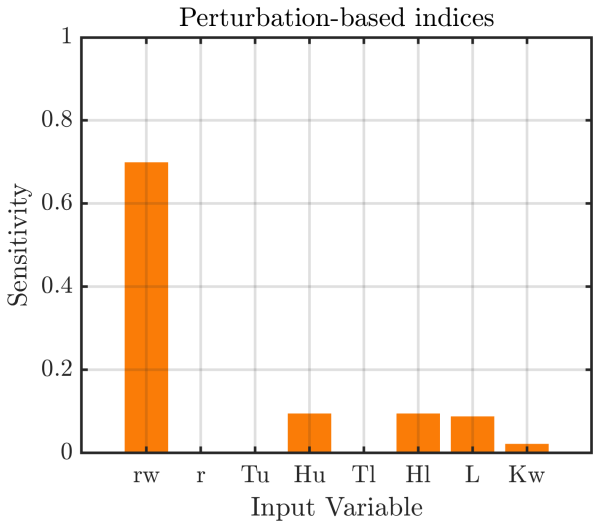

Figure 7: Perturbation-based sensitivity results from the analysis described in Section 2.1.4.

Note: In the preceding example no specifications are provided. The perturbation method runs with the following default options:

- Gradient steps: `'relative'`;
- Relative step size: 10^{-3};
- Finite difference scheme: `'forward'` .

2.1.4.1 Accessing the results

The analysis results can be accessed in the `PerturbationAnalysis.Results` structure:

```
PerturbationAnalysis.Results
ans =
  Mu: 7.093190986636091e+01
  Var: 7.541825411751066e+02
  Sensitivity: [1x8 double]
  Cost: 9
  VariableNames: {'rw'  'r'  'Tu'  'Hu'  'Tl'  'Hl'  'L'  'Kw'}
```

In the structure, the `Mu` and `Var` fields report the estimated mean and variance of the model output, while the `Sensitivity` field is the array containing the indices in Eq. (1.15). The `Cost` field represents the number of model evaluations needed to calculate the indices, in this case equal to the cost of evaluating partial derivatives of the model. A full description of the format of the output can be found in Table 24.

2.1.4.2 Advanced options

The advanced options for the perturbation method are related to the numerical computation of the gradient of the model. They allow one to:

- **Specify the gradient method**: by default, the gradient is calculated with the "forward" method (to minimize the number of model evaluations). It can be set to any of "Forward","Centered" (more accurate, but more expensive to calculate) or "Backwards" as follows:

```
PerturbationSensOpts.Gradient.Method = 'Centered';
```

- **Specify the gradient step mode and its value:** by default, the gradient is calculated numerically in the standardized space (*i.e.*in units of standard deviation), and the finite difference step is set to $h = 10^{-3}$. It is possible, however to specify both the mode and the value as follows:

```
% specify the FD gradient step as absolute (not normalized)
PerturbationSensOpts.Gradient.Step = 'absolute';
% and give it a value of 0.0001
PerturbationSensOpts.Gradient.h = 0.0001;
```

Note that no `SampleSize` needs to be specified. The `'absolute'` option should be used with caution when the magnitude of the various input parameters varies from one another.

For a comprehensive list of options available to the perturbation method, see Table 8.

2.1.5 Cotter Measure

The Cotter measure does not require any specific configuration option, because the input bounds (see Section 1.4.2) are set by default to the following values, depending on the input distributions:

- *Uniform, beta and truncated distributions:* $x_i^{\pm}$ match the lower and upper bounds of the corresponding input distribution.
- *Lognormal distribution:* $x_i^- = \max(\mu_i - \sigma_i, 0)$, $x_i^+ = \mu_i + \sigma_i$.
- *Unbounded distributions:* $x_i^- = \mu_i - \sigma_i$, $x_i^+ = \mu_i + \sigma_i$.

where μ_i and σ_i are the mean and standard deviation of variable X_i, respectively, as defined in the INPUT object (see Section 2.1.1). The Cotter sensitivity analysis is then created as follows:

```
CotterSensOpts.Type = 'Sensitivity';
CotterSensOpts.Method = 'Cotter';
CotterAnalysis = uq_createAnalysis(CotterSensOpts);
```

Once the analysis is performed, a report with the sensitivity results can be printed on screen by:

```
uq_print(CotterAnalysis)
```

which produces the following:

```
-----------------------------------------
Cotter sensitivity indices:
-----------------------------------------

rw        r         Tu        Hu        Tl        Hl        L         Kw
20.991    43.879    0.000     18.157    0.171     18.157    21.787    7.948
-----------------------------------------
```

The results can also be visualized graphically as follows:

```
uq_display(CotterAnalysis)
```

which produces the image in Figure 8.

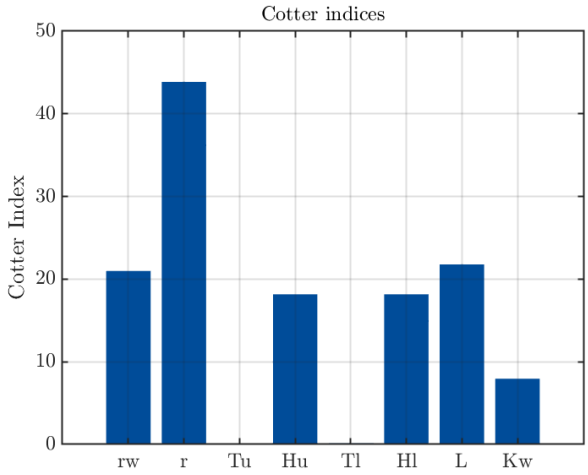

Figure 8: Cotter sensitivity results from the analysis described in Section 2.1.5.

2.1.5.1 Accessing the results

The analysis results can be accessed in the `CotterAnalysis.Results` structure:

```
CotterAnalysis.Results
ans =
  CotterIndices: [1x8 double]
  EvenOrder: [1x8 double]
  OddOrder: [1x8 double]
  Cost: 18
  VariableNames: {'rw'  'r'  'Tu'  'Hu'  'Tl'  'Hl'  'L'  'Kw'}
```

The `CotterIndices` field is the array containing the indices in Eq. (1.16), while the `OddOrder` and `EvenOrder` arrays contain the odd and even order effects in Eq. (1.17) and (1.18), respectively. The `Cost` field represents the number of model evaluations needed to calculate the indices, in this case $(2M + 2)$, where M is the number of input variables.
A full description of the format of the output can be found in Table 25.

2.1.5.2 Advanced options

The Cotter method only allows for the customization of the low and high factor bounds $[x_i^-, x_i^+]$ (see Section 1.4.2). The bounds can be specified in two ways: absolute or relative to the standard deviation of the inputs. The behaviour of the factors can be specified in the `CotterOptions.Factors` structure as follows:

- **Absolute bounds:** they can be specified by setting the 1×2 array `CotterOptions.Factors(ii).Boundaries` that contains lower and upper bounds for each variable. As an example, to set all of the M inputs factors low and high level to 0 and 1, respectively, the following code can be used:

```
for ii = 1:M
  CotterSensOpts.Factors(ii).Boundaries = [0 1];
  end
```

- **Relative bounds:** bounds can be specified relative to the standard deviation according to the following equation:

$$x_i^{\pm} = \mu_i \pm k_i \sigma_i \tag{2.2}$$

where k_i is a positive scalar, μ_i and σ_i are the $i-$th factor mean and standard deviations specified as in Section 2.1.1. There are two ways to specify the k_i:

  - If the `Factors` structure is a 1-element structure and `Factors.Boundaries` is a scalar, all the k_i are set to its value, *e.g.*:

```
CotterSensOpts.Factors.Boundaries = 0.5;
```

    will set all $k_1 = \cdots = k_m = 0.5$.

  - If the `Factors` structure has M elements and each `Factors(ii).Boundaries` is a scalar b_i, the $k_i = b_i$, *e.g.*:

```
for ii = 1:M
CotterSensOpts.Factors(ii).Boundaries = ii;
end
```

    will manually set each of the $k_i = i$.

For a comprehensive list of options available to the Cotter method, see Table 9.

2.1.6 Morris Method

The Morris method only requires one configuration option, namely the maximum number of full-model evaluations to be performed to estimate the elementary effects. Hence, a Morris analysis with default settings and a maximum of 10^4 model evaluations can be configured as follows:

```
MorrisSensOpts.Type = 'Sensitivity';
MorrisSensOpts.Method = 'Morris';
MorrisSensOpts.Morris.Cost = 10000;
MorrisAnalysis = uq_createAnalysis(MorrisSensOpts);
```

Note that the `MorrisSensOpts.Morris.Cost` represents the maximum allowed cost. UQLAB will use this constraint conservatively, so that the effective cost is lower than or equal to the specified value. Once the analysis is performed, a report with the sensitivity results can be printed on screen by:

```
uq_print(MorrisAnalysis)
```

which produces the following:

```
-------------------------------------------------
Morris sensitivity indices:
-------------------------------------------------
       rw      r        Tu      Hu      Tl      Hl       L        Kw
```

```
mu:     43.23   21.669   0.000   27.87   0.204   -27.91   -27.22   13.452
mu*:    43.23   21.799   0.000   27.87   0.204   27.911   27.226   13.452
sigma:  15.09   58.792   0.000   9.740   0.121   9.699    10.918   5.0250
-------------------------------------------------------
Total cost (model evaluations): 9999
```

The results can also be visualized graphically as follows:

```
uq_display(MorrisAnalysis)
```

which produces the image in Figure 9.

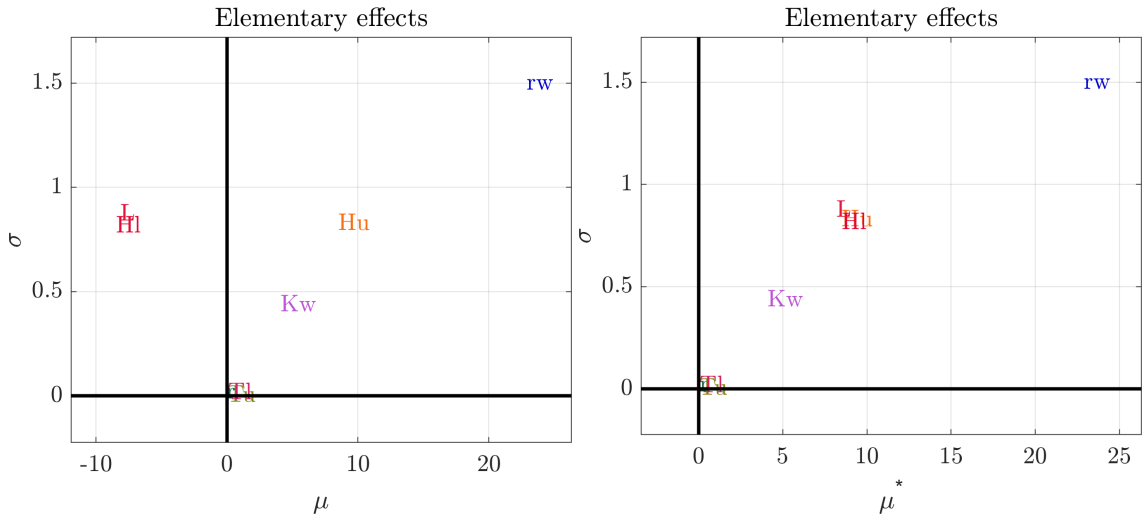

Figure 9: Morris sensitivity results from the analysis described in Section 2.1.6.

The results in Figure 9 are shown with the standard Morris format: the x-axis represents the mean value of the elementary effects related to each variable, while the y-axis represents the corresponding standard deviation. Values close to the origin (lower left corner) represent unimportant parameters. Parameters far from the origin in the x-direction are important, while a large y-coordinate means that the parameter has a strong degree of non-linearity/interaction.

Note: In the preceding example only the maximum cost for the analysis is provided. If not further specified the Morris method runs with the following default options:

- Number of grid levels: $2 \times \lceil \frac{M}{2} \rceil$ (closest even number of the number of input variables);
- Perturbation steps: one-half of the number of grid levels.

2.1.6.1 Accessing the results

The analysis results can be accessed in the `MorrisAnalysis.Results` structure:

```
MorrisAnalysis.Results
ans =
  Mu: [1x8 double]
```

```
MuStar: [1x8 double]
Std: [1x8 double]
Cost: 9999
VariableNames: {'rw'  'r'  'Tu'  'Hu'  'Tl'  'Hl'  'L'  'Kw'}
```

The mean value of the elementary effects for each variable is given in the `Mu` field, while the corresponding standard deviation in the `Std` field (see Eq. (1.22)). The `MuStar` field corresponds instead to the mean of the modified estimator in Eq. (1.25), especially useful for ranking purposes. The `Cost` field represents instead the effective number of model evaluations used to calculate the elementary effects. A full description of the format of the output can be found in Table 26.

2.1.6.2 Advanced options

There are several parameters that can be fine-tuned in a Morris' sensitivity analysis. They include:

- **Min-max boundaries of the parameters:** they are similar to the boundaries for the Cotter measure (see Section 2.1.5.2). They can be specified with the `SensOpts.Factors` field, with the same syntax and the same default values as for the Cotter sensitivity case.
- **Perturbation grid:** the input space between the variable boundaries is explored through a regular grid as defined in Eqs. (1.19) to (1.21). Its parameters can be specified as follows:

  ```
  MorrisSensOpts.GridLevels = 6;
  MorrisSensOpts.PerturbationSteps = 3;
  ```

  where `GridLevels` and `PerturbationSteps` correspond to p in Eq. (1.19) and $\bar{c}$ in Eq. (1.21), respectively. Note that according to Morris (1991), `GridLevels` should be equal to twice of `PerturbationSteps`.

The number of replications for each elementary effect (variable r, in Section 1.5.1) is determined directly from the specified `Cost` according to:

$$cost = r(M+1), \qquad r = \left\lfloor \frac{cost}{M+1} \right\rfloor \tag{2.3}$$

where $cost$ is the value specified in Section 2.1.6.
For a comprehensive list of options available to the Morris method, see Table 10.

2.1.7 Borgonovo indices

In order to set up a Borgonovo analysis only the size of the used sample needs to be specified. As mentioned, a large amount of samples is recommended in order to achieve converging results:

```
BorgonovoOpts.Type = 'Sensitivity';
BorgonovoOpts.Method = 'Borgonovo';
```

```
BorgonovoOpts.Borgonovo.SampleSize = 10000;
BorgonovoAnalysis = uq_createAnalysis(BorgonovoOpts);
```

Once the analysis is performed, a report with the sensitivity results can be printed on screen by:

```
uq_print(BorgonovoAnalysis);
```

which produces the following:

```
-------------------------------------------------
Borgonovo indices for output component 1
-------------------------------------------------
rw        r         Tu        Hu        Tl        Hl        L         Kw
0.415     0.058     0.054     0.117     0.048     0.132     0.121     0.078
-------------------------------------------------
Total cost (model evaluations): 10000
```

The results can be visualized graphically as follows:

```
uq_display(BorgonovoAnalysis);
```

which produces the image in Figure 10.

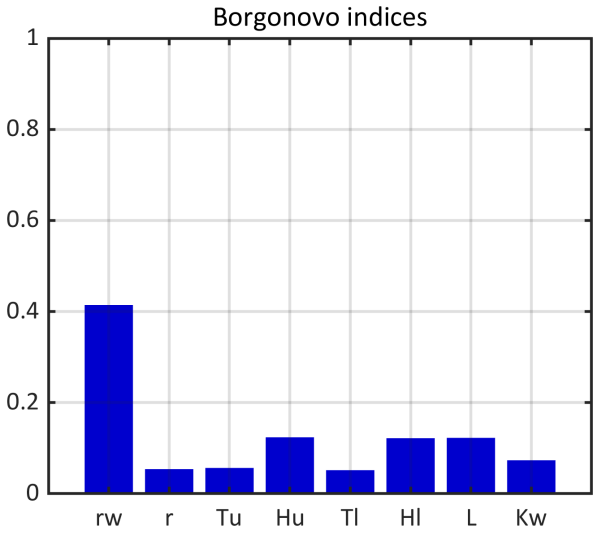

Figure 10: Borgonovo indices for the borehole model.

Note: In the preceding example only the sample size for the analysis is provided. If not further specified the Borgonovo analysis runs with the following default options:

- Sampling method: `'LHS'`;
- Number of classes: 20;
- Binnig strategy to create classes: `'Quantile'`
- Overlap between neighbouring classes: 2%;
- Method to estimate the inner integral: `'HistBased'`;
- Number of bins in the histograms: *chosen separately for each class* (`'auto'`).

2.1.7.1 Accessing the results

The analysis results can be accessed in the `BorgonovoAnalysis.Results` structure:

```
BorgonovoAnalysis.Results

ans =
Delta: [8x1 double]
JointPDF: {8x1 cell}
Cost: 100000
ExpDesign: [1x1 struct]
VariableNames: {'rw'  'r'  'Tu'  'Hu'  'Tl'  'Hl'  'L'  'Kw'}
```

The sensitivity indices can be found in `Delta`. The estimated joint PDF is also stored in the field `JointPDF`.

2.1.7.2 Advanced Options

In order to qualitatively assess the accuracy of the computation of a Borgonovo index, it is useful to inspect the estimated joint probability distribution function of the model output with the corresponding variable. For example, the estimate of the joint PDF f_{Y,X_1} can be visualized by:

```
uq_display(BorgonovoAnalysis,1,'Joint PDF',1);
```

In this advanced `uq_display` command, the second argument is the index of the model output, the third one is a flag determining the type of visualization and the last is the input variable index. The previous command produces the plot that is shown in Figure 11.
The Borgonovo analysis allows for many additional options to be specified, which can be found in Section 3.1.6:

- **CDF-based estimation of the inner integral:** instead of the default histogram-based estimation method `'HistBased'`, the CDF-based estimation method in Eq. (1.30) can be chosen:

  ```
  BorgonovoOpts.Borgonovo.Method = 'CDFBased';
  ```

- **Specify the binning strategy for the classes:** by default UQLAB creates non-constant classes of the X_i values by taking the quantiles of f_{X_i} into account. This binning strategy is labeled `'Quantile'`. In order to use constant width classes specify:

  ```
  BorgonovoOpts.Borgonovo.BinStrat = 'Constant';
  ```

- **Specify the amount and overlap of classes (in the X_i-direction):** by default UQLAB uses 20 classes with a 2% overlap. The user can modify both values:

  ```
  BorgonovoOpts.Borgonovo.NClasses = 15;
  BorgonovoOpts.Borgonovo.Overlap = 0.1;
  ```

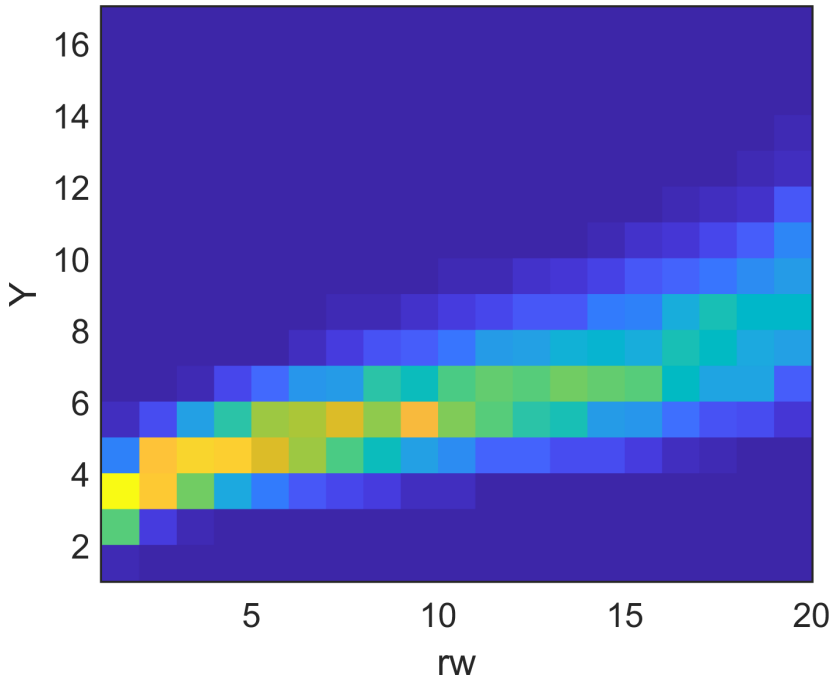

Figure 11: A histogram estimate of the joint PDF of Y and $X_1 = r_w$. This is the estimate used to compute the conditional distributions $f_{Y|X_1}$ in the computation of the corresponding index.

Note that `Overlap` specifies the ratio of samples that overlap between two neighbouring classes and therefore must be between 0 and 1.

- **Specify the amount of bins for the unconditional histogram (in the Y-direction):** by default the number of bins for the histogram of f_Y is optimally chosen for each class by the `histcounts` MATLAB function. To fix the amount of bins to a certain number specify:

```
BorgonovoOpts.Borgonovo.NHistBins = 60;
```

Note: it is important that the number of samples in each class is sufficiently large to allow the binning in the specified amount of bins. If the chosen amount of bins is too large for the available samples in a class, the histograms will not be an accurate depiction of the conditional distribution which, consequently, will lead to wrong estimates of the indices.

- **Provide the samples to be analysed:** instead of sampling, the user can provide samples to be used directly. If input and output samples are stored in variables `X` and `Y`, they can be provided:

```
BorgonovoOpts.Borgonovo.Sample.X = X;
BorgonovoOpts.Borgonovo.Sample.Y = Y;
```

Note that in this case `SampleSize` is set equal to the length of the provided samples.

For a complete account of the available computational options, refer to Section 3.1.6.

2.1.8 MC-based Sobol' indices

The Sobol' measure only requires one configuration option, namely the number of model evaluations needed to calculate *each index*. Hence, a Sobol' analysis with default settings (Total + first order indices only, Janon estimator in Eq. (1.47)) and a maximum of $10,000$ model evaluations per dimension can be configured as follows:

```
SobolSensOpts.Type = 'Sensitivity';
SobolSensOpts.Method = 'Sobol';
SobolSensOpts.Sobol.SampleSize = 10000;
SobolAnalysis = uq_createAnalysis(SobolSensOpts);
```

Note that the `SobolSensOpts.Sobol.SampleSize` represents the number of model evaluations used *for each index*. This is important as the number of indices that can be calculated increases with the input dimension. The increase is linear for total and first order indices, but increases factorially with the order of the indices. Once the analysis is performed, a report with the sensitivity results can be printed on screen by:

```
uq_print(SobolAnalysis)
```

which produces the following:

```
-------------------------------------------------
Total Sobol' indices for output component 1
-------------------------------------------------
rw        r         Tu        Hu        Tl        Hl        L         Kw
0.693     0.000     0.000     0.107     0.000     0.107     0.102     0.025
-------------------------------------------------
-------------------------------------------------
First Order Sobol' indices for output component 1
-------------------------------------------------
rw        r         Tu        Hu        Tl        Hl        L         Kw
0.661     0.001     0.001     0.097     0.001     0.096     0.090     0.024
-------------------------------------------------
Total cost (model evaluations): 1000000
```

The results can also be visualized graphically as follows:

```
uq_display(SobolAnalysis)
```

which produces the images in Figure 12.

Note: In the preceding example only the sample size for the analysis is provided. If not further specified the Sobol' analysis runs with the following default options:

- Sample size: $10,000$;
- Sampling method: `'MC'`;
- Estimator: `'Janon'`;
- Calculation of Sobol' indices up to order: 1.

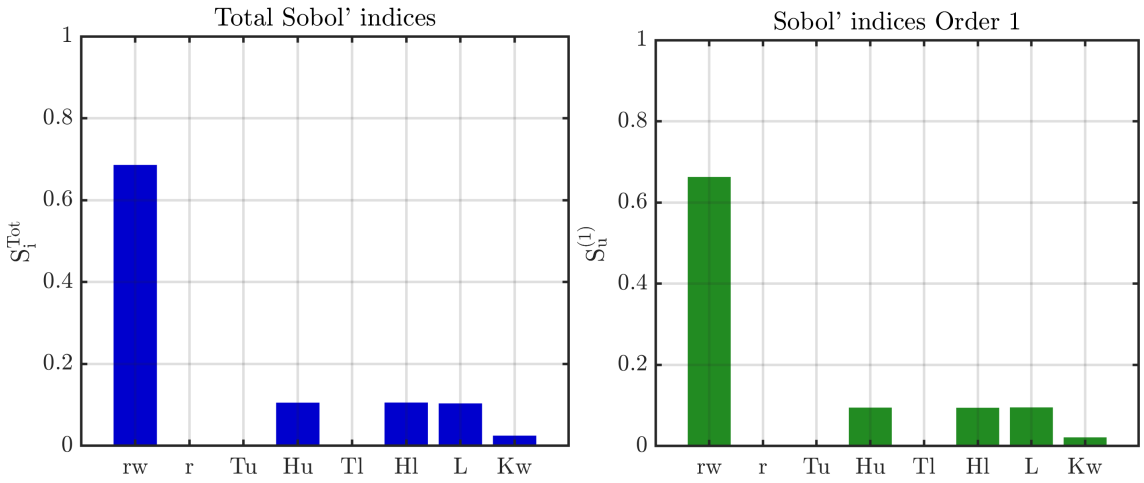

Figure 12: Sobol' sensitivity results from the analysis described in Section 2.1.8. Left: total Sobol' indices; Right: first order sobol' indices.

2.1.8.1 Accessing the results

The analysis results can be accessed in the `SobolAnalysis.Results` structure:

```
SobolAnalysis.Results
ans =
AllOrders: {[8x1 double]}
Total: [8x1 double]
FirstOrder: [8x1 double]
VarIdx: {[8x1 double]}
TotalVariance: 813.6991
Cost: 100000
ExpDesign: [1x1 struct]
CoefficientBased: 0
VariableNames: {'rw'  'r'  'Tu'  'Hu'  'Tl'  'Hl'  'L'  'Kw'}
```

The `Total` and `FirstOrder` arrays contain the total and first order Sobol' indices, respectively. All the indices (excluding the total Sobol' indices but including the first order ones) are grouped by order in the `AllOrders` *cell array*. The value of the indices of order j are given as column vectors in `AllOrders{j}`. The corresponding list of variables for each order is given in the `VarIdx{j}` array. To access the first order indices and the corresponding list of variables in our example, *e.g.*:

```
SobolAnalysis.Results.AllOrders{1}
ans =
  0.6439
  -0.0176
  -0.0176
  0.0800
  -0.0177
  0.0846
  0.0667
  0.0061
```

```
SobolAnalysis.Results.VarIdx{1}
ans =
  1
  2
  3
  4
  5
  6
  7
  8
```

Note: because the number of indices depends both on M and on the index order, `AllOrders` and `VarIdx` are MATLAB *cell arrays*, hence their contents must be accessed with the `{}` notation.

As in previous cases, `Cost` gives the total cost (in model evaluations) of the analysis. `TotalVariance` is an array that contains the total variance of the model response. The `CoefficientBased` flag is `0` if the indices are calculated via Sampling as in Section 1.5.3 or `1` if they are calculated via PCE expansion as in Section 1.5.3.4. Finally, the `ExpDesign` field contains information about the model evaluations used to calculate the Sobol' indices.

For a complete overview of the output format of the Sobol' analysis module, refer to Table 28.

2.1.8.2 Advanced Options

Several options can be fine-tuned for the calculation of Sobol' indices. The most common customization options are reported below:

- **Specify the maximum Sobol' index order:** the maximum Sobol' index order can be specified with the `Sobol.Order` option:

```
SobolSensOpts.Sobol.Order = 2;
```

- **Specify an estimator:** to use the estimator T in Eq. (1.47), set the following:

```
SobolSensOpts.Sobol.Estimator = 'T';
```

- **Specify the sampling scheme:** the Monte-Carlo sampling scheme can be specified via the `Sobol.Sampling` option. Note that all of the options Section 3.2, Page 37 of the UQLAB User Manual – the INPUT moduleare valid:

```
SobolSensOpts.Sobol.Sampling = 'LHS';
```

- **Calculate confidence intervals:** bootstrap-based confidence intervals can be calculated by UQLAB by specifying the `SobolSensOpts.Boostrap` option. The following code creates a bootstrap resampling with $B = 100$ samples and calculates confidence bounds corresponding to the 0.025 and 0.975 quantiles:

```
SobolSensOpts.Bootstrap.Replications = 100;
SobolSensOpts.Alpha = 0.05;
```

If bootstrap is used, an additional `Bootstrap` field is added to `SobolAnalysis.Results`:

```
SobolAnalysis.Results.Bootstrap
ans =
Total: [1x1 struct]
FirstOrder: [1x1 struct]
AllOrders: {[1x1 struct]}
```

Details on how the Bootstrap results are reported are given in Table 29.

- **Do not store the model evaluations:** by default, the full model evaluations that are run to calculate the Sobol' indices are saved, as they can be expensive to calculate. However, when higher order indices are calculated from inexpensive models, a very large number of model evaluations may be generated (order of 10^9), hence creating taxing storage needs. It is possible, however, to remove the model evaluations after the index calculation is complete, by setting:

```
SobolSensOpts.SaveEvaluations = false;
```

For a complete reference list of options available for the calculation of Sobol' indices, refer to Table 14.

2.1.9 PCE-based Sobol' indices

If a polynomial chaos expansion (PCE) metamodel created by UQLAB is available, it is possible to calculate Sobol' indices *analytically* from its coefficients. A comparison between Sobol' indices calculated with Sampling (see Section 1.5.3) and directly from PCE (see Section 1.5.3.4) for the borehole model is provided in the file:
`Examples/Sensitivity/uq_Example_Sensitivity_02_Sobol.m`.

The problem is set up as in Section 2.1.1, but, in addition, a PCE metamodel is created before defining the sensitivity analysis. This can be accomplished as follows (for details on how to use the UQLAB's metamodelling toolbox, refer to the Section 2 of the UQLAB User Manual – Polynomial Chaos Expansions). The following code creates a sufficiently accurate PCE of the borehole model based on an experimental design of 200 samples.

```
%  Select the 'metamodel tool' in UQLab
PCEOpts.Type = 'Metamodel';
%  Choose the Polynomial Chaos Expansion module
PCEOpts.MetaType = 'PCE';
%  PCE requires an input to be defined
PCEOpts.Input = myInput;
%  As well as a full model, if the Experimental Design is not provided
PCEOpts.FullModel = myModel;
%  Run 5th degree PCE (default: sparse PCE expansion)
PCEOpts.Degree = 5;
%  Specify the experimental design size (actual cost of the metamodel)
PCEOpts.ExpDesign.NSamples = 200;
```

```
%  Calculate the PCE coefficients
myPCE = uq_createModel(PCEOpts);
```

To create a second order Sobol' analysis the following is sufficient:

```
PCESobol.Type = 'Sensitivity';
PCESobol.Method = 'Sobol';
PCESobol.Sobol.Order = 2;
PCESobolAnalysis = uq_createAnalysis(PCESobol);
```

Note that there is no necessity to specify that the Sobol' indices should be calculated from the PCE coefficients: if the current model is of type "PCE metamodel", PCE-based Sobol' indices will be calculated. A comparison between Monte Carlos-based and PCE-based Sobol' indices is shown in Figure 13.

2.1.9.1 Accessing the results

The results are stored in the same format as for MC-based Sobol' indices, see Section 2.1.8.1.

```
PCESobolAnalysis.Results
ans =
  AllOrders: {[8x1 double]}
  Total: [8x1 double]
  FirstOrder: [8x1 double]
  TotalVariance: 804.2971
  VarIdx: {[8x1 double]}
  VariableNames: {'rw'  'r'  'Tu'  'Hu'  'Tl'  'Hl'  'L'  'Kw'}
  PCEBased: 1
```

Note that the `PCEBased` flag is now automatically set to 1 and that there are no `ExpDesign` nor `Cost` fields, because no new model evaluations are run.

2.1.9.2 Advanced Options

Except for the Sobol' order, that is specified as in Section 2.1.8.2, only one option can be specified for PCE-based Sobol' analysis:

- **Force Monte-Carlo based estimation of the indices:** for comparison/validation, it is possible to force the calculation of the Sobol' indices through MC estimator as in Section 2.1.8. The `PCEBased` flag is then specified as follows:

  ```
  PCESobol.Sobol.PCEBased = 0;
  ```

2.1.10 LRA-based Sobol' indices

As explained in Section 1.5.3.5, the Sobol' indices can also be computed by post-processing the coefficients of a canonical low-rank approximation (LRA) metamodel. A comparison between Sobol' indices calculated with Sampling and directly from LRA for the borehole model is provided in the file:

`Examples/Sensitivity/uq_Example_Sensitivity_02_Sobol.m`.

The problem is set up as in Section 2.1.9, but using a canonical LRA metamodel instead of a PCE one. The following code creates a sufficiently accurate LRA of the borehole model based on an experimental design of 200 samples.

```
% Select the metamodel tool in UQLab
LRAOpts.Type = 'Metamodel';
% Choose the low-rank approximation module
LRAOpts.MetaType = 'LRA';
% Specify the input and full model for LRA
LRAOpts.Input = myInput;
LRAOpts.FullModel = myModel;
% Specify the rank  and degree ranges
LRAOpts.Rank = 1:20;
LRAOpts.Degree = 1:20;
% Specify the experimental design size (total cost of the metamodel)
LRAOpts.ExpDesign.NSamples = 200;
% Calculate the LRA coefficients
myLRA = uq_createModel(LRAOpts);
```

To create a second order Sobol' analysis the following is sufficient:

```
LRASobol.Type = 'Sensitivity';
LRASobol.Method = 'Sobol';
LRASobol.Sobol.Order = 2;
LRASobolAnalysis = uq_createAnalysis(LRASobol);
```

Note that if the current model is of type "LRA metamodel", LRA-based Sobol' indices will be calculated analytically from the expansion coefficients. A comparison between Monte Carlo-based and LRA-based Sobol' indices is shown in Figure 13.

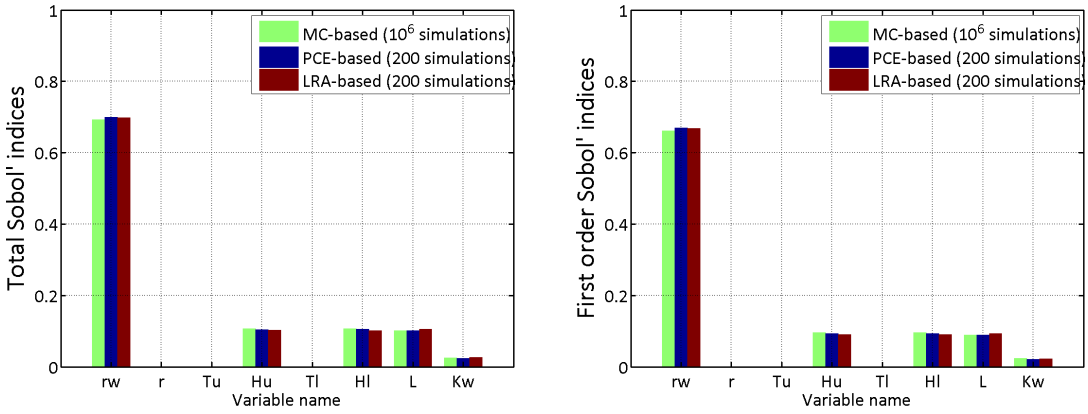

Figure 13: Comparison between MC-based, PCE-based and LRA-based Sobol indices for the borehole model. Left: total Sobol' indices; Right: first order Sobol' indices.

The results are stored in the same format as for MC-based and PCE-based Sobol' indices (see Sections 2.1.8.1 and 2.1.9.1).
Similarly to the PCE-based Sobol analysis, it is possible to force Monte-Carlo based calculation of the Sobol' indices even with a LRA meta-model. This is specified with the command:

```
LRASobol.Sobol.LRABased = 0;
```

2.2 Dependent input variables: short column function

2.2.1 Problem statement and set-up

The so-called short column function (http://www.sfu.ca/~ssurjano/shortcol.html) is an analytical 3-dimensional function that models the limit state of a short steel column with a rectangular ($b = 5$ mm, $h = 15$ mm) cross-section that is subjected to a bending moment and an axial force, given by the following equation:

$$g(\boldsymbol{x}) = 1 - \frac{4M}{bh^2Y} - \frac{P^2}{b^2h^2Y^2} \tag{2.4}$$

where the description and the ranges of the non-constant parameters are given in Table 2.

Table 2: Distributions and descriptions of the parameters of the short column function in Eq. (2.4). The parameters refer to μ and σ for Gaussian distributions and to those of the associated Gaussian for the lognormal distribution.

Name	Distributions	Parameters	Description
Y	Lognormal	$[5, 0.5]$	yield stress (MPa)
M	Gaussian	$[2000, 400]$	bending moment (Nmm)
P	Gaussian	$[500, 100]$	axial force (N)

The axial force and the bending moment are assumed to be *dependent*. The dependence is modeled by a Gaussian copula with a copula parameter of 0.72 between the two input variables M and P.

The model in Eq. (2.4) is implemented as a MATLAB `m-file` in:

`Examples/SimpleTestFunctions/uq_shortcol.m`

To use it in a UQLAB analysis, we need to configure a MODEL module:

```
ModelOpts.mFile = 'uq_shortcol';
myModel = uq_createModel(ModelOpts);
```

The distributions in Table 2 and the dependence can be implemented as follows:

```
% Marginals
ModelOpts.Marginals(1).Name = 'Y';
ModelOpts.Marginals(1).Type = 'Lognormal';
ModelOpts.Marginals(1).Parameters = [5 0.5];

ModelOpts.Marginals(2).Name = 'M';
ModelOpts.Marginals(2).Type = 'Gaussian';
ModelOpts.Marginals(2).Parameters = [2000 400];

ModelOpts.Marginals(3).Name = 'P';
ModelOpts.Marginals(3).Type = 'Gaussian';
ModelOpts.Marginals(3).Parameters = [500 100];
```

```
% Dependence
ModelOpts.Copula.Type = 'Gaussian';
ModelOpts.Copula.Parameters = [ 1 0 0;...
                                0 1 0.72;...
                                0 0.72 1];

myInput = uq_createInput(ModelOpts);
```

2.2.2 ANCOVA indices

To run an ANCOVA analysis for the problem introduced in Section 2.2.1, one specifies:

```
ANCOVASensOpts.Type = 'Sensitivity';
ANCOVASensOpts.Method = 'ANCOVA';
ANCOVASensOpts.ANCOVA.SampleSize = 150;
ANCOVAAnalysis = uq_createAnalysis(ANCOVASensOpts);
```

When performing the analysis, UQLAB will create a PCE metamodel (required by the method, see Section 1.5.4) with default options. Advanced PCE Options can also be set manually, see Section 2.2.2.2. Once the analysis is performed, a report with the results can be printed on screen by:

```
uq_print(ANCOVAAnalysis)
```

which produces the following:

```
-------------------------------------------------
ANCOVA indices for output component 1
-------------------------------------------------
Indices     Y           M           P
S           0.852       0.106       0.013
S^U         0.837       0.096       0.001
S^I         0.005       -0.005      0.000
S^C         0.009       0.016       0.012
-------------------------------------------------
Total cost (model evaluations): 150 (for the experimental design)
```

The results can also be visualized graphically as follows:

```
uq_display(ANCOVAAnalysis)
```

which produces the plots in Figure 14.

Note: In this example only the size of the experimental design was provided. If not further specified, the ANCOVA runs with the following default options:

- Size of the experimental design for the PCE: 200;
- Sampling strategy: `'LHS'`;
- Number of samples used for the ANCOVA: 10,000;
- PCE of degree `1-10` based on `'LARS'` (see also UQLAB User Manual – Polynomial Chaos Expansions).

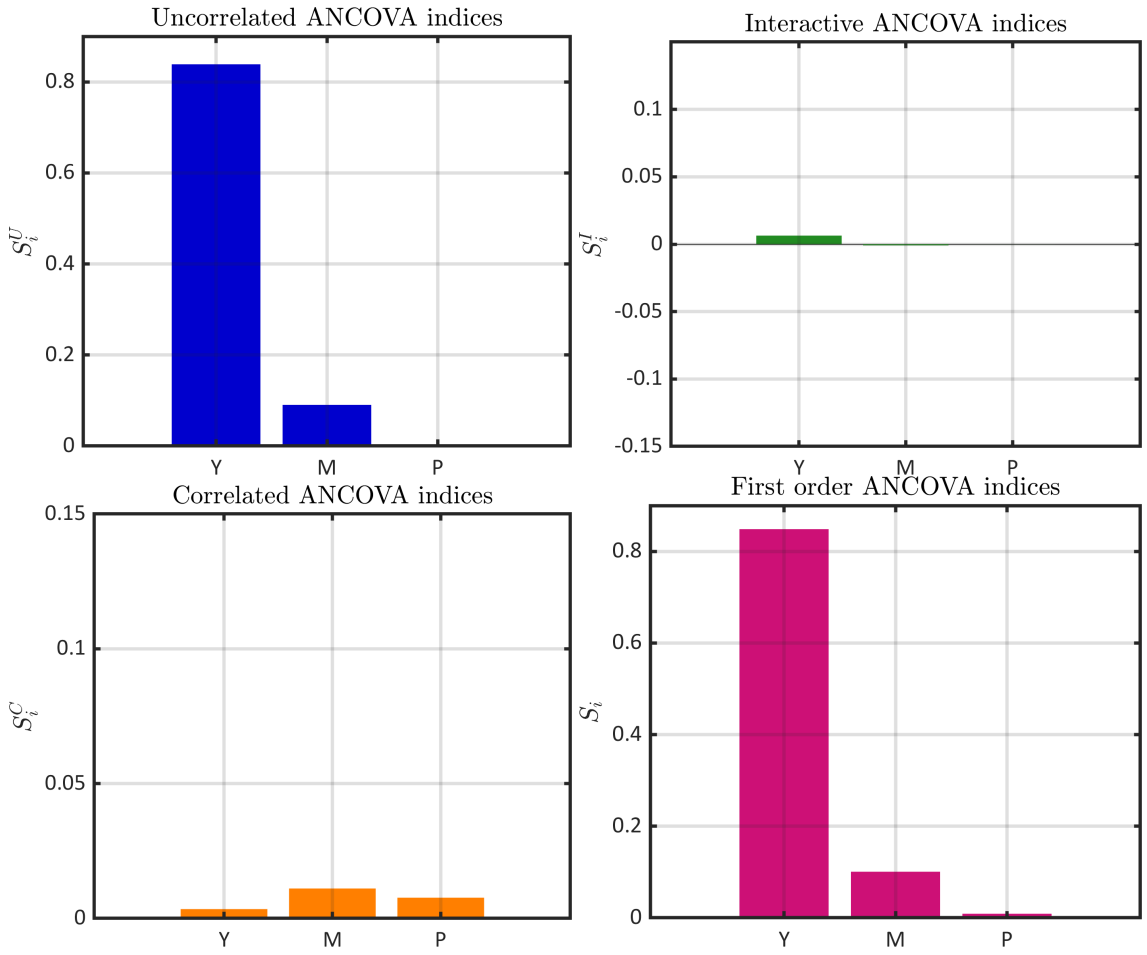

Figure 14: ANCOVA sensitivity results from the analysis described in Section 2.2.2.

2.2.2.1 Accessing the results

The results of the ANCOVA can be accessed in the `ANCOVAAnalysis.Results` structure:

```
ANCOVAAnalysis.Results
ans =
PCE: [1x1 uq_model]
Cost: 150
TotalVariance: 0.0013
Uncorrelated: [3x1 double]
Interactive: [3x1 double]
Correlated: [3x1 double]
FirstOrder: [3x1 double]
VariableNames: {'Y'  'M'  'P'}
```

The `Uncorr`, `Interact` and `Corr` arrays contain the uncorrelated, interactive and correlated ANCOVA indices, respectively. The array `FirstSum` contains the first-order indices (the sum of the aforementioned indices) per variable and output. The `Cost` field shows the number of model runs and is equal to the size of the experimental design. The used PCE metamodel is stored in the `PCE` field.

2.2.2.2 Advanced Options

The ANCOVA analysis requires the set-up of a PCE metamodel and sampling of the dependent variables (see Section 1.5.4). Different specifications are possible:

- **Specify the sampling strategy of the input variables:** if desired, one can specify the sampling strategy for the input variables using the `ANCOVASensOpts.ANCOVA.Sampling` option. Note that all of the options in Section 3.2, Page 37 of the UQLAB User Manual – the INPUT moduleare valid.

```
ANCOVASensOpts.ANCOVA.Sampling = 'LHS';
```

- **Specify the number of samples used for the ANCOVA:** the number of samples used for the Monte Carlo estimation of the indices can be specified:

```
ANCOVASensOpts.ANCOVA.MCSamples = 10000;
```

  Note that these samples are used only with the PCE, which does not increase the computational cost in terms of full model evaluations.

- **Provide the experimental design for the set-up of the PCE:** the main cost of the method stems from the full model evaluations that are needed for the experimental design. If a sample set of the input random vector and the corresponding model evaluations are available, they can be provided as the experimental design for the PCE:

```
ANCOVASensOpts.ANCOVA.Samples.X = X_ED;
ANCOVASensOpts.ANCOVA.Samples.Y = Y_ED;
```

- **Specify the set-up of the PCE metamodel:** to ensure the success of the ANCOVA method, UQLAB replaces the provided model by a PCE metamodel. For this set-up, several options are available (see also UQLAB User Manual – Polynomial Chaos Expansions). For example, to specify an adaptive degree range of $5:10$ and to force the ordinary least squares method to compute the coefficients, the following shall be used:

```
ANCOVASensOpts.ANCOVA.PCE.Degree = 5:10;
ANCOVASensOpts.ANCOVA.PCE.Method = 'OLS';
```

- **Use an existing UQLab PCE metamodel:** by default UQLAB replaces the provided model by a PCE before the ANCOVA analysis is performed, even if the provided model already is a PCE metamodel. If the user wishes to provide a PCE metamodel `myPCE` that should be used directly in order to avoid the cost arising from the set-up, the latter can be specified as follows:

```
ANCOVASensOpts.ANCOVA.CustomPCE = myPCE;
```

> **Note:** If this option is used, it is essential that the provided PCE is a UQLAB PCE object created with `uq_createModel` (see UQLAB User Manual – Polynomial Chaos Expansions). Any other MODEL object provided in `ANCOVASensOpts.Model` will be ignored.

For a complete reference list of options available for the calculation of ANCOVA indices, refer to Section 3.1.10.

2.2.3 Kucherenko indices

The main parameter for the Kucherenko analysis is the number of sample points N to be used. The total cost of the analysis depends on the chosen estimator (see Section 1.5.5.1):

```
KucherenkoSensOpts.Type = 'Sensitivity';
KucherenkoSensOpts.Method = 'Kucherenko';
KucherenkoSensOpts.Kucherenko.SampleSize = 1e4;
KucherenkoAnalysis = uq_createAnalysis(KucherenkoSensOpts);
```

Once the analysis is performed, a report with the results can be printed on screen by:

```
uq_print(KucherenkoAnalysis)
```

which produces the following:

```
-------------------------------------------------
Total Kucherenko indices for output component 1
-------------------------------------------------
Y           M           P
0.884794    0.056842    0.001734
-------------------------------------------------
-------------------------------------------------
First Order Kucherenko indices for output component 1
-------------------------------------------------
Y           M           P
0.843087    0.048177    0.050116
-------------------------------------------------
Total cost (model evaluations): 80000
```

The results can also be visualized graphically as follows:

```
uq_display(KucherenkoAnalysis)
```

which produces the images in Figure 15.

> **Note:** In this example, only the number of samples was specified. If nothing else is specified, the Kucherenko analysis runs with the following default options:
>
> - Number of samples used for the Kucherenko analysis: 10,000;
> - Sampling strategy: `'LHS'`;
> - Estimator: `'Modified'`.

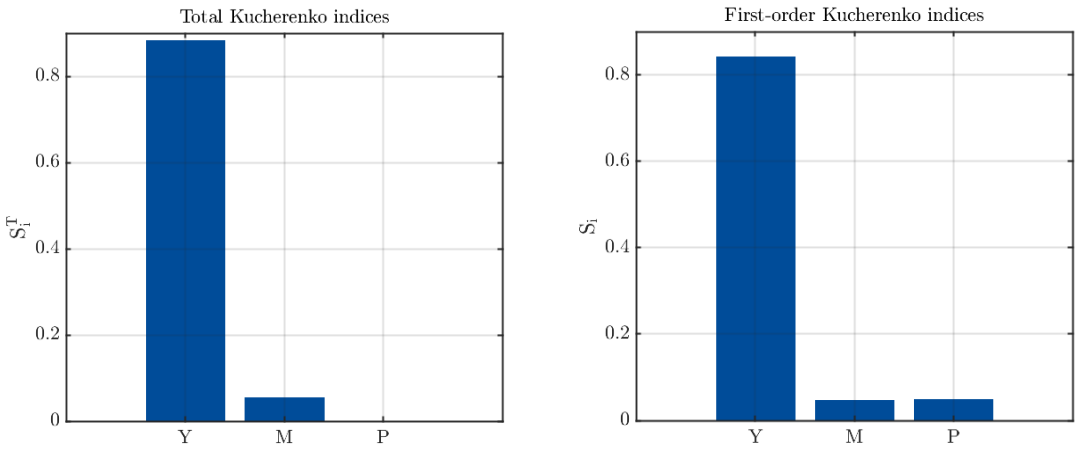

Figure 15: Kucherenko sensitivity results from the analysis described in Section 2.2.3. Left: total Kucherenko indices; right: first order Kucherenko indices.

2.2.3.1 Accessing the results

The results of the Kucherenko analysis are stored in `KucherenkoAnalysis.Results`, similarly to the other global sensitivity analyses.

```
ans =

  struct with fields:

      FirstOrder: [3x1 double]
           Total: [3x1 double]
   TotalVariance: 0.0013
            Cost: 80000
       ExpDesign: [1x1 struct]
   VariableNames: {'Y'  'M'  'P'}
```

2.2.3.2 Advanced Options

The following advanced options can be configured by the user:

- **Specify the first-order index estimator:** different estimators were presented in Section 1.5.5. The `'Standard'` estimator can be specified as follows:

```
KucherenkoSensOpts.Kucherenko.Estimator = 'Standard';
```

- **Provide a sample to estimate the indices :** by specifying the `'SampleBased'` estimator (see sample-based estimation, Section 1.5.5.1), the user can provide the input `X` (*i.e.*, $\mathcal{X}$) and (optionally) the output samples `Y` (*i.e.*, $\mathcal{Y}$) used for the estimation:

```
KucherenkoSensOpts.Kucherenko.Samples.X = sampleX;
KucherenkoSensOpts.Kucherenko.Samples.Y = sampleY;
```

In case only `X` is specified, `sampleY = uq_evalModel(sampleX)` will be automatically calculated.

In alternative, the user can also define the size of the sample to be generated and used for the estimation:

```
KucherenkoSensOpts.Kucherenko.SampleSize = 1e5;
```

For a complete reference list of options available for the calculation of Kucherenko indices, refer to Section 3.1.11.

2.3 Sensitivity analysis of models with multiple outputs

2.3.1 Introduction

If a model has a vector-valued response $\boldsymbol{Y} = \mathcal{M}(\boldsymbol{X}) \in \mathbb{R}^{N_{out}}$, UQLAB will treat each of the N_{out} outputs independently and automatically. Full-model evaluations are of course performed only once, regardless of the number of output components. Hence, no changes in the syntax are needed to run the analyses presented throughout this chapter.
An example analysis calculating Sobol' indices on a multi-component model (deflection of a simply supported beam at multiple points) can be found in:
`Examples/Sensitivity/uq_Example_Sensitivity_04_MultipleOutputs.m`

2.3.2 Accessing multi-output results

Results are stored in the same format as described in the previous section, with the only difference that each output is stored in a different column of the results array. Consider as an example the Sobol' indices results in Section 2.1.8.1. If the number of output components were increased to $N_{out} = 5$, the results structure would read:

```
SobolAnalysis.Results
ans =
  AllOrders: {[8x5 double]}
  Total: [8x5 double]
  FirstOrder: [8x5 double]
  VarIdx: {[8x1 double]}
  TotalVariance: [1x5 double]
  Cost: 100000
  ExpDesign: [1x1 struct]
  PCEBased: 0
  VariableNames: {'rw'  'r'  'Tu'  'Hu'  'Tl'  'Hl'  'L'  'Kw'}
```

The only difference with respect to the scalar model output case ($N_{out} = 1$) lies in the fact that now all of the result arrays have one column for each output.
Note: the `VarIdx` cell array for Sobol' analysis is independent on the output component, hence its dimension is unchanged, as it corresponds to $\mathbf{v}$ in Eq. (1.40).

2.4 Excluding parameters from the analysis

In some analyses, one may want to perform sensitivity analysis on a reduced set of input variables only. This can be important for methods like Morris', MC-based Sobol' or Kucherenko sensitivity indices, whose costs (in terms of model evaluations) increase significantly with the number of input variables. There are two ways to achieve this within UQLAB: using a parameter index (factor masking) or using constant input variables.
This process is transparent to the user as the analysis results will still show the excluded variables, but their sensitivities will be set to 0. Whenever applicable, UQLAB will automatically and appropriately account for the set of input parameters which were declared constant so as to avoid unnecessary model evaluations.

2.4.1 Using a factor index

Sensitivity calculation for selected factors can be included/excluded explicitly by means of the `SensOpts.FactorIndex` configuration variable. `SensOpts.FactorIndex` is a logical index of size $(1 \times M)$ that is `1` for the factors that are to be included and `0` those that are to be excluded. As an example, to remove the calculation of the first index in the MC-based Sobol' analysis in Section 2.1.8, one can write:

```
SobolSensOpts.FactorIndex = [0 1 1 1 1 1 1 1];
```

which results in the following `SobolAnalysis.Results` structure:

```
SobolAnalysis.Results
ans =
  AllOrders: {[8x1 double]}
  Total: [8x1 double]
  FirstOrder: [8x1 double]
  VarIdx: {[8x1 double]}
  TotalVariance: 240.4744
  Cost: 90000
  ExpDesign: [1x1 struct]
  CoefficientBased: 0
  VariableNames: {'rw'  'r'  'Tu'  'Hu'  'Tl'  'Hl'  'L'  'Kw'}
```

Comparing to the same printout in Section 2.1.8, all output structures have the same format. However, the total cost is lower ($90,000$ vs $100,000$ model runs), as well as the total variance.

2.4.2 Using constant input variables

In some analyses, one may need to assign a constant value to one or to a set of parameters input. When this is the case, the sensitivity analysis is built in UQLAB by appropriately setting the `SensOpts.FactorIndex` parameter so as to exclude the constant variables from the analysis.
To set a parameter to constant, the following command can be used when the probabilistic input is defined (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 sensitivity analysis

Syntax

```
myAnalysis = uq_createAnalysis(SOpts)
```

Input

The Struct variable `SOpts` contains the configuration information for a sensitivity analysis. The detailed list of options available is reported in Table 3.

Table 3: `SOpts`

●	`.Type`	`'Sensitivity'`	Identifier of the sensitivity module.
●	`.Method`	String	Sensitivity analysis method. The available options are listed below:
		`'Correlation'`	I/O correlation method (Section 2.1.2).
		`'SRC'`	Standard Regression Coefficients (Section 2.1.3).
		`'Perturbation'`	Perturbation Method (Section 2.1.4).
		`'Cotter'`	Cotter Method (Section 2.1.5).
		`'Morris'`	Morris method (Section 2.1.6).
		`'Borgonovo'`	Borgonovo indices (Section 2.1.7).
		`'Sobol'`	Sobol' indices (Section 2.1.8, 2.1.9 and 2.1.10).
		`'ANCOVA'`	ANCOVA indices (Section 2.2.2).
		`'Kucherenko'`	Kucherenko indices (Section 2.2.3).
☐	`.Input`	`INPUT` object	Specification of the used input object for the analysis. If not specified, the currently active one is used.
☐	`.Model`	`MODEL` object	Specification of the used model object for the analysis. If not specified, the currently active one is used.
☐	`.FactorIndex`	$1 \times M$ Logical default: `true(1,M)`	Vector with M elements that contains `true` if the factor must be taken into account or `false` if it should be ignored in the computations. Because the total computational costs are related to the number of factors, this option can help save model evaluations.
☐	`.SaveEvaluations`	Logical default: 1	Store or not the performed model evaluations.

		1	Model evaluations are saved.
		0	Model evaluations are not saved.
☐	`.Display`	String default: `'standard'`	Level of information displayed by the methods.
		`'quiet'`	Minimum display level, displays nothing or very few information.
		`'standard'`	Default display level, shows the most important information.
		`'verbose'`	Maximum display level, shows all the information on runtime, like updates on iterations, etc.
☐	`.Gradient`	See Table 8	Define the options for computing the gradient. It applies only to the perturbation method, when `SOpts.Method = 'Perturbation'`.
☐	`.Factors`	See Table 9	Defines the intervals for input parameters. It applies only when `SOpts.Method = 'Cotter'` or `SOpts.Method = 'Morris'`.
☐	`.Morris`	See Table 10	Define the options for the Morris method. It applies only when `SOpts.Method = 'Morris'`.
☐	`.Sobol`	See Table 14, Table 16 and Table 17	Define the options for the Sobol' indices. It applies only when `SOpts.Method = 'Sobol'`.
☐	`.Borgonovo`	See Table 12	Define the options for the Borgonovo indices. It applies only when, `SOpts.Method = 'Borgonovo'`.
☐	`.ANCOVA`	See Table 18	Define the options for the ANCOVA indices. It applies only when `SOpts.Method = 'ANCOVA'`.
☐	`.Kucherenko`	See Table 20	Define the options for the Kucherenko indices. It applies only when `SOpts.Method = 'Kucherenko'`.

3.1.1 Input/Output correlation options

For details on the usage, please refer to Section 2.1.2.

Table 4: `SOpts.Correlation`			
☐	`.Sampling`	String default: `'LHS'`	Sampling method to generate the sample. All methods in Table 11, page 37 of the UQLAB User Manual – the INPUT module are supported.
☐	`.Sample`	See Table 5	Manual specification of samples.
☐	`.SampleSize`	Integer	Number of samples to be generated.

Table 5: `SOpts.Correlation.Sample`			
●	`.X`	$N \times M$ Double	Manually provided input sampling.
●	`.Y`	$N \times N_{out}$ Double	Manually provided model response.

3.1.2 Standard Regression Coefficients (SRC and SRRC) options

For details on the usage, please refer to Section 2.1.3.

Table 6: `SOpts.SRC`			
●	`.SampleSize`	Integer	Number of samples to be generated.
☐	`.Sampling`	String default: `'LHS'`	Sampling method to generate the sample. All methods in Table 11, page 37 of the UQLAB User Manual – the INPUT module are supported.
☐	`.Sample`	See Table 7	Manually provided input sampling.

Table 7: `SOpts.SRC.Sample`			
●	`.X`	$N \times M$ Double	Manually provided input sampling.
●	`.Y`	$N \times N_{out}$ Double	Manually provided model response.

3.1.3 Perturbation Method options

For details on the usage, please refer to Section 2.1.4.

Table 8: `SOpts.Gradient`			
☐	`.Step`	String default: `'relative'` `'relative'`	Defines how the perturbation h used in the approximation is calculated. h is an amount relative to the standard deviation of each variable $h_i = (\texttt{Gradient.h}) \cdot \sigma_i$

		`'absolute'`	h is defined as an absolute value.
☐	`.h`	Double default: 10^{-3}	Value of the difference for the scheme. If `Gradient.Step` is set to `'relative'`, the difference in each coordinate is computed as $h_i = (\texttt{Gradient.h}) \cdot \sigma_i$
☐	`.Method`	String default: `'forward'`	Specifies the type of finite differences scheme to be used.
		`'forward'`	Forward finite differences. Two model evaluations $\mathcal{M}(\boldsymbol{x})$ and $\mathcal{M}(x_1, \ldots, , x_i + h_i, x_{i+1}, \ldots, x_M)$ are used.
		`'backward'`	Same as forward, but the increment is backwards, $\mathcal{M}(\boldsymbol{x} - h)$.
		`'centred'` or `'centered'`	Evaluates the model at two points in the neighbourhood of the point of interest, but not at the point itself. It is more accurate, but it needs more model evaluations.

3.1.4 Cotter Measure options

For details on the usage, please refer to Section 2.1.5.

Table 9: `SOpts.Factors(i)`

●	`.Boundaries`	1×2 Double	lower and upper boundaries for the i-th variable $[x_i^-, x_i^+]$
		Double default: 1	If `Factors(i).Boundaries` is a single scalar k, the intervals for each factor are as $x_i^{\pm} = \mu_i \pm k\sigma_i$

3.1.5 Morris Measure

For details on the usage, please refer to Section 2.1.6.

Table 10: `SOpts.Morris`			
⊕	`.Cost`	Integer	Maximum number of evaluations allowed for the Morris method, *i.e.*, Cost $= (M+1)r$.
⊕	`.FactorSamples`	Integer	Number of replications r for each elementary effect, *i.e.*, Cost $= (M+1)r$.
☐	`.GridLevels`	Integer default: `2×PerturbationSteps` if the latter is defined, $2 \times \lceil \frac{M}{2} \rceil$ otherwise	Number of points used to discretize each interval where the factors vary.
☐	`.PerturbationSteps`	Integer default: $\lfloor \frac{\texttt{GridLevels}}{2} \rfloor$	Number of grid levels used for computing each elementary effect.
☐	`.Factors(i)`	See Table 11	Minimum and maximum boundaries for the i-th parameter.

Table 11: `SOpts.Morris.Factors(i)`			
☐	`.Boundaries`	1×2 Double	Absolute lower and upper boundaries for the i-th parameter (*factor*), *i.e.*, $[x_i^-, x_i^+]$.
		Double default: 1.0	A scalar that defines the relative boundaries for the i-th parameter (similar to the one for the Cotter method, Eq. (2.2)). If a scalar k_i is defined for each parameter, the upper and lower boundaries for the i-th parameter are $x_i^{\pm} = \mu_i \pm k_i \sigma_i$, where μ_i and σ_i are the mean and standard deviation of the i-th parameter. If a single scalar k is defined for all parameters (*i.e.*, `SOpts.Factors.Boundaries = k`), the upper and lower boundaries for the i-th parameter are $x_i^{\pm} = \mu_i \pm k\sigma_i$.

3.1.6 Borgonovo indices

For details on the usage, please refer to Section 2.1.7.

Table 12: `SOpts.Borgonovo`			
⊕	`.SampleSize`	Integer default: 10000	Specify the number of random samples.
☐	`.Sampling`	String default: `'LHS'`	Sampling strategy for generating the samples. (see Table 14)
⊕	`.Sample`	See Table 13	Provide the samples on which the analysis will be conducted.
☐	`.Sampling`	String default: `'LHS'`	Sampling strategy for generating the samples. (see Table 14)
☐	`.BinStrat`	String default: `'Quantile'`	Specify the binning strategy for the creation of the classes.
		`'Quantile'`	Classes are created based on quantiles to ensure enough samples in each class.
		`'Constant'`	Classes have a constant width.
☐	`.NClasses`	Integer default: 20	Specify the amount of classes (in X_i-direction).
☐	`.Overlap`	Double default: 0.02	Specify the ratio of shared samples between two neighbouring classes (in X_i-direction).
☐	`.NHistBins`	Integer or String default: `'auto'`	Specify the amount of bins for the histogram (in Y-direction).
		`'auto'`	Automatically choses a suitable amount of bins for each class.
☐	`.Method`	String default: `'HistBased'`	Specify the method used for the estimation of the inner integral.
		`'HistBased'`	Estimation based on the absolute differences between histograms.
		`'CDFBased'`	Estimation based on approximated CDF's.

Table 13: `SOpts.Borgonovo.Sample`			
●	`.X`	$N \times M$ Double	Manually provided input sample.
●	`.Y`	$N \times N_{out}$ Double	Manually provided model response.

3.1.7 MC-based Sobol' indices

For details on the usage, please refer to Section 2.1.8.

Table 14: `SOpts.Sobol'` (MC-Sampling indices)			
●	`.SampleSize`	Integer default: 1000	Size of each of the samples to be generated. The total number of evaluations is $N(M+2)$, for a problem of dimension M.
☐	`.Estimator`	String default: `'t'`	The type of estimator used to approximate the Sobol' indices.
		`'sobol'`, `'classic'`	Original estimator presented in Sobol' (1993), based on equations (1.42)-(1.44).
		`'homma'`, `'s'`	The estimator presented in Saltelli and Homma (1996), as described in (1.46).
		`'janon'`, `'t'`	Estimator proposed by Janon et al. (2014), described in equation (1.47). It is the default estimator.
☐	`.Order`	Integer default: 1	Maximum order of the interactions for which the Sobol' indices are estimated.
☐	`.Sampling`	String default: `'mc'`	Sampling strategy for generating the initial samples.
		`'halton'`	The samples are created with Halton pseudo-random sampling.
		`'lhs'`	The samples are created with Latin Hypercube sampling.
		`'mc'`	The samples are created with Monte Carlo random sampling.
		`'sobol'`	The samples are created with Sobol' pseudo-random sampling.
☐	`.Bootstrap`	See Table 15	Specify the options for generating bootstrap confidence intervals for the estimates.

3.1.7.1 Boostrap error estimate on the Sobol' indices

For details on the usage, please refer to Section 2.1.8.2.

Table 15: `SOpts.Bootstrap`			
☐	`.Replications`	Integer default: 0	Number of bootstrap replicates to carry out for generating confidence intervals.
☐	`.Alpha`	Double default: 0.05	The bootstrap confidence intervals are constructed with confidence level $1-\alpha$.

3.1.8 PCE-based Sobol' indices

For details on the usage, please refer to Section 2.1.9.

Table 16: `SOpts.Sobol (PCE-based indices)`			
☐	`.PCEBased`	Logical default: true	If set to false, UQLAB will compute the Monte Carlo estimates of the Sobol' indices, not the analytical ones.
☐	`.Order`	Integer default: 1	Maximum order of the interactions for which the Sobol' indices are calculated.

3.1.9 LRA-based Sobol' indices

For details on the usage, please refer to Section 2.1.10.

Table 17: `SOpts.Sobol (LRA-based indices)`			
☐	`.LRABased`	Logical default: true	If set to false, UQLAB will compute the Monte Carlo estimates of the Sobol' indices, not the analytical ones.
☐	`.Order`	Integer default: 1	Maximum order of the interactions for which the Sobol' indices are calculated.

3.1.10 ANCOVA indices

For details on the usage, please refer to Section 2.2.2.

Table 18: `SOpts.ANCOVA`

⊞	`.SampleSize`	Double default: 200	Size of the experimental design for the PCE. Drives the cost of the analysis.
☐	`.Sampling`	String default: `'LHS'`	Sampling strategy for generating the samples (see Table 14).
⊞	`.Samples`	See Table 19	Provide the experimental design for the PCE.
☐	`.MCSamples`	Integer default: 10000	Specify how many samples are used to get the MC estimations of the indices.
☐	`.CustomPCE`	UQLAB MODEL object of type PCE metamodel	Provide a PCE MODEL object to be directly used by the analysis. Any MODEL object in `SOpts.Model` will be ignored.

☐	`.PCE`	See UQLAB User Manual – Polynomial Chaos Expansions	Define the options for the set-up of the PCE. By default
			`PCEOpts.Degree = 1:10`
			`PCEOpts.Method = 'LARS'`

Table 19: `SOpts.ANCOVA.Samples`

●	`.X`	$N \times M$ Double	Input sample. Its length will be set as `.SampleSize`.
☐	`.Y`	$N \times N_{out}$ Double	Model evaluations of `.X`.

3.1.11 Kucherenko indices

For details on the usage, please refer to Section 2.2.3.

Table 20: `SOpts.Kucherenko`

⊕	`.SampleSize`	Double default: 10^4	Size of the samples to be generated and evaluated. If the `'SampleBased'` estimator is chosen, the default value is 10^5.
☐	`.Sampling`	String default: `'LHS'`	Sampling strategy for generating the samples (see Table 14).
☐	`.Estimator`	String default: `'Modified'`	Estimator for the indices.

		`'Standard'`	The first-order index is estimated with Eq. (1.76) and the total index with Eq. (1.77). The total cost is $N(2M+1)$.
		`'Modified'`	The first-order index is estimated with Eq. (1.78) and the total index with Eq. (1.77). The total cost is $N(2M+2)$.
		`'SampleBased'`	The first-order index is estimated with Eq. (1.80) and the total index with Eq. (1.79). If no output sample is provided, the total cost is N.
☐	`.Samples`	See Table 21	Manually provided input and output samples. Only valid for the `'SampleBased'` estimator.

Note: Both the `'Standard'` and the `'Modified'` estimators use Eq. (1.77) for the estimation of the total index.

Table 21: `SOpts.Kucherenko.Samples`

●	`.X`	$N \times M$ Double	Manually provided input sample.
☐	`.Y`	$N \times N_{out}$ Double	Manually provided model response.

3.2 Accessing the results

Syntax

```
myAnalysis = uq_createAnalysis(SOpts);
```

Output

Each of the sensitivity analyses presented in Chapter 2 produces different results, albeit with an overall similar structure. The details for each one of them are given in the following tables:

- Input/output correlation - Table 22
- Standard regression coefficients - Table 23
- Perturbation - Table 24
- Cotter - Table 25
- Morris - Table 26
- Borgonovo indices - Table 27
- Sobol' indices - Table 28
- PCE-based Sobol' indices - Table 32
- LRA-based Sobol' indices - Table 33
- ANCOVA indices - Table 34
- Kucherenko indices - Table 35

3.2.1 Input/Output correlation

See also Section 2.1.2.1.

Table 22: `myAnalysis.Results`		
`.CorrIndices`	$M \times N_{out}$ Double	Correlation-based sensitivity indices, according to Eq. (1.2).
`.RankCorrIndices`	$M \times N_{out}$ Double	Rank-correlation-based sensitivity indices, according to Eq. (1.4).
`.Cost`	Integer	Total model evaluations carried out by the method.
`.ExpDesign`	See Table 31	Model evaluations used during the calculation.
`.VariableNames`	$1 \times M$ Cell	Cell array with the names of the input variables.

3.2.2 Standard Regression Coefficients

See also Section 2.1.3.1.

Table 23: `myAnalysis.Results`		
`.SRCIndices`	$M \times N_{out}$ Double	SRC sensitivity indices, according to Eq. (1.8).
`.SRRCIndices`	$M \times N_{out}$ Double	Rank-correlation-based sensitivity indices, according to Eq. (1.11).
`.Cost`	Integer	Total model evaluations carried out by the method.
`.ExpDesign`	See Table 31	Model evaluations used during the calculation.
`.VariableNames`	$1 \times M$ Cell	Cell array with the names of the input variables.

3.2.3 Perturbation Method

See also Section 2.1.4.1.

Table 24: `myAnalysis.Results`		
`.Mu`	Double	First order estimate of the model mean.
`.Var`	Double	First order estimate of the model variance.

`.Sensitivity`	$M \times N_{out}$ Double	Sensitivity estimates of the perturbation method, from equation (1.15).
`.Cost`	Integer	Total model evaluations carried out by the method.
`.ExpDesign`	See Table 31	If `SOpts.SaveEvaluations` was set to true, the model evaluations are saved in this struct.
`.VariableNames`	$1 \times M$ Cell	Cell array with the names of the input variables.

3.2.4 Cotter Method

See also Section 2.1.5.1.

Table 25: `myAnalysis.Results`

`.CotterIndices`	$M \times N_{out}$ Double	Vector containing the value of the Cotter sensitivity measure for each of the factors from equation (1.16).
`.EvenOrder`	$M \times N_{out}$ Double	Vector containing the value of the even order effect for each of the factors.
`.OddOrder`	$M \times N_{out}$ Double	Vector containing the value of the odd order effect for each of the factors.
`.Cost`	Integer	Total model evaluations carried out by the method.
`.ExpDesign`	See Table 31	If `SOpts.SaveEvaluations` was set to true, the model evaluations are saved in this struct.
`.VariableNames`	$1 \times M$ Cell	Cell array with the names of the input variables.

3.2.5 Morris Method

See also Section 2.1.6.1.

Table 26: `myAnalysis.Results`		
`.Mu`	$M \times N_{out}$ Double	Original Morris sensitivity measure, μ, mean of the elementary effects.
`.MuStar`	$M \times N_{out}$ Double	Improved Morris sensitivity measure, μ^* (Eq. (1.25)).
`.Std`	$M \times N_{out}$ Double	Standard deviation of the elementary effects.
`.Cost`	Integer	Total model evaluations performed.
`.ExpDesign`	See Table 31	Model evaluations used for the estimation (if `SOpts.SaveEvaluations=1`).
`.VariableNames`	$1 \times M$ Cell	Cell array with the names of the input variables.

3.2.6 Borgonovo indices

See also Section 2.1.7.

Table 27: `myAnalysis.Results`		
`.Delta`	$M \times N_{out}$ Double	Borgonovo index estimations for each input variable and output component.
`.JointPDF`	$M \times N_{out}$ Cell	2D histogram approximations of the joint distributions of each input variable and output component.
`.Cost`	Integer	Total model evaluations carried out by the method.
`.ExpDesign`	Similar to Table 31	The used experimental design. Does not contain a `.Shuffled` field!
`.VariableNames`	$1 \times M$ Cell	Cell array with the names of the input variables.

3.2.7 MC-based Sobol' indices

See also Section 2.1.8.1.

Table 28: `myAnalysis.Results`		
`.Total`	$M \times N_{out}$ Double	Array of total Sobol' indices. Each column corresponds to an output variable.
`.FirstOrder`	$M \times N_{out}$ Double	Array of first order Sobol' indices. Each column corresponds to an output variable.
`.AllOrders`	Cell	Each element `AllOrders{i}` is a $(M_i \times N_{out})$ array of i-th order Sobol' indices for each of the J outputs.
`.VarIdx`	Cell	Each element `VarIdx{i}` is a $(M_i \times i)$ array with interactions considered in the Sobol' indices in `AllOrders{i}`. For Example, if `VarIdx{2} = [1 2]`, then `AllOrders{2}` contains the second order Sobol' index $S_{1,2}$ (see Eq. (1.40)).
`.TotalVariance`	$1 \times N_{out}$ Double	Total variance of each output of the model.
`.Cost`	Integer	Total model evaluations carried out by the method.
`.Bootstrap`	See Table 29	Results of the bootstrap method.
`.ExpDesign`	See Table 31	If `SOpts.SaveEvaluations` is `true`, the model evaluations are saved in this Struct.
`.VariableNames`	$1 \times M$ Cell	Cell array with the names of the input variables.
`.CoefficientBased`	Logical	If true, the indices are calculated analytically for either a PCE or a LRA metamodel, otherwise they are Monte Carlo estimates.

3.2.7.1 Bootstrap

See also Section 2.1.8.1.

Table 29: `myAnalysis.Results.Bootstrap`		
`.Total`	See Table 30	Struct containing the bootstrap information for the total Sobol' indices.
`.FirstOrder`	See Table 30	Struct containing the bootstrap information for the first order Sobol' indices.
`.AllOrders`	See Table 30	Cell array containing the bootstrap information for all the calculated Sobol' indices, grouped by order. Each element `AllOrders{i}` refers the i-th order indices.

Table 30: `myAnalysis.Results.Bootstrap.Total/FirstOrder/AllOrders{i}`		
`.CI`	$N_{ind} \times N_{out} \times 2$ Double	Upper and lower bounds of the confidence interval for each of the N_{ind} indices and N_{out} output variables.
`.ConfLevel`	$N_{ind} \times N_{out}$ Double	Confidence level for each of the N_{ind} indices and N_{out} output variables.
`.Mean`	$N_{ind} \times N_{out}$ Double	Means of the bootstrap estimates for each of the N_{ind} indices and N_{out} output variables.

3.2.7.2 Model Evaluations

If the `SOpts.SaveEvaluations` option is set to `true`, the model evaluations will be saved in the `myAnalysis.Results.ExpDesign` structure detailed in Table 31.

Table 31: `myAnalysis.Results.ExpDesign`		
`.X`	$N \times M$ Double	Coordinates of the experimental design.
`.Y`	$N \times N_{out}$ Double	Model responses.
`.Shuffled(ii)`	Struct	Experimental design for higher order indices. (Monte Carlo estimators only.)

3.2.8 PCE-based Sobol' indices

See also Section 2.1.9.1.

Note: this table is essentially identical to Table 28, with the exclusion of the model-evaluation and LRA related fields (`Cost`, `ExpDesign`, `Bootstrap` and `CoefficientBased`).

Table 32: `myAnalysis.Results`

`.Total`	$M \times N_{out}$ Double	Array of total Sobol' indices. Each column corresponds to an output variable.
`.FirstOrder`	$M \times N_{out}$ Double	Array of first order Sobol' indices. Each column corresponds to an output variable.
`.AllOrders`	Cell	Each element `AllOrders{i}` is a $(M_i \times N_{out})$ array of i-th order Sobol' indices for each of the J outputs.
`.VarIdx`	Cell	Each element `VarIdx{i}` is a $(M_i \times i)$ array with interactions considered in the Sobol' indices in `AllOrders{i}`. For Example, if `VarIdx{2} = [1 2]`, then `AllOrders{2}` contains the second order Sobol' index $S_{1,2}$ (see Eq. (1.40)).
`.TotalVariance`	$1 \times N_{out}$ Double	Total variance of each output of the model.
`.VariableNames`	$1 \times M$ Cell	Cell array with the names of the input variables.
`.PCEBased`	Logical	If true, the indices are analytical from the PCE coefficients, otherwise they are Monte Carlo estimates.

3.2.9 LRA-based Sobol' indices

Note: this table is essentially identical to Table 28, with the exclusion of the model-evaluation and PCE related fields (`Cost`, `ExpDesign`, `Bootstrap` and `PCEBased`).

Table 33: `myAnalysis.Results`

`.Total`	$M \times N_{out}$ Double	Array of total Sobol' indices. Each column corresponds to an output variable.
`.FirstOrder`	$M \times N_{out}$ Double	Array of first order Sobol' indices. Each column corresponds to an output variable.
`.AllOrders`	Cell	Each element `AllOrders{i}` is a $(M_i \times N_{out})$ array of i-th order Sobol' indices for each of the J outputs.
`.VarIdx`	Cell	Each element `VarIdx{i}` is a $(M_i \times i)$ array with interactions considered in the Sobol' indices in `AllOrders{i}`. For Example, if `VarIdx{2} = [1 2]`, then `AllOrders{2}` contains the second order Sobol' index $S_{1,2}$ (see Eq. (1.40)).

`.TotalVariance`	$1 \times N_{out}$ Double	Total variance of each output of the model.
`.VariableNames`	$1 \times M$ Cell	Cell array with the names of the input variables.
`.CoefficientBased`	Logical	If true, the indices are analytical from the LRA coefficients, otherwise they are Monte Carlo estimates.

3.2.10 ANCOVA indices

See also Section 2.2.2.1

Table 34: `myAnalysis.Results`		
`.PCE`	Struct	The set up PCE model for the analysis.
`.Cost`	Double	Total model evaluations carried out by the method.
`.TotalVariance`	$1 \times N_{out}$ Double	Total variance of each output of the model.
`.Uncorrelated`	$M \times N_{Out}$ Double	Array of the uncorrelated ANCOVA indices. Each column corresponds to an output variable.
`.Interactive`	$M \times N_{Out}$ Double	Array of the interactive ANCOVA indices. Each column corresponds to an output variable.
`.Correlated`	$M \times N_{Out}$ Double	Array of the correlated ANCOVA indices. Each column corresponds to an output variable.
`.FirstOrder`	$M \times N_{Out}$ Double	Array of the first order ANCOVA indices which are equal to the sum of the uncorrelated, interactive and correlated indices. Each column corresponds to an output variable.
`.VariableNames`	$1 \times M$ Cell	Cell array with the names of the input variables.

3.2.11 Kucherenko indices

See also Section 2.2.3.1

Table 35: `myAnalysis.Results`		
`.Cost`	Double	Total model evaluations carried out by the method.
`.TotalVariance`	$1 \times N_{out}$ Double	Total variance of each output of the model.
`.FirstOrder`	$M \times N_{out}$ Double	Array of first order Kucherenko indices. Each column corresponds to an output variable.
`.Total`	$M \times N_{out}$ Double	Array of total Kucherenko indices. Each column corresponds to an output variable.
`.VariableNames`	$1 \times M$ Cell	Cell array with the names of the input variables.

3.3 Printing/Visualizing of the results

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

3.3.1 Printing the results: uq_print

Syntax

```
uq_print(myAnalysis);
uq_print(myAnalysis,outidx);
```

Description

`uq_print(myAnalysis)` print a report on the results of the sensitivity analysis in object `myAnalysis`. If the model has multiple outputs, only the results for the first output variable are printed.

`uq_print(myAnalysis, outidx)` print a report on the results of the sensitivity analysis in object `myAnalysis` for the output variables specified in the array `outidx`.

Examples:

`uq_print(myAnalysis, [1 3])` will print the sensitivity analysis results for output variables 1 and 3.

3.3.2 Graphically display the results: uq_display

Syntax

```
uq_display(myAnalysis);
uq_display(myAnalysis, outidx);
```

Description

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

`uq_display(myAnalysis, outidx)` create a visualization of the results of the sensitivity analysis in object `myAnalysis` for the output variables specified in the array `outidx`.

Examples:

`uq_display(myAnalysis, [1 3])` will display the sensitivity analysis results for output variables 1 and 3.
See also the examples in Part 2.

References

Borgonovo, E. (2007). A new uncertainty importance measure. *Reliability Engineering & System Safety*, 92:771–784. 9, 10

Campolongo, F., Cariboni, J., and Saltelli, A. (2007). An effective screening design for sensitivity analysis of large models. *Environmental modelling & software*, 22(10):1509–1518. 8

Caniou, Y. (2012). *Global sensitivity analysis for nested and multiscale models*. PhD thesis, Université Blaise Pascal, Clermont-Ferrand. 10, 11, 18, 19

Cotter, S. (1979). A screening design for factorial experiments with interactions. *Biometrika*, 66:317–320. 5

Efron, B. (1979). Bootstrap methods: another look at the Jackknife. *Annals of Statistics*, 7(1):1–26. 18

Iooss, B. and Lemaître, P. (2015). A review on global sensitivity analysis methods. In Meloni, C. and Dellino, G., editors, *Uncertainty management in Simulation-Optimization of Complex Systems: Algorithms and Applications*. Springer. 1

Janon, A., Klein, T., Lagnoux, A., Nodet, M., and Prieur, C. (2014). Asymptotic normality and efficiency of two Sobol' index estimators. *ESAIM: Probability and Statistics*, 18:342–364. 15, 65

Konakli, K. and Sudret, B. (2016). Global sensitivity analysis using low-rank tensor approximations. *Reliability Engineering & System Safety*, 156:64 –83. 17, 18

Kucherenko, S., Tarantola, S., and Annoni, P. (2012). Estimation of global sensitivity indices for models with dependent variables. *Computer Physics Communications*, 183:937–946. 20, 21, 22

Li, G., Rabitz, H., Yelvington, P., Oluwole, O., Bacon, F., Kolb, C., and Schoendorf, J. (2010). Global sensitivity analysis for systems with independent and/or correlated inputs. *The Journal of Physical Chemistry A*, 114:6022–6032. 19

Liu, Q. and Homma, T. (2009). A new computational method of a moment-independent uncertainty importance measure. *Reliability Engineering & System Safety*, 94(7):1205–1211. Special Issue on Sensitivity Analysis. 11

Morris, M. D. (1991). Factorial sampling plans for preliminary computational experiments. *Technometrics*, 33(2):161–174. 6, 8, 38

Plischke, E., Borgonovo, E., and Smith, L. S. (2013). Global sensitivity measures from given data. *European Journal of Operational Research*, 226(3):536 – 550. 10

Saltelli, A. and Homma, T. (1996). Importance measures in global sensitivity analysis of model output. *Reliability Engineering & System Safety*, 52:1–17. 15, 65

Sobol', I. M. (1993). Sensitivity estimates for nonlinear mathematical models. 1(4):407–414. 12, 13, 65

Sudret, B. (2008). Global sensitivity analysis using polynomial chaos expansions. *Reliability Engineering & System Safety*, 93(7):964–979. 16

Sudret, B. and Caniou, Y. (2013). Analysis of covariance (ANCOVA) using polynomial chaos expansions. In Deodatis, G., editor, *Proc. 11th Int. Conf. Struct. Safety and Reliability (ICOSSAR'2013), New York, USA*. 18

Xu, C. and Gertner, G. Z. (2008). Uncertainty and sensitivity analysis for models with correlated parameters. *Reliability Engineering & System Safety*, 93(10):1563–1573. 18, 19