ETH zürich

UQLab User Manual
Random Fields

M. Moustapha, N. Fajraoui, S. Marelli, and 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

M. Moustapha, N. Fajraoui, S. Marelli, and B. Sudret, UQLab user manual – Random fields, Report UQLab-V2.1-119, Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland, 2024

BibTeX entry

```
@TechReport{UQdoc_21_119,
author = {Moustapha, M. and Fajraoui, N. and Marelli, S. and Sudret, B.},
title = {{UQLab user manual -- Random fields}},
institution = {Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich,
Switzerland},
year = {2024},
note = {Report UQLab-V2.1-119}
}
```

Document Data Sheet

Document Ref.	UQLAB-V2.1-119
Title:	UQLAB user manual – Random fields
Authors:	M. Moustapha, N. Fajraoui, S. Marelli, and 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 • Modified the apprach to discretize conditional random fields.
V2.0	31/01/2022	Initial release

Abstract

In many engineering problems, ranging from earthquake ground motions to acoustic propagation and environmental engineering problems, stochastic analysis involves the modeling of input parameters that vary randomly in space and/or time (*e.g.*, load distributions or material parameters). This type of uncertainty can be modeled by means of random fields.

The UQLAB random field module offers an easy way to define a random field, discretize it for use in engineering problems featuring time/space variability and, eventually, to sample trajectories from it.

The manual is divided into the following three parts:

- A short review of the main methods that are used to define, discretize and generate a random field;
- An in-depth, example-driven user guide, with the explanation of most of the available options and methods;
- A comprehensive reference list detailing all the available functionalities in UQLAB.

Keywords: Random field, Karhunen-Loève, EOLE, Probabilistic input model, Sampling.

Contents

Chapter 1

Theory

1.1 Introduction

In many engineering fields, statistical analysis can require the modeling of input parameters that vary randomly in space and/or time (*e.g.*, load distributions or material properties). Random fields (RF) theory is a mathematically rigorous framework that allows for the modelling of this type of uncertainty. Random fields represent random quantities that are defined over a continuous domain, and thus consist of an infinite number of random variables. To make them computationally treatable, it is therefore necessary to discretize them.

Various discretization methods have been developed in the literature for representing random fields in a way that is suitable for a subsequent sampling (Sudret and Der Kiureghian, 2000). Two well-known methods are currently implemented in UQLAB, namely *Karhunen-Loève expansion* (KLE) (Ghanem and Spanos, 1991) and *expansion optimal linear estimation* (EOLE) (Li and Der Kiureghian, 1993). They belong to a class of so-called series expansion methods, where a particular realization of the random field is represented by a finite sum of deterministic functions. In KLE and EOLE, these functions are obtained using the spectral decomposition of the covariance of the random field (Karhunen, 1947; Spanos and Ghanem, 1989). These methods are particularly efficient since they capture the variability of the uncertain quantity using only a small number of random variables, hence reducing the dimensionality of the problem while preserving, to a given extent, the key statistics of the considered random field (Marzouk and Najm, 2009).

In some applications, additional information, such as the exact values of the random field at some locations, is available. Such cases are dealt with in the framework of conditional random fields. Two such techniques are implemented in UQLAB and briefly presented in this report.

1.2 Basics of random fields

The theory of random fields is treated in Lin (1967); Vanmarcke and Grigoriu (1983); Vanmarcke et al. (1986). In this section, we describe the basic elements necessary to understand and use the random field module of UQLAB.

Let us consider the Hilbert space $\mathcal{L}^2(\Theta, \mathcal{F}, \mathcal{P})$ of all random variables with finite variance. Let $\mathcal{D} \subset \mathbb{R}^d$ be a domain that describes the geometry of the system. A random field $H(\boldsymbol{x}, \theta) : \mathcal{D} \times \Theta \longrightarrow \mathbb{R}$ is a set of random variables indexed by a continuous parameter $\boldsymbol{x} \in \mathcal{D}$ representing a coordinate in the topological space $\mathcal{D}$, with $\theta \in \Theta$ being an outcome in the sample space Θ.

The dimension d of a random field is the dimension of its topological space $\mathcal{D}$. One usually distinguishes between a one- and a multi-dimensional random field, depending on the dimension of $\boldsymbol{x}$, that is whether $d = 1$ or $d > 1$. Furthermore, a random field is said to be univariate or real-valued if the random quantity attached to each point $\boldsymbol{x}$ is a random variable. The random field is multivariate if the random quantity is a random vector. It is worth to note that the domain is not reserved only to a space domain, it can be any type of domain, such as a time domain, in which the random field is usually called stochastic process.

Two main categories of random fields can be defined based on their probability distribution: Gaussian and non-Gaussian. In this manual, we will only consider Gaussian random fields, as well as a special class of non-Gaussian ones, the so-called translation non-Gaussian random fields.

The Gaussian assumption is usually employed for its simplicity. A random field is said Gaussian if the distribution of $H(\boldsymbol{x}_1, ..., \boldsymbol{x}_n)$ is jointly Gaussian, for any $(\boldsymbol{x}_1, ..., \boldsymbol{x}_n)$ and any $n \in \mathbb{N}$. It is fully characterized by its mean function $\mathbb{E}[H(\boldsymbol{x})] : \mathcal{D} \longrightarrow \mathbb{R}$ and autocovariance function $C(\boldsymbol{x}_1, \boldsymbol{x}_2) : \mathcal{D} \times \mathcal{D} \longrightarrow \mathbb{R}$, expressed respectively as:

$$\mu(\boldsymbol{x}) \equiv \mathbb{E}[H(\boldsymbol{x}, \theta)] = \int_{\Theta} H(\boldsymbol{x}, \theta)\, dP(\theta), \tag{1.1}$$

$$C(\boldsymbol{x}_1, \boldsymbol{x}_2) = \mathrm{Cov}\left[H\left(\boldsymbol{x}_1, \theta\right), H\left(\boldsymbol{x}_2, \theta\right)\right]. \tag{1.2}$$

The autocorrelation function $\rho(\boldsymbol{x}_1, \boldsymbol{x}_2) : \mathcal{D} \times \mathcal{D} \longrightarrow [-1, 1]$ is defined as:

$$\rho(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{C(\boldsymbol{x}_1, \boldsymbol{x}_2)}{\sigma(\boldsymbol{x}_1)\sigma(\boldsymbol{x}_2)} \tag{1.3}$$

where $\sigma(\boldsymbol{x}) : \mathcal{D} \longrightarrow \mathbb{R}$ is the standard deviation function of the random field.

When the statistical properties of the random field are invariant by translation, the random field is said to be stationary. This implies that the mean $\mu(\boldsymbol{x})$ and the variance $\sigma^2(\boldsymbol{x})$ are constant and, in the case of weak-sense stationarity, that the covariance depends only on $\boldsymbol{x}_1 - \boldsymbol{x}_2$. When this dependence is further restricted to $\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2$, the random field is said to be isotropic.

Note: Only stationary univariate random fields are currently implemented in UQLAB.

1.3 Discretization of a random field

The discretization of a random field $H(\boldsymbol{x}, \theta)$ is a process by which the random field is approximated using a finite set of random variables, making it suitable for computational applications. The main goal is to define the best approximation with respect to some error estimator, that is the one requiring the minimal number of random variables (Sudret and Der Kiureghian, 2000).

Various techniques have been developed for the discretization of random fields in the literature, as reviewed in Li and Der Kiureghian (1993); Ditlevsen (1996); Matthies et al. (1997). They can be divided into three groups (Sudret and Der Kiureghian, 2000):

- *point discretization*: the random field is described by a set of random variables that represent the values of the random field at some given points, such as the mid-point or the shape function methods;
- *average discretization*: the random variables are averaged over local domains, such as the weighted integral or the spatial average methods;
- *series expansion methods*: where the random field is exactly represented as a series involving a complete set of deterministic functions with corresponding random variables, such as Karhunen-Loève.

Only series expansion methods are considered in UQLAB. This manual focuses on the two methods currently implemented, namely the Karhunen-Loève expansion and the expansion optimal linear estimation.

1.3.1 Karhunen Loève expansion (KLE)

Karhunen-Loève, also known as proper orthogonal decomposition, is one of the most popular random field discretization techniques. It is optimal among series expansion methods in terms of the global mean square error with respect to the number of random variables in the discretization (Ghanem and Spanos, 1991). In essence, KLE is based on the spectral expansion of the process covariance function, using orthogonal deterministic basis functions and uncorrelated random coefficients.

Let $H(\boldsymbol{x}, \theta)$ be a random field, with finite second moments, mean $\mu(\boldsymbol{x})$ and a bounded, symmetric and positive definite covariance function $C(\boldsymbol{x}_1, \boldsymbol{x}_2)$. The KLE of the random field

$H(\boldsymbol{x},\theta)$ writes:

$$H(\boldsymbol{x},\theta) = \mu(\boldsymbol{x}) + \sum_{i=1}^{\infty} \sqrt{\lambda_i}\xi_i(\theta)\varphi_i(\boldsymbol{x}) \tag{1.4}$$

where λ_i and $\varphi_i(\boldsymbol{x})$ are the eigenvalues and eigenfunctions of the covariance function, respectively, and $\{\xi_i(\theta);\, i = 1, ...\}$ is a sequence of uncorrelated, zero-mean and unit-variance random variables.

Following Mercer's theorem, $C(\boldsymbol{x}_1, \boldsymbol{x}_2)$ can be decomposed as follows:

$$C(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sum_{i=1}^{\infty} \lambda_i \varphi_i(\boldsymbol{x}_1)\varphi_i(\boldsymbol{x}_2) \tag{1.5}$$

where the eigenvalues λ_i and eigenfunctions $\varphi_i(\boldsymbol{x})$ are obtained by solving the homogeneous Fredholm integral equation of the second kind given by:

$$\int_{\mathcal{D}} C(\boldsymbol{x}_1, \boldsymbol{x}_2)\varphi_i(\boldsymbol{x}_2)\, d\boldsymbol{x}_2 = \lambda_i \varphi_i(\boldsymbol{x}_1) \tag{1.6}$$

The eigenvalues are generally indexed in decreasing order (*i.e.*, $\lambda_1 \geq \lambda_2 \geq \cdots$). The set of eigenfunctions $\varphi_i(\boldsymbol{x})$ are orthogonal and form a complete basis for the random field, *i.e.*,

$$\int_{\mathcal{D}} \varphi_i(\boldsymbol{x})\varphi_j(\boldsymbol{x})\, d\boldsymbol{x} = \delta_{ij}, \tag{1.7}$$

where δ_{ij} is the Kronecker symbol defined such that $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise.

Due to the orthonormality of the eigenfunctions, the random variable appearing in the series in Eq. (1.4) can be expressed as the following linear transformation:

$$\xi_i(\theta) = \frac{1}{\sqrt{\lambda_i}} \int_{\mathcal{D}} \left[H(\boldsymbol{x},\theta) - \mu(\boldsymbol{x})\right] \varphi_i(\boldsymbol{x})\, d\boldsymbol{x} \tag{1.8}$$

For Gaussian random fields, the random variables $\{\xi_i(\theta), i = 1, \ldots\}$ are independent standard Gaussian random variables. In any other case, the joint distribution of the set $\{\xi_i(\theta), i = 1, \ldots\}$ is unknown and almost impossible to obtain analytically. Hence, KLE is mainly adopted for the discretization of Gaussian fields. The simulation of non-Gaussian random fields by means of KLE is extensively discussed in the literature (see, e.g., Phoon et al. (2005)).

For practical implementation, the Karhunen-Loève expansion in Eq. (1.4) is truncated after a finite number of terms, corresponding to the M largest eigenvalues, *i.e.*,

$$H(\boldsymbol{x},\theta) \approx \widehat{H}(\boldsymbol{x},\theta) = \mu(\boldsymbol{x}) + \sum_{i=1}^{M} \sqrt{\lambda_i}\xi_i(\theta)\varphi_i(\boldsymbol{x}). \tag{1.9}$$

For a fixed M, KLE is optimal with respect to the total mean square error. In other words, among all possible bases for $\mathcal{L}^2(\mathcal{D})$, and for any a finite number M of terms, the functions

$\{\varphi_i(\boldsymbol{x}), i = 1, \ldots, M\}$ satisfying Eq. (1.6) minimimize the mean-square error resulting from a finite representation of the process. That is, they minimize

$$\int_{\theta \times \mathcal{D}} \left(H(\boldsymbol{x}, \theta) - \mu(\boldsymbol{x}) - \sum_{i=1}^{M} \sqrt{\lambda_i} \xi_i(\theta) \varphi_i(\boldsymbol{x}) \right)^2 dP(\theta)\, d\boldsymbol{x} \tag{1.10}$$

The number of terms M to keep in the expansion in Eq (1.9) depends on the correlation length of the covariance function: the shorter the correlation length, the higher M. It is easy to verify that KLE underestimates the true variance of the field after truncation. Generally, the quality of the approximation is deemed sufficient when the so-called *"energy" ratio* (also known as *explained variance*) is larger than a given threshold δ_e:

$$\sum_{i=1}^{M} \lambda_i / \sum_{i=1}^{\infty} \lambda_i \geq \delta_e. \tag{1.11}$$

The error variance obtained when truncating the expansion after M terms can be shown to read as follows:

$$\mathrm{Var}\left[H(\boldsymbol{x}) - \widehat{H}(\boldsymbol{x}) \right] = \sigma^2(\boldsymbol{x}) - \sum_{i=1}^{M} \lambda_i^2 \varphi_i^2(\boldsymbol{x}). \tag{1.12}$$

1.3.1.1 Numerical solution schemes

At the core of Karhunen-Loève expansion lies the Fredholm equation (Eq. (1.6)). In some special cases, *e.g.*, when using the exponential kernel and with $d \leq 2$, the eigenvalues and the corresponding eigenfunctions can be computed analytically (Ghanem and Spanos, 2003). However, in most cases, Eq. (1.6) is solved by means of numerical methods. In this regard, several methodologies have been proposed in the literature. A comprehensive overview of the numerical methods to solve the Fredholm integral equation is given in Atkinson (1997). In UQLAB, the *Nyström* method is implemented and briefly discussed in the sequel.

Nyström method

The Nyström method solves the Fredholm integral equation defined in Eq. (1.6) using a quadrature approach, generally Gaussian quadrature (Press et al., 2007):

$$\sum_{j=1}^{n} w_j C(\boldsymbol{x}, \boldsymbol{x}_j) \varphi(\boldsymbol{x}_j) = \lambda \varphi(\boldsymbol{x}_j); \tag{1.13}$$

where $w_j \in \mathbb{R}$ and $\boldsymbol{x}_j \in \mathcal{D}$, $j \in \{1, \ldots n\}$ are respectively the integration weights and points of the chosen quadrature type. Solving Eq. (1.13) on the integration points yields:

$$\sum_{j=1}^{n} w_j C(\boldsymbol{x}_i, \boldsymbol{x}_j) \varphi_i(\boldsymbol{x}_j) = \lambda_i \varphi_i(\boldsymbol{x}_j), \quad i \in \{1, \ldots, n\}, \tag{1.14}$$

which can be written in matrix form as:

$$\boldsymbol{C}\boldsymbol{W}\boldsymbol{\Phi}_i = \lambda_i \boldsymbol{\Phi}_i, \tag{1.15}$$

where $\boldsymbol{C}$ is the covariance matrix with elements $c_{ij} = C(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $\boldsymbol{W} = \text{diag}\,(w_j)$ is a diagonal matrix of weights and $\boldsymbol{\Phi}_i$ is an n-dimensional vector with the j-th entry defined by $\boldsymbol{\Phi}_{i,j} = \varphi_i\,(\boldsymbol{x}_j)$. While this is a matrix eigenvalue problem, $\boldsymbol{C}\boldsymbol{W}$ is not necessarily symmetric. To solve the eigenvalue problem in a computationally efficient manner, a symmetric matrix is often required. Assuming that the weights are positive (and they are for Gaussian quadrature), the trick is to introduce the square root of the matrix $\boldsymbol{W}$, denoted by $\boldsymbol{W}^{\frac{1}{2}} = \text{diag}\left(\sqrt{w_j}\right)$, and then to restore symmetry by multiplying both sides of Eq. (1.15) with $\boldsymbol{W}^{\frac{1}{2}}$, *i.e.*,

$$\left(\boldsymbol{W}^{\frac{1}{2}}\boldsymbol{C}\boldsymbol{W}^{\frac{1}{2}}\right)\boldsymbol{\Phi}_i^{\star} = \lambda_i \boldsymbol{\Phi}_i^{\star}, \tag{1.16}$$

where $\boldsymbol{\Phi}_i^{\star} = \boldsymbol{W}^{\frac{1}{2}}\boldsymbol{\Phi}_i$. The matrix $\widetilde{\boldsymbol{C}} = \boldsymbol{W}^{\frac{1}{2}}\boldsymbol{C}\boldsymbol{W}^{\frac{1}{2}}$ is symmetric and Eq. (1.16) can henceforth be easily solved by standard eigenvalue routines.

Another advantage of the Nyström method is that the eigenvectors can be extrapolated to any other point in $\mathcal{D}$ while maintaining a high degree of accuracy (Press et al., 2007). This is achieved using the Nyström interpolation (Betz et al., 2014) defined by:

$$\varphi_i\,(\boldsymbol{x}) = \frac{1}{\lambda_i}\sum_{j=1}^{n}\sqrt{w_j}\varphi_i^{\star}\,(\boldsymbol{x}_j)\,C\,(\boldsymbol{x}, \boldsymbol{x}_j)\,, \tag{1.17}$$

where $\varphi_i^{\star}\,(\boldsymbol{x}_j)$ is the j-th element of $\boldsymbol{\Phi}_i^{\star}$.

Discrete approach

Alternatively to solving the Integral Fredholm equation, one may formulate the Karhunen-Loève problem in a discrete form as discussed in Schenk and Schuëller (2005). In this case, the correlation matrix is directly computed on the discretization mesh. Carrying out a spectral decomposition on the resulting matrix, the resulting pairs of eigenvalues and eigenvectors can be directly used for the Karhunen-Loève approximation in Eq. (1.9). This method is equivalent to principal component analysis, where the data are generated using the specification of the random field, more specifically the correlation matrix.

1.3.2 Expansion optimal linear estimation (EOLE)

The expansion optimal linear estimation (EOLE) is a series expansion method that was introduced by Li and Der Kiureghian (1993) as an extension of the optimal linear estimation (OLE) method. In the following, we first briefly review the OLE method.

1.3.2.1 Optimal linear estimation

The OLE method was first introduced by Li and Der Kiureghian (1993) and is sometimes referred to as the Kriging method. This method requires a set of nodal points $\boldsymbol{\mathcal{X}} = \{\chi_i;\, i = 1, \ldots, n\}$ lying in the domain of the random field. The approximation field $\widehat{H}$ is then expressed as a linear function of nodal values $\boldsymbol{\mathcal{Y}} = \left\{y^i = H\left(\chi_i\right)\right\}$ as follows:

$$\widehat{H}(\boldsymbol{x}) = a(\boldsymbol{x}) + \sum_{i=1}^{n} b(\boldsymbol{x}) y^i = a(\boldsymbol{x}) + \boldsymbol{b}^T(\boldsymbol{x}) \cdot \boldsymbol{\mathcal{Y}}, \tag{1.18}$$

where n is the total number of nodal points involved in the approximation. The functions $a(\boldsymbol{x})$ and $\boldsymbol{b}(\boldsymbol{x})$ are obtained by minimizing in each point $\boldsymbol{x}$ the variance of the error $\mathrm{Var}\left[H(\boldsymbol{x}) - \widehat{H}(\boldsymbol{x})\right]$ under the condition that $\widehat{H}(\boldsymbol{x})$ is an unbiased estimator of the real random field in the mean. These conditions are expressed as:

$$\begin{aligned} &\forall \boldsymbol{x} \in \mathcal{D}, \min \mathrm{Var}\left[H(\boldsymbol{x}) - \widehat{H}(\boldsymbol{x})\right] \\ &\text{subject to: } \mathbb{E}\left[H\left(\boldsymbol{x}\right) - \widehat{H}\left(\boldsymbol{x}\right)\right] = 0, \end{aligned} \tag{1.19}$$

which require that:

$$\mathbb{E}\left[H\left(\boldsymbol{x}\right)\right] = a(\boldsymbol{x}) + \boldsymbol{b}^T(\boldsymbol{x}) \cdot \mathbb{E}\left[\boldsymbol{\mathcal{Y}}\right]. \tag{1.20}$$

After some algebraic manipulations, the variance of the error reads:

$$\mathrm{Var}\left[H(\boldsymbol{x}) - \widehat{H}(\boldsymbol{x})\right] = \sigma^2(\boldsymbol{x}) - 2\sum_{i=1}^{n} b_i(\boldsymbol{x}) \mathrm{Cov}\left[H(\boldsymbol{x}), \mathcal{Y}_i\right] + \sum_{i=1}^{n}\sum_{j=1}^{n} b_i(\boldsymbol{x}) b_j(\boldsymbol{x}) \mathrm{Cov}\left[\mathcal{Y}_i, \mathcal{Y}_j\right]. \tag{1.21}$$

The minimization problem is solved for each $b_i(\boldsymbol{x})$. Requiring that the partial derivatives of $b_i(\boldsymbol{x})$ be equal to zero for each i yields:

$$-\mathrm{Cov}\left[H(\boldsymbol{x}), \mathcal{Y}_i\right] + \sum_{i=1}^{n} b_i(\boldsymbol{x}) \mathrm{Cov}\left[\mathcal{Y}_i, \mathcal{Y}_j\right] = 0, \quad i = 1, \ldots, n \tag{1.22}$$

In a matrix form, Eq. (1.22) is equivalent to

$$-C_{H\boldsymbol{\mathcal{Y}}} + C_{\boldsymbol{\mathcal{Y}}\boldsymbol{\mathcal{Y}}} \cdot \boldsymbol{b}(\boldsymbol{x}) = 0, \tag{1.23}$$

where $C_{\boldsymbol{\mathcal{Y}}\boldsymbol{\mathcal{Y}}}$ is the covariance matrix of $\boldsymbol{\mathcal{Y}}$. The random field can then be written as:

$$\widehat{H}(\boldsymbol{x}, \theta) = \mu(\boldsymbol{x}) + C^T_{H(\boldsymbol{x}),\boldsymbol{\mathcal{Y}}} \cdot C^{-1}_{\boldsymbol{\mathcal{Y}}\boldsymbol{\mathcal{Y}}} \left(\boldsymbol{\mathcal{Y}} - \mu\left(\boldsymbol{x}\right)\right). \tag{1.24}$$

1.3.2.2 Expansion optimal linear estimation

The EOLE method is based on a spectral decomposition of the covariance matrix of the OLE method. Let $H(\boldsymbol{x}, \theta)$ be a Gaussian random field, using the spectral decomposition of the

covariance matrix $C_{\boldsymbol{\mathcal{Y}}\boldsymbol{\mathcal{Y}}}$ of $\boldsymbol{\mathcal{Y}} = \{y^i = H(\boldsymbol{x}^i)\}$, the random field can be approximated as:

$$\widehat{H}(\boldsymbol{x},\theta) = \mu(\boldsymbol{x}) + \sum_{i=1}^{M} \sqrt{\lambda_i}\xi_i(\theta)\varphi_i(\boldsymbol{x}), \tag{1.25}$$

where $\{\xi_i, i = 1, ..., M\}$ are independent standard normal variables and (λ_i, φ_i) are the respectively eigenvalues and eigenvectors of the covariance matrix $C_{\boldsymbol{\mathcal{Y}}\boldsymbol{\mathcal{Y}}}$, verifying that:

$$C_{\boldsymbol{\mathcal{Y}}\boldsymbol{\mathcal{Y}}}\varphi_i = \lambda_i\varphi_i. \tag{1.26}$$

Substituting Eq. (1.25) in Eq. (1.18) and solving the OLE problem yields:

$$\widehat{H}(\boldsymbol{x},\theta) = \mu(\boldsymbol{x}) + \sum_{i=1}^{M} \frac{\xi_i(\theta)}{\sqrt{\lambda_i}}\varphi_i^T C_{H\boldsymbol{\mathcal{Y}}}. \tag{1.27}$$

The series is also truncated after M terms and only the most important eigenmodes are considered for the approximation of the random field. The quality of the approximation is determined by the number of the retained terms.

The error variance of a random field representation based on the EOLE method is given by:

$$\mathrm{Var}\left[H(\boldsymbol{x}) - \widehat{H}(\boldsymbol{x})\right] = \sigma^2(\boldsymbol{x}) - \sum_{i=1}^{M} \frac{1}{\lambda_i}(\varphi_i^T(\boldsymbol{x})C_{H\boldsymbol{\mathcal{Y}}})^2. \tag{1.28}$$

Similarly to Karhunen-Loève, the EOLE method also underestimates the total variance of the random field. Hence, M should be chosen large enough in order to ensure a good approximation of $H(\boldsymbol{x})$.

In both KLE and EOLE, the quality of the approximation can be estimated using the energy ratio described in Eq. (1.11). However, when using a numerical solution scheme the number of computed eigenvalues is limited to the size n of the covariance mesh. We therefore can only compute the following approximate energy ratio:

$$\sum_{i=1}^{M} \lambda_i / \sum_{i=1}^{n} \lambda_i \geq \delta_e. \tag{1.29}$$

Furthermore, it is not necessary to compute all the n eigenvalues to get the denominator of Eq. (1.29). It can indeed be shown that the sum of the eigenvalues of a matrix is equal to its trace, therefore:

$$\sum_{i=1}^{n} \lambda_i = \mathrm{tr}\left(\boldsymbol{C}\right). \tag{1.30}$$

1.4 Conditional random fields

Let us assume that the measurements of $H(\boldsymbol{x})$ at some locations $\boldsymbol{\mathcal{X}} = \{\boldsymbol{x}^{(i)},\, i = 1, 2, \ldots, K\}$ are deterministically known, *i.e*, $\boldsymbol{\mathcal{Y}} = \{y^{(i)} = H(\boldsymbol{x}^{(i)}),\, i = 1, 2, \ldots, K\}$. Then, a conditional random field can be defined as a random field conditioned on the observations $\boldsymbol{\mathcal{X}}$.

Let $H^c(\boldsymbol{x})$ be the conditional random field that is defined as follows:

$$H^c(\boldsymbol{x}) = H(\boldsymbol{x} \mid \boldsymbol{\mathcal{Y}}). \tag{1.31}$$

To discretize this conditional random fields, we first derive the conditional mean $\mu^c(\boldsymbol{x})$ and covariance function $C^c_{HH}(\boldsymbol{x}_1, \boldsymbol{x}_2)$ given the data and then perform the spectral decomposition on the latter using the methods described earlier.

Let consider a random vector $\boldsymbol{Z} \in \mathbb{R}^{K+n}$ which can be defined by concatenating the observations $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^K$ and the unknown values of the conditional random field on the discretization mesh $\widehat{\boldsymbol{Y}} = \left\{H^c\left(\boldsymbol{x}^{(i)}\right), i = 1, \ldots, n\right\} \in \mathbb{R}^n$. The vector $\boldsymbol{Z}$ follows by definition a Gaussian distribution, *i.e.* $\boldsymbol{Z} \sim \mathcal{N}(\boldsymbol{\mu}_Z, \boldsymbol{\Sigma}_{ZZ})$ with parameters defined by

$$\boldsymbol{\mu}_Z = \begin{bmatrix} \boldsymbol{\mu}_{\widehat{\boldsymbol{Y}}} \\ \boldsymbol{\mu}_{\boldsymbol{\mathcal{Y}}} \end{bmatrix}, \qquad \boldsymbol{\Sigma}_{\boldsymbol{Z}\boldsymbol{Z}} = \begin{bmatrix} \boldsymbol{\Sigma}_{\widehat{\boldsymbol{Y}}\widehat{\boldsymbol{Y}}} & \boldsymbol{\Sigma}_{\widehat{\boldsymbol{Y}}\boldsymbol{\mathcal{Y}}} \\ \boldsymbol{\Sigma}^T_{\widehat{\boldsymbol{Y}}\boldsymbol{\mathcal{Y}}} & \boldsymbol{\Sigma}_{\boldsymbol{\mathcal{Y}}\boldsymbol{\mathcal{Y}}}. \end{bmatrix} \tag{1.32}$$

The distribution of $\widehat{\boldsymbol{Y}}$ conditioned on the observations $\boldsymbol{\mathcal{Y}}$ is also a multivariate Gaussian $\widehat{\boldsymbol{Y}}|\boldsymbol{\mathcal{Y}} \sim \mathcal{N}\left(\boldsymbol{\mu}_{\widehat{\boldsymbol{Y}}|\boldsymbol{\mathcal{Y}}}, \boldsymbol{\Sigma}_{\widehat{\boldsymbol{Y}}|\boldsymbol{\mathcal{Y}}}\right)$, where

$$\begin{aligned} \boldsymbol{\mu}_{\widehat{\boldsymbol{Y}}|\boldsymbol{\mathcal{Y}}} &= \boldsymbol{\mu}_{\widehat{\boldsymbol{Y}}} + \boldsymbol{\Sigma}_{\widehat{\boldsymbol{Y}}\boldsymbol{\mathcal{Y}}}\boldsymbol{\Sigma}^{-1}_{\boldsymbol{\mathcal{Y}}\boldsymbol{\mathcal{Y}}}\left(\boldsymbol{\mathcal{Y}} - \boldsymbol{\mu}_{\boldsymbol{\mathcal{Y}}}\right), \\ \boldsymbol{\Sigma}_{\widehat{\boldsymbol{Y}}|\boldsymbol{\mathcal{Y}}} &= \boldsymbol{\Sigma}_{\widehat{\boldsymbol{Y}}\widehat{\boldsymbol{Y}}} - \boldsymbol{\Sigma}^T_{\widehat{\boldsymbol{Y}}\boldsymbol{\mathcal{Y}}}\boldsymbol{\Sigma}^{-1}_{\boldsymbol{\mathcal{Y}}\boldsymbol{\mathcal{Y}}}\boldsymbol{\Sigma}^T_{\widehat{\boldsymbol{Y}}\boldsymbol{\mathcal{Y}}}. \end{aligned} \tag{1.33}$$

To sample from $H^c(\boldsymbol{x})$, UQLAB discretizes the random field whose mean and covariance is given in Eq. (1.33) using the approaches described in Sections 1.3.1 and 1.3.2.

1.5 Non-Gaussian translation random fields

Non-Gaussian random fields generally require special techniques for discrete representation. However, there exists a sub-class of non-Gaussian random fields that can be obtained by a non-linear mapping in the form (Betz et al., 2014):

$$\widetilde{H}(\boldsymbol{x}, \theta) = g\left(H(\boldsymbol{x}, \theta)\right), \tag{1.34}$$

where $H(\boldsymbol{x}, \theta)$ is a Gaussian random field and $g : \mathbb{R} \longrightarrow \mathbb{R}$ is a strictly increasing non-linear function. Such random fields are known as *translation* random fields (Grigoriu, 1984).

In UQLAB, we consider the class of non-Gaussian random fields such that g is defined as an

isoprobabilistic transform, *i.e.*,

$$g\left(\boldsymbol{x}\right) = F_{\widetilde{H}}^{-1} \circ \Phi_{\mathcal{N}}\left(\boldsymbol{x}\right), \tag{1.35}$$

where $\Phi_{\mathcal{N}}$ is the Gaussian cumulative distribution function (CDF) with parameters equal to the mean and standard deviation of the random field $H\left(\boldsymbol{x},\theta\right)$ and $F_{\widetilde{H}}$ is the CDF of the target non-Gaussian random field.

Chapter 2

Usage

In this section, the discretization of a random field as discussed in Chapter 1 is described. An example involving a one-dimensional Gaussian random field is employed to illustrate the usage of the module.

2.1 Gaussian random fields

2.1.1 Introductory example

As an introductory example, a one-dimensional Gaussian random field defined on a domain $\Omega = [-1, 1]$ is investigated. The random field is assumed to have a constant mean $\mu = 1$ and standard deviation $\sigma = 1$. The correlation function is Gaussian with correlation length 0.2. The expansion optimal linear estimation (EOLE) is chosen as discretization technique with a target energy ratio of 0.99.

2.1.2 Problem set-up

The random field module creates an INPUT object that can be used exactly as any other. The basic options common to any random field module read:

```
uqlab;
RFInput.Type = 'RandomField';
```

The ingredients selected by the user to describe the random field are stored in the structure `RFInput`. The following fields are mandatory:

- `RFInput.RFType` the type of random field (e.g. Gaussian or lognormal).
- `RFInput.Corr.Family` correlation family.
- `RFInput.Corr.Length` the correlation length.

- `RFInput.Mean` the mean of the random field.
- `RFInput.Std` the standard deviation of the random field.
- `RFInput.Mesh` the domain of definition of the random field and the mesh used when sampling trajectories.

The remaining parameters are optional and given default values in UQLAB as described in Table 1.

2.1.3 Creating a random field object

A minimal configuration example to setup a random field in UQLAB (Section 2.1.2) is given in the following block of code.

```
uqlab;
RFInput.Type = 'RandomField';
RFInput.Corr.Family = 'Gaussian';
RFInput.Corr.Length = 0.2;
RFInput.Mean=1;
RFInput.Std=1;
RFInput.Mesh= linspace(-1,1,200)';

myRF = uq_createInput(RFInput);
```

Note: The remaining parameters are set by default within the module:

- Random field type (`RFInput.Type`): Gaussian
- Energy ratio threshold (δ_e in Eq. (1.11), `RFInput.EnergyRatio`): 0.99. The corresponding expansion order is computed using Eq. (1.29).
- Discretization scheme (`RFInput.DiscScheme`): EOLE

Once the random field is created, a report with basic information about the random field results can be printed with the command:

```
uq_print(myRF);
```

which, in our example, produces the following output:

```
------------------- Random Field properties ------------------
Input object name         : Input 1
Random field type          : Gaussian
Discretization scheme     : EOLE
Autocorrelation family    : Gaussian
Expansion order           : 10
Average error Variance    : 4.82e-03
Explained variance        : 0.9948
============================================================
```

This shows the basic information about the random field. The average error variance is computed over the covariance mesh while the explained variance or energy ratio is computed using Eq. (1.29).

Note: the total number of eigenvalues used to compute the approximate energy ratio is limited by the mesh grid. The user must therefore make sure that the discretization mesh used is sufficient to accurately represent the random field. An heuristic approach is to assess visually the convergence of the energy ratio using `uq_display` (Figure 1b).

A visual representation of the INPUT object can be obtained by:

```
uq_display(myRF)
```

which produces the images in Figure 1. The plots in Figure 1a are only available for one- and two-dimensional cases. The left panel displays five realizations of the trajectories for one-dimensional problems and a single realization for two-dimenstional problems. The right panel shows the variance error. Figure 1b shows the eigenvalues resulting from the spectral decomposition of the auto-correlation matrix.

2.1.4 Accessing the random field properties

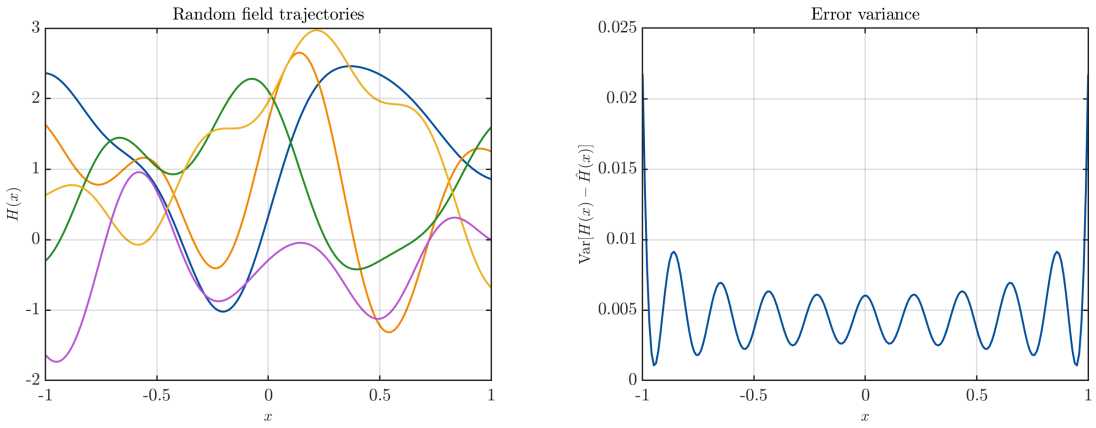
The random field INPUT object is created using the options given by the user. When options are not provided, default parameters are set such that all the necessary information required to build the random field is defined. The resulting random field properties are structured in the INPUT object, herein `myRF`, and can be accessed by the user.

Discretization

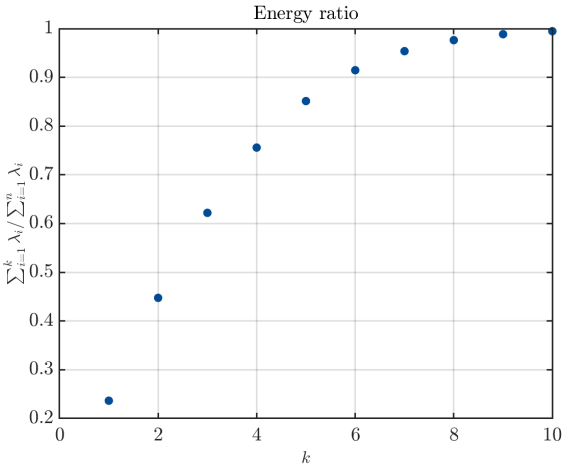
The basic results of the discretization, namely the correlation matrix, the eigenvalues, the eigenvectors, the mesh over which the covariance matrix is computed and the error variance are given in the field `myRF.RF`:

```
myRF.RF
ans =
struct with fields:
      EOLE: [1x1 struct]
       Phi: [51x10 double]
      Eigs: [10x1 double]
  VarError: [1x200 double]
   CovMesh: [51x1 double]
```

where `Phi` represents the eigenvectors, `Eigs` represents the eigenvalues (See Eqs. (1.4) and (1.27)), `VarError` contains the variance error as calculated over the discretization mesh (See Eqs. (1.12) and (1.21)) and `CovMesh` is the grid over which the covariance matrix used in the spectral decomposi-



(a) Sample trajectories and error variance over the discretization mesh.



(b) Eigenvalues of the correlation matrix up to the expansion order.

Figure 1: The output of the function `uq_display` on the INPUT object myRF. By default, the samples are drawn using Monte Carlo sampling.

tion is built (See Section 2.4.3). The structure `.EOLE`, which would reads `.KL` when using KLE, contains information directly related to the discretization scheme. In case of EOLE, it contains the following fields:

```
myRF.RF.EOLE
ans =
struct with fields:
    Corr: [51x51 double]
  Rho_vV: [200x51 double]
```

where `Corr` denotes the correlation matrix evaluated on the covariance mesh ($C_{\boldsymbol{\mathcal{Y}}\boldsymbol{\mathcal{Y}}}/\sigma^2$ in Eq. (1.24)) and `Rho_vV` is the cross-correlation matrix ($C_{H\boldsymbol{\mathcal{Y}}}/\sigma^2$ in Eq. (1.27)).

Note: The correlation matrix is not necessarily built on the mesh given by the user. An internal mesh (`CovMesh`) is created by default within the module according to the discretization scheme selected. The associated options can be specified by the user as described in Section 2.4.3.

Underlying standard Gaussian input

The random variables $\{\xi_i, i = 1, \ldots, M\}$ actually sampled to generate trajectories of the random field are realizations of an M-dimensional independent standard Gaussian distribution. With other properties of the random field being deterministic, they are the only information needed to generate the random trajectories used in many uncertainty quantification analyses. The corresponding INPUT object that can be used for other analyses within UQLAB is therefore also returned with the random field object and can be accessed in the field `myRF.UnderlyingGaussian`.

2.2 Multi-dimensional random fields

It is also possible to generate a multi-dimensional random fields. The size of the mesh or the correlation length should be adapted accordingly. No additional configuration is needed to enable this behaviour. For example, a two-dimensional random field can be specified by enabling a mesh of size $m \times 2$. In UQLAB, this can be specified as follows:

```
x = linspace(-1,1,50);
y= linspace(-1,1,50);
[X,Y] = meshgrid(x,y);
RFInput.Mesh= [X(:) Y(:)];
```

One may also define an anisotropic correlation function by choosing different correlation lengths per direction. This can be specified as follows:

```
RFInput.Corr.Length= [0.2, 0.6] ;
```

Figure 2 displays one realization of a two-dimensional random field using the settings defined above. The resulting expansion order is $M = 38$.

2.3 Drawing samples from a random field

Samples from the INPUT object `myRF` are realizations of the random field on the discretization mesh. They can be obtained as follows:

```
X = uq_getSample(300);
```

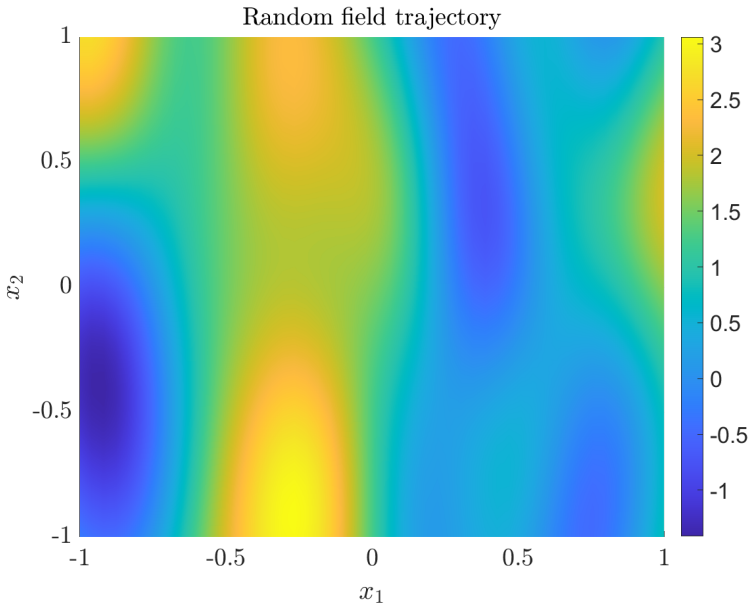


Figure 2: A realization of a two-dimensional random field with anisotropic correlation function.

where `X` is a vector of size $300 \times m$, with m being the number of points in the discretization mesh.

Notice that we do not need to define the INPUT object in `uq_getSample`, because after an INPUT object is created, if not specified otherwise, it is the one that is going to be used when calling `uq_getSample`. The MATLAB standard random number generator is used by default.

When using `uq_getSample` in order to obtain samples from a random field, various sampling methods can be used to generate the underlying Gaussian random variables $\boldsymbol{\xi} = \{\xi_i, i = 1, \ldots, M\}$. For instance, Latin Hypercube sampling (McKay et al., 1979) can be used to sample from the INPUT object `myRF` as follows:

```
X = uq_getSample(300, 'LHS');
```

Advanced options

It may be of interest to directly get the underlying Gaussian random variables $\boldsymbol{\xi}$ that are used to construct the random field trajectories. For instance, using

```
[X, xi] = uq_getSample(300);
```

would additionally return a vector `xi` of size $300 \times M$ where each row is a vector $\boldsymbol{\xi}$.

When some vectors $\boldsymbol{\xi}$ and a random field object `myRF` are available, it is possible to construct the random field trajectories using the function `uq_RF_Xi_to_X` as follows:

```
X = uq_RF_Xi_to_X(myRF,xi)
```

Note: Advanced options regarding the use of `uq_getSample` can be found in Section 2.2 of UQLAB User Manual – the INPUT module.

2.4 Set-up of the random field

2.4.1 Discretization scheme

The default discretization scheme is EOLE. The user may alternatively consider Karhunen-Loève by specifying the following:

```
RFInput.DiscScheme = 'KL';
```

There are different solution schemes used within KLE, which may be specified using the `.KL.Method` field. Generally, the Nyström method with full quadrature is considered by default, except when the exponential kernel is used to compute the covariance (in which case an analytical solution scheme is adopted).

2.4.2 Domain of discretization

The generation of a random field requires the definition of a set of points (or mesh) where it will be evaluated. UQLAB provides ways to do it:

- Define the input domain $\mathcal{D}$, through `RFInput.Domain`, together with the number of points along each direction `RFInput.SPD`. This results in a uniform grid over the input domain. `RFInput.Domain` is a $2 \times d$ vector where the first and second row represent the lower and upper bounds of the domain, respectively, and d is the dimension of the random field. `RFInput.SPD` is a scalar in case of one-dimensional random fields. For multidimensional fields, it may either be a vector with size $1 \times d$ or a scalar, in which case the same value is repeated in all directions.
  The following code creates a one-dimensional regular mesh with 200 nodes on $\mathcal{D} = [-1, 1]$:

  ```
  RFInput.Domain  = [-1; 1];
  RFInput.SPD  = 200;
  ```

- Alternatively, one may directly provide the coordinates of the mesh nodes using the `.Mesh`, field as shown in Section 2.1.3. `RFInput.Mesh` is a matrix of size $m \times d$, where m is the number of nodes in the mesh.

2.4.3 Covariance mesh

The mesh defined using the options described in Section 2.4.2 is used for sampling the random field. To actually perform the spectral decomposition of the covariance, however, a different mesh is used in general. The actual mesh depends on the discretization method considered and the properties of the field. In some cases, this mesh is entirely determined by the method and cannot be modified by the user.

Note: For both KL and EOLE, it is recommended to use a non-regular grid only for problems of dimension larger than 2.

Karhunen-Loève

For Karhunen-Loève, the mesh used to compute the covariance mesh cannot be specified directly. However the number of samples may be controlled depending on the chosen solution scheme:

- *Analytical/discrete solution*: in both cases the covariance matrix is computed directly on the sampling mesh `RFInput.Mesh`. No modification can be made by the user.
- *Nyström approach*: in this case, the samples are obtained by quadrature. The user may specify the quadrature type and the number of nodes. The former can be specified using the option `.KL.Quadrature`. For instance, to define a full grid, the following code may be used:

```
RFInput.KL.Quadrature = 'Full' ;
```

  Currently, only Gaussian full quadrature is supported by UQLAB. The number of quadrature nodes may be specified in one of two ways:

  - the number of nodes per correlation length per dimension S_{pc} with `.KL.SPC`. The total number of quadrature nodes n is obtained by tensorization, *i.e.*, $n = \prod_{i=1}^{d} n_i$ where n_i is the number of points in each direction, given by:

$$n_i = 1 + \lceil S_{\text{pc}} \times \Delta\mathcal{D}_i / \ell_i \rceil, \tag{2.1}$$

    where $\Delta\mathcal{D}_i$ is the range of the discretization domain in the i-th dimension, ℓ_i is the correlation length in the i-th dimension and $\lceil\ \rceil$ is the ceiling operator;
  - the total number of nodes in each direction n_i may directly be given using `.KL.SPD` as a row vector.

  Note that the exact total number of samples cannot be directly provided when using quadrature.

Expansion optimal linear estimation (EOLE)

For EOLE, the covariance mesh may be different from the sampling mesh defined in Section 2.4.2. By default and for random fields with dimension smaller than 3, it is a regular grid with 5 samples points are used per correlation length in each direction. For higher-dimensional random fields, the mesh is generated randomly using Latin hypercube sampling (LHS).

In case of EOLE, there are many ways to specify the covariance mesh:

- The user can directly enter a mesh which is consistent with the dimension of the random field using the following option:

  ```
  RFInput.EOLE.CovMesh  = Z;
  ```

  Here `Z` is a vector of size $n \times d$, where n is the number of points in the mesh. Any sampling scheme can be used, even a non-regular grid. The user must however make sure that the number of points is large enough to obtain a good approximation accuracy.

- The user may directly enter the number of nodes per correlation length and dimension, together with a sampling technique. For example, to use 10 points per correlation length and a regular grid obtained by full tensorization, one may specify:

  ```
  RFInput.EOLE.SPC = 5 ;
  RFInput.EOLE.Sampling = 'grid' ;
  ```

  In this case, the number of nodes per i-th dimension is given by Eq. (2.1) and the total number of points in the mesh is $n = \prod_{i=1}^{d} n_i$.

- The user may directly specify the total number of nodes per dimension n_i and the sampling technique. The multi-dimensional mesh is created by tensorization. For instance, the following code produces a 2D mesh with 30 nodes in the first dimension and 20 in the second, sampled using LHS:

  ```
  RFInput.EOLE.SPD = [30 20] ;
  RFInput.EOLE.Sampling = 'LHS' ;
  ```

  In this example, a total of $n = 600 = 30 \times 20$ samples are drawn uniformly by LHS in the discretization domain (*i.e.*, without any tensorization).

- Finally, the user may directly specify the total number of samples and the sampling technique. For instance, the user may specify the following to create a mesh containing $n = 1000$ uniformly distributed points using LHS:

  ```
  RFInput.EOLE.NSamples = 1000 ;
  RFInput.EOLE.Sampling = 'LHS' ;
  ```

2.4.4 Correlation function

The correlation function is defined using the UQLIB library (See UQLIB user manual). Any kernel available in UQLAB can be used within the random field module.

The main ingredients for specifying the correlation function are:

- Correlation type: separable or ellipsoidal (see UQLIB user manual, Section 2.3.1). For example, in order to use a separable correlation function the following option is needed:

```
RFInput.Corr.Type = 'Separable';
```

- Correlation family: the kernel family (see UQLIB user manual, Section 2.3.1). For example, in order to use an exponential correlation function the following option is needed:

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

- Correlation length: The scaling parameters for the correlation function. It may be a scalar for one-dimensional and isotropic multi-dimensional cases. For an anisotropic random field, the vector must have d elements. As an example, the correlation length for an anisotropic two-dimensional random field in may be specified as follows:

```
RFInput.Corr.Length = [0.3 0.5] ;
```

For more details about the configuration options available for the correlation function, please refer to the UQLIB user manual. In Figure 3, various one-dimensional random field trajectories are plotted using different correlation families.

2.5 Non-Gaussian translation random fields

The module currently supports only Gaussian random fields and a special class of non-Gaussian ones, known as translation random fields. Such fields are obtained by iso-probabilistically transforming Gaussian random fields, such that the marginals at each mesh point follow a desired non-Gaussian distribution (*e.g.*, lognormal or Gumbel), while the copula remains Gaussian.

A popular example of such fields is the lognormal random field, often used to describe parameters that are constrained to be positive. Such a random field can be generated in UQLAB using the following command:

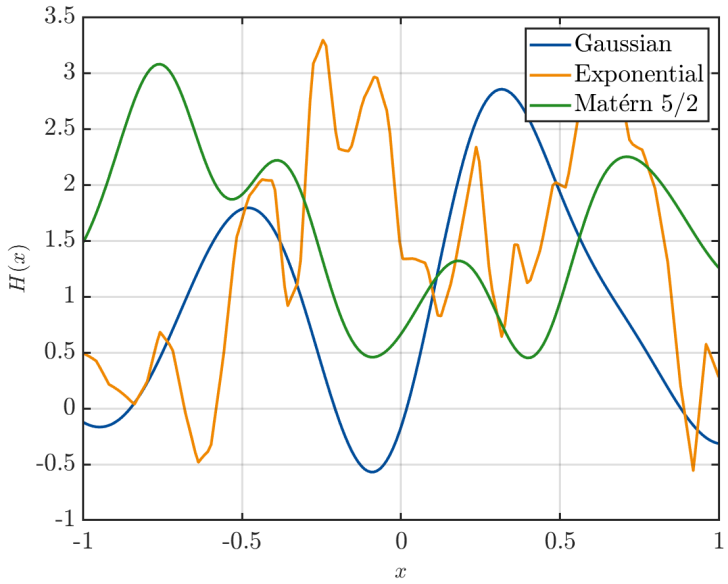


Figure 3: Three random field trajectories using the Gaussian, exponential and Matérn $5/2$ correlations families.

```
RFInput.RFType = 'Lognormal';
```

In general, UQLAB allows setting any marginal that is completely defined by its mean and variance using the command `RFType`.

Finally, once a random field is created, the user can easily transform its realizations into ones with a different marginal distribution but the same moments by using the helper function:

```
Y = uq_translateRF(X,RF, TargetDistrib)
```

where `X` are realizations of a random field in from object `RF` and `TargetDistrib` is the desired distribution.

Currently any marginal distribution available in UQLAB (See Section 1.2 of UQLAB User Manual – the INPUT module) and uniquely defined by its moments can be used.

2.6 Conditional random fields

Configuring a conditional random field is very similar to setting up a Gaussian RF. In this case, a set of observations, defined as a set of locations at which the random field values are known, is needed. The data can be provided through the `RFData.X` and `RFData.Y` fields:

```
RFInput.RFData.X= [-0.5 0 0.5]';
RFInput.RFData.Y = [0.5, 1, 1.5]';
```

`RFInput.RFData.X` and `RFInput.RFData.Y` are matrices of sizes $K \times d$ and $K \times 1$, respectively, that contain the locations and values of the observations and K is the number of data points.

Figure 4 shows an illustration of the random field obtained using the example defined in Section 2.1.1 together with the conditional data added in the previous block of code.

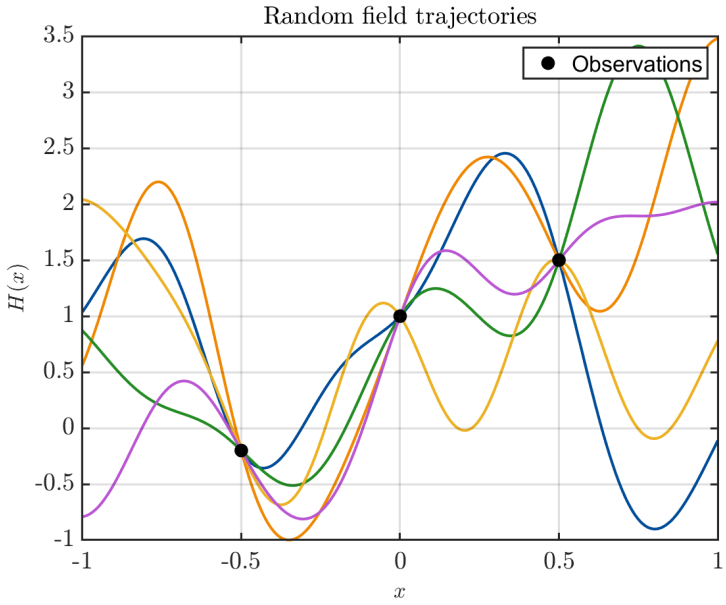


Figure 4: Five realizations of a conditional random field using EOLE discretization.

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 Creating an INPUT object of type Random field: `uq_createInput`

Syntax

```
myRF = uq_createInput(RFInput)
```

RFInput

The `struct` variable `RFInput` contains the description of the main ingredients needed to generate a random field. The content of the `RFInput` structure is listed in Table 1.

Table 1: `RFInput`

●	`.Type`	`'Randomfield'`	Type of input object
☐	`.RFType`	String default: `'Gaussian'`	Random field marginal distribution
		`'Gaussian'`	Gaussian distribution (Section A.2)
		`'Lognormal'`	Lognormal distribution (Section A.3)
		`'Uniform'`	Uniform distribution (Section A.1)
		`'Exponential'`	Exponential distribution (Section A.8)
		`'Gumbel'`	Gumbel maximum extreme value distribution (Section A.4)
		`'GumbelMin'`	Gumbel minimum extreme value distribution (Section A.5)
		`'Gamma'`	Gamma distribution (Section A.7)
		`'Logistic'`	Logistic distribution (Section A.11)
		`'Laplace'`	Laplace distribution (Section A.12)
☐	`.DiscScheme`	String default: `'EOLE'`	Discretization method
		`'KL'`	Karhunen-Loève expansion
		`'EOLE'`	Expansion optimal linear estimation
☐	`.KL`	Table 2	Options for the KL discretization
☐	`.EOLE`	Table 3	Options for the EOLE discretization
●	`.Corr`	Table 4	Options for the correlation function
●	`.Mean`	Double	Mean of the random field
●	`.Std`	Double	Standard deviation of the random field

☐	`.EnergyRatio`	Double default: 0.99	Energy ratio • Should be a positive double smaller than 1.
☐	`.OrderExp`	Integer	Expansion order M • Its value is computed according to the target energy ratio, using Eq. (1.29)
⊕	`.Mesh`	$m \times d$ Double	Mesh used to generate trajectories of the random field
⊕	`.Domain`	$2 \times d$ Double	Domain of discretization • First and second row contain the lower and upper bounds, respectively. • It is not taken into account if the `.Mesh` option is defined.
⊕	`.SPD`	Integer	Number of points per dimension needed to generate a mesh when the `.Domain` is defined • A scalar in one-dimensional cases. • A vector of length d or a scalar in multi-dimensional cases. When a scalar is given, the same value is repeated along each direction.
☐	`.RFData`	Table 5	Observation data for conditional random fields (Section 1.4)

When using Karhunen-Loève, the options listed in Table 2 may be specified

Table 2: `RFInput.KL`

☐	`.Method`	String default: `'Nystrom'` or `'Analytical'` when using the exponential kernel	Numerical solution scheme (See Section 1.3.1.1)
		`'Nystrom'`	Nyström solution scheme
		`'Analytical'`	Analytical solution scheme. Only available when using the exponential kernel
		`'Discrete'`	Discrete / PCA approach
☐	`.SPC`	Scalar or $1 \times d$ Double default: 5	Number of samples per correlation length per dimension for the computation of the covariance mesh (See Section 2.4.3). The total number of samples is given by Eq. (2.1).
☐	`.SPD`	Scalar or $1 \times d$ Double	Total number of samples per dimension for the computation of the covariance mesh (See Section 2.4.3).

When using EOLE, the options listed in Table 3 may be specified. The user may either provide a mesh through the option `.EOLE.CovMesh` or provide the sampling technique and number of sample points.

Table 3: `RFInput.EOLE`			
☐	`.CovMesh`	$n \times d$ Double	Mesh used for the computation of the covariance (See Section 2.4.3). Its dimension should be consistent with that of the random field.
☐	`.Sampling`	String default: `'Grid'` when $d < 3$, `'LHS'` otherwise	Sampling method for the covariance mesh
		`'Grid'`	Full grid
		`'MCS'`	Monte Carlo Sampling
		`'LHS'`	Latin hypercube Sampling
		`'Sobol'`	Sobol series
		`'Halton'`	Halton series
☐	`.SPC`	Scalar or $1 \times d$ Double default: 5	Number of samples per correlation length per dimension for the computation of the covariance mesh (See Section 2.4.3). The total number of samples is given by Eq. (2.1).
☐	`.SPD`	Scalar or $1 \times d$ Double	Number of samples per dimension for the computation of the covariance mesh (See Section 2.4.3).
☐	`.NSamples`	Scalar or $1 \times d$ Double	Total number of samples for the computation of the covariance mesh (See Section 2.4.3) • Ignored when using quadrature. • When $d < 3$, the default number of samples is computed using Eq. (2.1). • When $d > 3$, the default number of samples is 2000.

The options for the correlation function are given in `RFInput.Corr` and listed in Table 4. More details are given in UQLab user manual.

Table 4: `RFInput.Corr`			
☐	`.Type`	String default: `'Separable'`	The correlation function type
		`'Separable'`	Separable correlation function
		`'Ellipsoidal`	Ellipsoidal correlation function
☐	`.Family`	String default: `'Gaussian'`	The correlation family, that is the elementary 1-D function that is used by the correlation function.
		`'Exponential'`	Exponential correlation function

		'Gaussian'	Gaussian correlation function
		'Linear'	Linear correlation function
		'Matern-3_2'	Matérn-3/2 correlation function
		'Matern-5_2	Matérn-5/2 correlation function
⊞	.Length	Scalar or $1 \times d$ Double	Correlation length

For conditional random fields the data pair $(\mathcal{X}, \mathcal{Y})$ defined in Section 1.4 can be provided under RFInput.RFData whose subfields are listed in Table 5.

Table 5: RFInput.RFData

●	.X	$K \times d$ Double	Location of the observations $\mathcal{X} = \left\{\boldsymbol{x}^{(i)} \in \mathbb{R}^d, i = 1, \ldots, K\right\}$
●	.Y	$K \times 1$ Double	Values of the observations $\mathcal{Y} = \left\{y^{(i)} = H\left(\boldsymbol{x}^{(i)}\right) \in \mathbb{R}, i = 1, \ldots, K\right\}$

Output

After uq_createInput completes its operation a new INPUT object is created that contains the following fields:

Table 6: myRF

.Type	String	The input object of Type 'randomfield'
.Name	String	Unique name of the random filed
.Internal	Table 7	Internal state of the RF object (useful for debug/diagnostics)
.RF	Table 8	Information about the final RF object
.Options	Table 1	Copy of the RFInput structure used to create the random field
.UnderlyingGaussian	INPUT object	Probabilistic input model for the underlying standard Gaussian random variables sampled to generate trajectories of the random field.

3.1.1 Internal fields (advanced)

The .Internal field contains all the information necessary to discretize and sample a random field. It contains the options given by the user and the one set by default by the module.

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

Table 7: `myRF.Internal`		
`.Runtime`	Structure	Parameters not corresponding to any option of the module and that are created only for internal use.
`.RF`	Structure	Internal fields with data related to the random field input object

Table 8: `myRF.RF`		
`.Eigs`	$1 \times M$ Double	Eigenvalues of the correlation matrix.
`.Phi`	$n \times M$ Double	Eigenvectors of the correlation matrix.
`.VarError`	$n \times d$ Double	Variance error as computed on the discretization mesh.

3.2 Getting samples from an INPUT object: `uq_getSample`

Syntax

```
X = uq_getSample(N)
X = uq_getSample(myInput,N)
X = uq_getSample(N,method)
X = uq_getSample(myInput,N,method)
X = uq_getSample(N,method,Name,Value)
X = uq_getSample(myInput,N,method,Name,Value)
X = uq_getSample(...)
[X, xi] = uq_getSample(...)
```

Description

`X = uq_getSample(N)` returns N samples of a random field/vector defined in the currently selected INPUT module using the default sampling method. If not set otherwise by the user, the default sampling method used to generate the ξ_i's variables is `'MC'` (Monte Carlo). The user can call the function `uq_setDefaultSampling` (described in the UQLAB User Manual – the INPUT module) to change the default sampling method.

`X = uq_getSample(myInput,N)` returns N samples of the random field/vector defined in the INPUT object `myInput` using the default sampling method.

`X = uq_getSample( myInput,N,method)` returns N samples of a random field/vector defined in the currently selected INPUT object using the sampling method defined in `method`. This is a string that can take one of the following values:

Table 9: `method` option of `uq_getSample`	
`'MC'`	Monte Carlo
`'LHS'`	Latin Hypercube
`'Sobol'`	Sobol series
`'Halton'`	Halton series

`X = uq_getSample( myInput,N,method, Name, Value)` allows for specification of additional `Name - Value` pairs of options. The supported options are listed in Table 10.

Table 10: Available options of `uq_getSample`		
`Name`	`Value`	Description
`method = 'LHS'`		
`'LHSiterations'`	Integer default: 5	Maximum number of iterations to perform in an attempt to improve the design. For more information refer to the MATLAB function `lhsdesign`.

Output

Note: The options described below are defined in the context of random field. The output of `uq_getSample` may be different when the default or selected INPUT object is not a random field.

`X = uq_getSample(...)` returns N realizations of the random field.

`[X, xi] = uq_getSample(...)` additionally returns an N-by-M matrix (`xi`) corresponding to the random variables ξ_i used in the expansion.

3.2.1 Printing information: `uq_print`

Syntax

```
uq_print(myRF);
```

Description

`uq_print(myRF);` prints a report about the INPUT object `myRF` (Discretization scheme, correlation function, expansion order, variance error and explained variance).

3.2.2 Graphical visualization: `uq_display`

Syntax

```
uq_display(myRF);
```

Description

`uq_display(myRF)` produces one or three plots:

- Five or one realizations of the random field for problems of dimension 1 and 2 respectively (for higher dimensional problem, this plot is not produced).
- The variance error as computed over the discretization mesh (for higher dimensional problem, this plot is not produced).
- The M first eigenvalues of the spectral decomposition of the correlation matrix.

References

Atkinson, K. E. (1997). *The Numerical Solution of Integral Equations of the Second Kind*, volume 4. Cambridge University Press. 5

Betz, W., Papaioannou, I., and Straub, D. (2014). Numerical methods for the discretization of random fields by means of Karhunen-loève expansion. *Comput. Methos Appl. Mech. Engrg.*, 271:109–129. 6, 9

Ditlevsen, O. (1996). Dimension reduction and discretization in stochastic problems by regression method. In Casciati, F. and Roberts, B., editors, *Mathematical models for structural reliability analysis, CRC Mathematical Modelling Series*, chapter 2, pages 51–138. 3

Ghanem, R. and Spanos, P.-D. (1991). Spectral stochastic finite-element formulation for reliability analysis. *J. Eng. Mech.*, 117(10):2351–2372. 1, 3

Ghanem, R. G. and Spanos, P. D. (2003). *Stochastic finite elements: a spectral approach*. Courier Corporation. 5

Grigoriu, M. (1984). Crossing of non-Gaussian translation processes. *J. Eng. Mech.*, 110:610–620. 9

Karhunen, K. (1947). Under Lineare Methoden in der Wahr Scheinlichkeitsrechnung. *Annales Academiae Scientiarun Fennicae Series A1: Mathematia Physica*, 47. 1

Li, C. and Der Kiureghian, A. (1993). Optimal discretization of random fields. *J. Eng. Mech.*, 119(6):1136–1154. 1, 3, 6, 7

Lin, Y. (1967). Probabilistic theory of structural dynamics. *New York*, pages 214–215. 2

Marzouk, Y. M. and Najm, H. N. (2009). Dimensionality reduction and polynomial chaos acceleration of Bayesian inference in inverse problems. *Journal of Computational Physics*, 228(6):1862–1902. 1

Matthies, G., Brenner, C., Bucher, C., and Guedes Soares, C. (1997). Uncertainties in probabilistic numerical analysis of structures and solids – Stochastic finite elements. *Struct. Saf.*, 19(3):283–336. 3

McKay, M. D., Beckman, R. J., and Conover, W. J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 2:239–245. 16

Phoon, K., Huang, H., and Quek, S. (2005). Simulation of strongly non Gaussian processes using Karhunen-Loève expansion. *Prob. Eng. Mech.*, 20(2):188–198. 4

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007). *Numerical recipes: The art of scientific computing*. Cambridge university press. 5, 6

Schenk, C. A. and Schuëller, G. I. (2005). *Uncertainty assessment of large finite element systems*, chapter PKarhunen-Loève expansion, pages 47–58. Springer, Berlin, Heidelberg. 6

Spanos, P.-D. and Ghanem, R. (1989). Stochastic finite element expansion for random media. *J. Eng. Mech.*, 115(5):1035–1053. 1

Sudret, B. and Der Kiureghian, A. (2000). *Stochastic finite element methods and reliability: a state-of-the-art report*. Department of Civil and Environmental Engineering, University of California. 1, 3

Vanmarcke, E. and Grigoriu, M. (1983). Stochastic finite element analysis of simple beams. *J. Eng. Mech.*, 109(5):1203–1214. 2

Vanmarcke, E., Shinozuka, M., Nakagiri, S., Schueller, G., and Grigoriu, M. (1986). Random fields and stochastic finite elements. *Struct. Saf.*, 3(3-4):143–166. 2