ETH zürich

UQLab user manual
The Input module

C. Lataniotis, S. Marelli, B. Sudret

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

Risk, Safety & Uncertainty Quantification

How to cite UQLAB

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

How to cite this manual

C. Lataniotis, S. Marelli and B. Sudret, UQLAB user manual – The Input module, Report UQLab-V1.2-102, Chair of Risk, Safety & Uncertainty Quantification, ETH Zurich, 2019.

BIBTEX entry

```
@TechReport{UQdoc_12_102,
author = {Lataniotis, C. and Marelli, S. and Sudret, B.},
title = {{UQLab user manual -- The Input module}},
institution = {Chair of Risk, Safety \& Uncertainty Quantification, ETH Zurich},
year = {2019},
note = {Report \# UQLab-V1.2-102},
}
```

Document Data Sheet

Document Ref.	UQLAB-V1.2-102
Title:	UQLAB user manual – The Input module
Authors:	C. Lataniotis, S. Marelli, B. Sudret Chair of Risk. Safety and Uncertainty Quantification, ETH Zurich, Switzerland
Date:	22/02/2019

Doc. Version	Date	Comments
V1.2	22/02/2019	UQLAB V1.2 release
V1.1	05/07/2018	UQLAB V1.1 release • New section in the reference list about uq_subsample
V1.0	01/05/2017	UQLAB V1.0 release • New distributions: triangular, logistic, Laplace • Updated custom distributions section • Updated description of several functions
V0.9	01/07/2015	Initial release

Abstract

The UQLAB INPUT module is used to define the probabilistic input model in uncertainty quantification problems. It offers extensive possibilities to perform operations like drawing samples of random vectors, or transforming samples of random vectors to samples of different random vectors (isoprobabilistic transforms). The dependence structure between the components of random vectors is specified with the copula formalism.
This user manual includes a review of the methods that are used to define, draw and transform samples of random vectors. It also contains information about each of the available probability distributions that can be used in the current version of UQLAB. After introducing the theoretical aspects, an in-depth example-driven user guide is provided to help new users to properly set up and use the INPUT module objects. Finally, a comprehensive reference list of the methods and functions available in the UQLAB INPUT module is given at the end of the manual.

Keywords: Probabilistic Input Model, Marginals, Copula, Sampling

Contents

Chapter 1

Theory

1.1 Introduction

Identification and modelling of the sources of uncertainty are crucial steps for the solution of any uncertainty quantification problem. In a probabilistic setting, each uncertain model parameter can be represented by a random variable and a corresponding probability density function (PDF) in the form $X \sim f_X(x)$. Several input parameters, including their dependence structure, can be grouped together in a random vector with joint PDF $\boldsymbol{X} \sim f_{\boldsymbol{X}}(\boldsymbol{x})$. The INPUT module in UQLAB offers a suite of tools to handle the representation, transformation and sampling of a wide variety of PDFs and joint PDFs.

1.2 Representation of common univariate distributions

A random variable can be represented by its univariate PDF $X \sim f_X(x)$. UQLAB supports a number of distributions employed in many fields of applied sciences, namely:

- Uniform
- Normal or Gaussian
- Lognormal
- Gumbel (maxima)
- Gumbel (minima)
- Beta
- Gamma
- Exponential
- Weibull
- Triangular

- Logistic
- Laplace

In this section, a brief overview of each distribution and its properties (*e.g.*PDF, cumulative distribution function, support, moments,parameters) is given for reference.

1.2.1 Uniform distribution

Uniform distributions are commonly used to represent variables with unknown moments and known support. The uniform distribution is the maximum entropy distribution on any closed support.

$$
\begin{aligned}
&\text{Notation} && : X \sim \mathcal{U}(a,b)\\
&\text{Parameters} && : a,b \in \mathbb{R}\ ;\ a<b\\
&\text{Support} && : \mathcal{D}_X = [a,b]\\
&\text{PDF} && : f_X(x) = \frac{\mathbb{1}_{[a,b]}(x)}{b-a} = \begin{cases} \frac{1}{b-a} & \text{if } x \in [a,b] \\ 0 & \text{if } x \notin [a,b] \end{cases}\\
&\text{CDF} && : F_X(x) = \frac{x-a}{b-a}\mathbb{1}_{[a,b]}(x) = \begin{cases} 0 & \text{if } x \le a \\ \frac{x-a}{b-a} & \text{if } x \in [a,b] \\ 1 & \text{if } x \ge b \end{cases}\\
&\text{Moments} && : \mu_X = \frac{a+b}{2}\\
& && \quad\ \sigma_X = \frac{b-a}{2\sqrt{3}}
\end{aligned}
$$

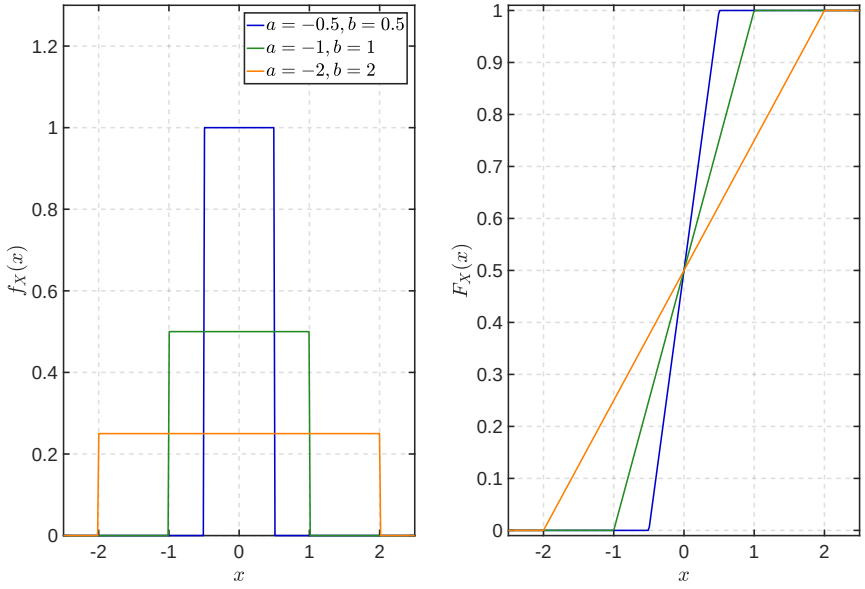

Figure 1: PDF and CDF of uniform distributions for various parameter values.

A particularly important uniform distribution in the field of numerical statistics is $X = \mathcal{U}(0,1)$. In fact, every class of random number generators produces samples from this PDF (or its multidimensional version), which are then manipulated to produce samples distributed according any other distribution as needed. For details, see Section 1.5.3.

1.2.2 Gaussian (Normal)

Gaussian distributions are another basic family of PDF that are pervasive throughout any fields of applied science. They are commonly employed to represent measurement error, noise terms etc..

$$
\begin{aligned}
\text{Notation} \quad &: X \sim \mathcal{N}(\mu, \sigma) \\
\text{Parameters} &: \mu \in \mathbb{R}\ ,\ \sigma > 0 \\
\text{Support} \quad &: \mathcal{D}_X = \mathbb{R} \\
\text{PDF} \quad &: f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \\
\text{CDF} \quad &: F_X(x) = \frac{1}{2}\left[1 + \text{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right] \\
&\quad = \Phi\left(\frac{x-\mu}{\sigma}\right) \\
\text{Moments} \quad &: \mu_X = \mu \\
&\quad \sigma_X = \sigma
\end{aligned}
$$

where μ is the mean, σ^2 the variance and $\text{erf}(\cdot)$ is the error function, defined by:

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \tag{1.1}$$

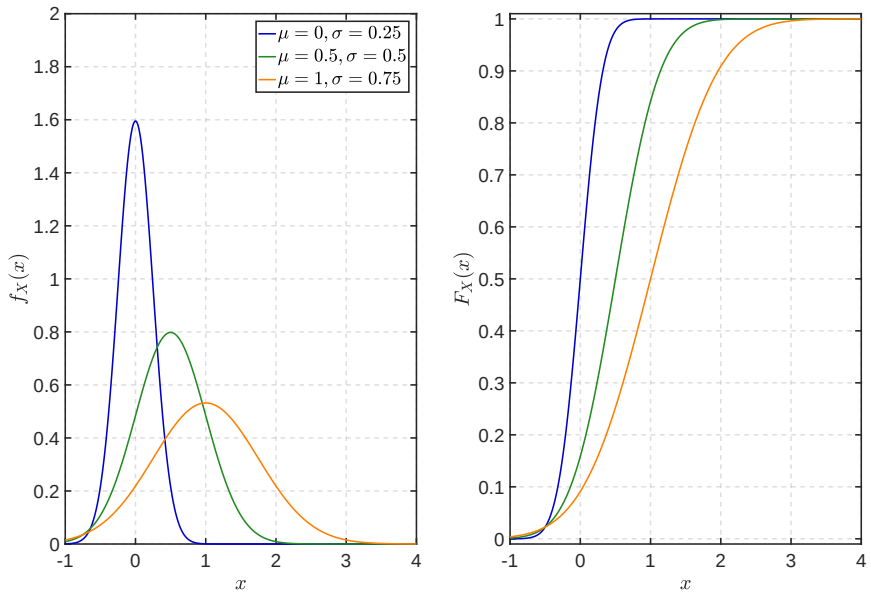

Figure 2: PDF and CDF of Gaussian distributions for various parameter values.

An important distribution belonging to the normal family is the so-called *standard normal distribution* $\mathcal{N}(0,1)$, characterized by $\mu = 0,\ \sigma = 1$. The notation $\Phi(x)$ is used to identify the standard normal CDF:

$$\Phi(x) = \int_{-\infty}^{x} \frac{e^{-t^2/2}}{\sqrt{2\pi}} dt \tag{1.2}$$

All normal distributions can be represented as linear transforms of standard normal distributions as follows:

$$\mathcal{N}(\mu, \sigma) = \mu + \sigma\mathcal{N}(0, 1) \tag{1.3}$$

1.2.3 Lognormal distribution

A lognormal variable is a random variable $X \sim \mathcal{LN}(\lambda, \zeta)$ such that its logarithm is a Gaussian variable:

$$X \sim \mathcal{LN}(\lambda, \zeta) \Leftrightarrow \ln(X) \sim \mathcal{N}(\lambda, \zeta) \tag{1.4}$$

$$\begin{aligned}
&\text{Notation} && : X \sim \mathcal{LN}(\lambda, \zeta) \\
&\text{Parameters} && : \lambda \in \mathbb{R}\ ,\ \zeta > 0 \\
&\text{Support} && : \mathcal{D}_X = (0, +\infty) \\
&\text{PDF} && : f_X(x) = \frac{1}{\sqrt{2\pi}\zeta x} \exp\left(-\frac{(\ln\, x - \lambda)^2}{2\zeta^2}\right) \\
&\text{CDF} && : F_X(x) = \frac{1}{2} + \frac{1}{2}\text{erf}\left(\frac{\ln\, x - \lambda}{\sqrt{2}\zeta}\right) \\
& && \quad = \Phi\left(\frac{\ln x - \lambda}{\zeta}\right) \\
&\text{Moments} && : \mu_X = e^{\lambda + \zeta^2/2} \\
& && \quad \sigma_X = e^{\lambda + \zeta^2/2}\sqrt{e^{\zeta^2} - 1}
\end{aligned}$$

where λ and ζ are the mean and standard deviation *of the natural logarithm of the variable* and the error function is defined in Eq. (1.1).

Lognormal distributions are commonly used in Engineering to describe parameters which are positive in nature, such as material or physical properties. An important property of lognormally distributed variables is that their products and ratios are also lognormally distributed.

1.2.4 Gumbel distribution

The Gumbel distribution is also referred to as Extreme Value distribution of type I (EV I). Note that in the literature the name 'Gumbel distribution' is used to refer to either the maximum or the minimum extreme value distribution. In UQLAB "Gumbel" refers to the *maximum*

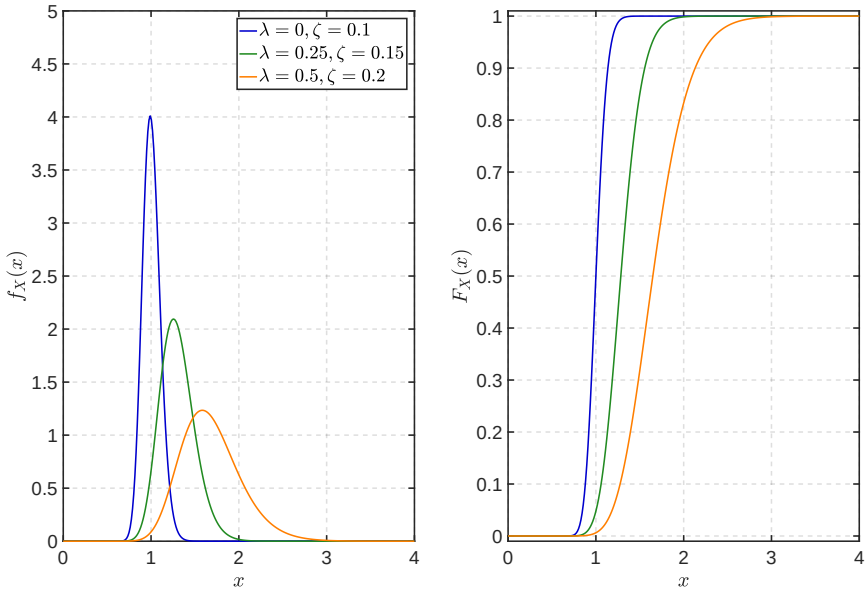

Figure 3: PDF and CDF of lognormal distributions for various parameter values.

Gumbel distribution .

Notation	: $X \sim \mathcal{G}(\mu, \beta)$
Support	: $\mathcal{D}_X = \mathbb{R}$
Parameters	: $\mu \in \mathbb{R}$, $\beta > 0$
PDF	: $f_X(x) = \frac{1}{\beta} e^{-\frac{x-\mu}{\beta} - e^{-\frac{x-\mu}{\beta}}}$
CDF	: $F_X(x) = e^{-e^{-\frac{x-\mu}{\beta}}}$
Moments	: $\mu_X = \mu + \beta\gamma_e$
	where $\gamma_e = 0.577216\ldots$ is the Euler constant
	$\sigma_X = \frac{\pi\beta}{\sqrt{6}}$

Note that parameter μ coincides with the mode of the distribution.
The Gumbel distribution is used for modelling random variables obtained as the maximum of identically distributed variables. It is used for instance in hydrology to model flood intensity.

1.2.5 Gumbel-min distribution

The Gumbel-min is also referred to as the Smallest Extreme Value (SEV) distribution or the Smallest Extreme Value (Type I) distribution.

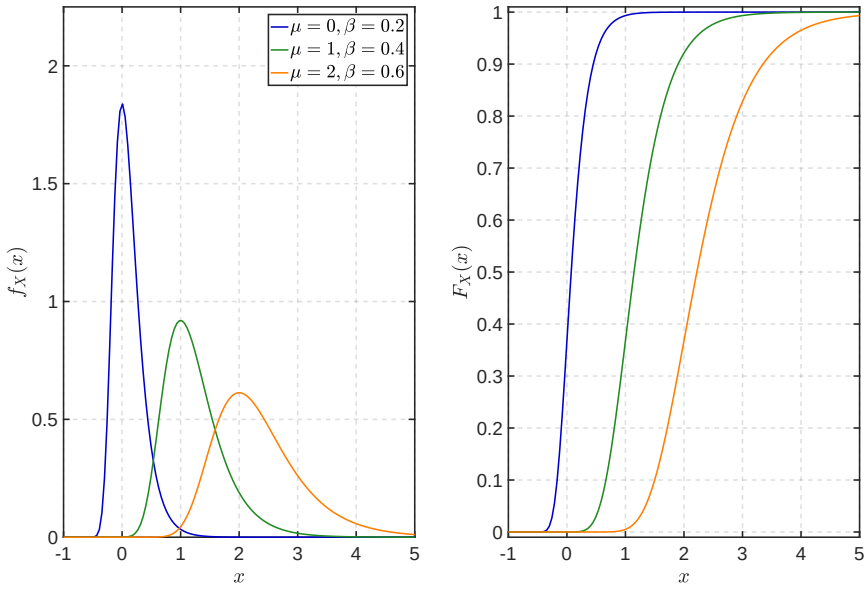

Figure 4: PDF and CDF of Gumbel maximum extreme value distributions for various parameter values.

Notation	: $X \sim \mathcal{G}(\mu, \beta)$
Parameters	: $\mu \in \mathbb{R}$, $\beta > 0$
Support	: $\mathcal{D}_X = \mathbb{R}$
PDF	: $f_X(x) = \frac{1}{\beta} e^{\frac{x-\mu}{\beta} + e^{-\frac{x-\mu}{\beta}}}$
CDF	: $F_X(x) = 1 - e^{-e^{-\frac{x-\mu}{\beta}}}$
Moments	: $\mu_X = \mu - \beta\gamma_e$
	where $\gamma_e = 0.577216\ldots$ is the Euler constant
	$\sigma_X = \frac{\pi\beta}{\sqrt{6}}$

Note that parameter μ coincides with the mode of the distribution. .

The Gumbel-min distribution's PDF is skewed to the left, unlike the Gumbel-max which is skewed to the right (see Figures 4 and 5).

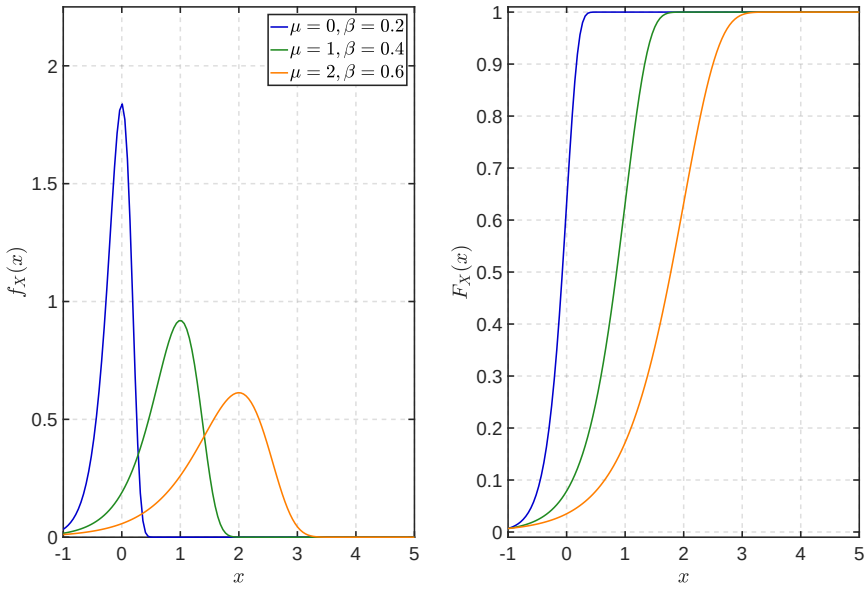

Figure 5: PDF and CDF of Gumbel-min extreme value distributions for various parameter values.

1.2.6 Weibull distribution

The Weibull distribution is the last type of extreme-value distributions. It is commonly employed to parametrize time-to-failure-type variables.

$$
\begin{aligned}
&\text{Notation} && : X \sim \mathcal{W}(\alpha, \beta) \\
&\text{Parameters} && : \alpha > 0\ ,\ \beta > 0 \\
&\text{Support} && : \mathcal{D}_X = [0, +\infty) \\
&\text{PDF :} && : f_X(x) = \begin{cases} \frac{\beta}{\alpha}\left(\frac{x}{\alpha}\right)^{\beta-1} e^{-(x/\alpha)^\beta} & \text{if } x \geq 0 \\ 0 & , x < 0 \end{cases} \\
&\text{CDF :} && : F_X(x) = \begin{cases} 1 - e^{-(x/\alpha)^\beta} & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \\
&\text{Moments} && : \mu_X = \alpha\,\Gamma\left(1 + 1/\beta\right) \\
& && \quad \sigma_X = \alpha\sqrt{\Gamma\left(1 + 2/\beta\right) - \Gamma\left(1 + 1/\beta\right)^2}
\end{aligned}
$$

Other uses of the Weibull distribution include the parametrization of strength or strength-related lifetime parameters, material strength and lifetime parameters for brittle materials (for which the *weakest-link-theory* is applicable).

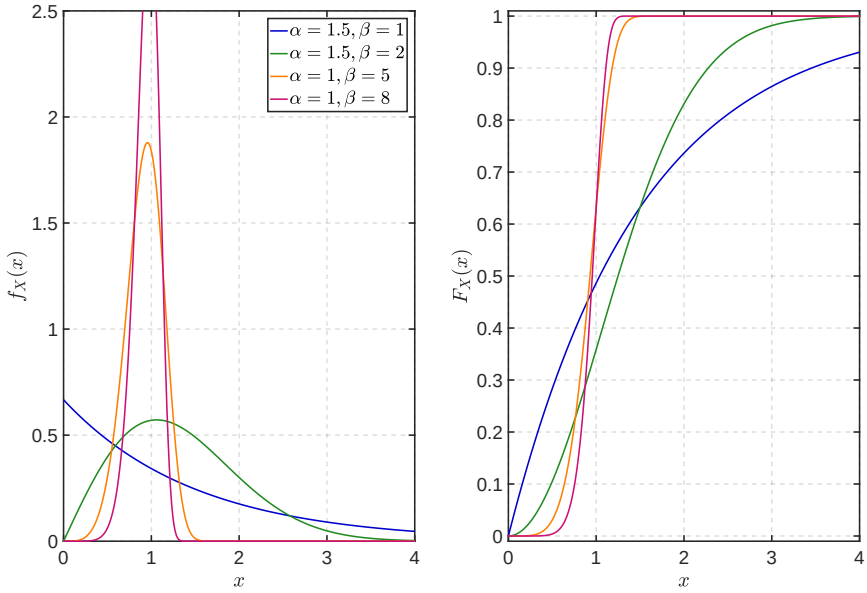

Figure 6: PDF and CDF of Weibull distributions for various parameter values.

1.2.7 Gamma distribution

The Gamma distribution may be used to model variables which are positive in nature, such as those connected to Poisson processes. Assuming events occur randomly in time in a Poisson process at a constant rate λ, the time to first occurrence follows an exponential distribution $\Gamma(\lambda, 1)$. The time to k-th occurrence follows a Gamma distribution $\Gamma(\lambda, k)$.

$$
\begin{aligned}
&\text{Notation} && : X \sim \Gamma(\lambda, k) \\
&\text{Parameters} && : \lambda > 0 \ , \ k > 0 \\
&\text{Support} && : \mathcal{D}_X = [0, +\infty) \\
&\text{PDF} && : f_X(x) = \frac{\lambda^k}{\Gamma(k)} x^{k-1} e^{-\lambda x} \\
&\text{CDF} && : F_X(x) = \frac{\gamma(k, \lambda x)}{\Gamma(k)} \\
&\text{Moments} && : \mu_X = \frac{k}{\lambda} \\
& && \quad \sigma_X = \frac{\sqrt{k}}{\lambda}
\end{aligned}
$$

where $\Gamma(x)$ is the Gamma function defined by

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt \tag{1.5}$$

and $\gamma(x, y)$ is the incomplete Gamma function defined by

$$\gamma(k, x) = \int_0^x t^{k-1} e^{-t} dt \tag{1.6}$$

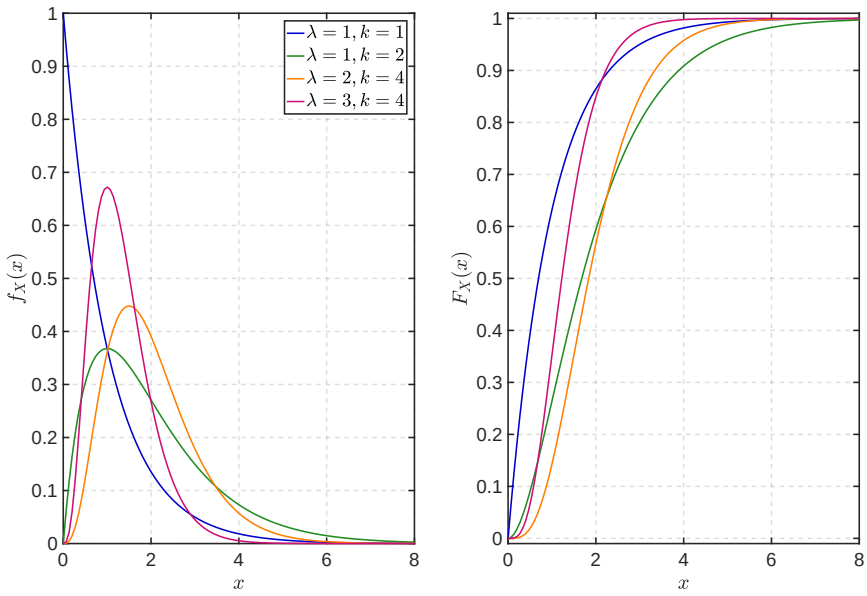

Figure 7: PDF and CDF of Gamma distributions for various parameter values.

A special case of Gamma distribution is $X \sim \Gamma(\lambda, 1)$, which corresponds to the exponential distribution (see Section 1.2.8).

1.2.8 Exponential distribution

A special case of the Gamma distribution, the exponential distribution is commonly used to represent the time to first-occurrence of Poissonian-type processes, *e.g.* radioactive decays.

$$\begin{aligned}
&\text{Notation} && : X \sim \mathcal{E}(\lambda) \\
&\text{Parameters} && : \lambda > 0 \\
&\text{Support} && : \mathcal{D}_X = [0, +\infty) \\
&PDF && : f_X(x) = \lambda e^{-\lambda x} \\
&CDF && : F_X(x) = 1 - e^{-\lambda x} \\
&\text{Moments} && : \mu_X = 1/\lambda \\
& && \quad \sigma_X = 1/\lambda
\end{aligned}$$

where $\lambda > 0$ is the scale parameter.
The Exponential distribution is also used to model the waiting times in queuing problems (*e.g.*for load balancing applications in large computational facilities).

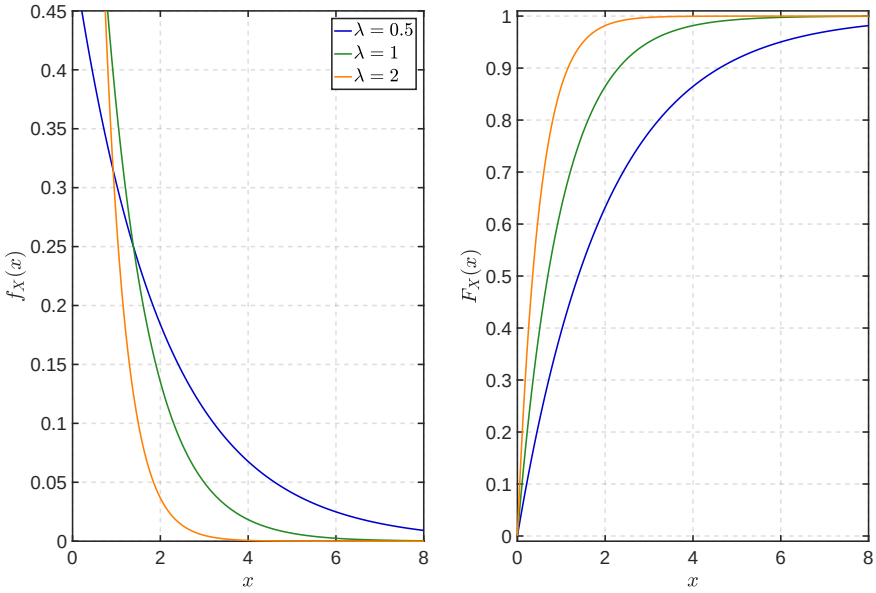

Figure 8: PDF and CDF of exponential distributions for various parameter values.

1.2.9 Beta distribution

The Beta distribution commonly used to model bounded variables.

Notation : $X \sim \mathcal{B}(r, s, a, b)$

Parameters : $r > 0\ ,\ s > 0$

Support : $\mathcal{D}_X = [a, b]$

PDF : $$f_X(x) = \begin{cases} \frac{(x-a)^{r-1}(b-x)^{s-1}}{(b-a)^{r+s-1}\,B(r,s)} & \text{if } x \in [a,b] \\ 0 & \text{if } x \notin [a,b] \end{cases}$$

CDF : $$F_X(x) = \begin{cases} 0 & \text{if } x \le a \\ \frac{1}{(b-a)^{r+s-1}\,B(r,s)} \int_a^x (t-a)^{r-1}\,(b-t)^{s-1}\,dt & \text{if } x \in [a,b] \\ 1 & \text{if } x \ge b \end{cases}$$

Moments : $$\mu_X = a + (b-a)r/(r+s)$$
$$\sigma_X = \frac{b-a}{r+s}\sqrt{\frac{r\,s}{r+s+1}}$$

where $B(r, s)$ is the Beta function:

$$B(r,s) = \int_0^1 t^{r-1}(1-t)^{s-1}dt = \frac{\Gamma(r)\Gamma(s)}{\Gamma(r+s)} \tag{1.7}$$

The range of the variable is given by the parameters $[a, b]$. The shape of the distribution is related to parameters $[r, s]$ and their ratio:

- When $r = s$ the PDF is symmetrical;
- If $r, s > 1$ the PDF is unimodal;
- if $r, s < 1$ the PDF is maximum at the boundaries.

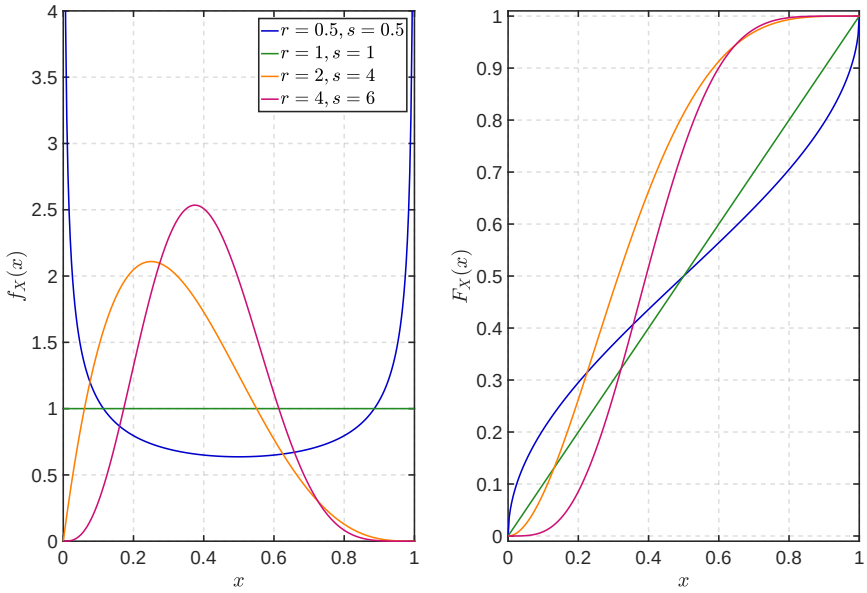

Figure 9: PDF and CDF of Beta distributions for various parameter values ($\mathcal{D}_X = [0, 1]$).

1.2.10 Triangular distribution

$$
\begin{aligned}
&\text{Notation} && : X \sim \text{Tr}(a, b, c)\\
&\text{Parameters} && : a \in \mathbb{R}\ ,\ a < b\ ,\ a < c < b\\
&\text{Support} && : \mathcal{D}_X = [a, b]\\
&\text{PDF} && : f_X(x) = \begin{cases} 0 & \text{if } x \leq a \\ \frac{2(x-a)}{(b-a)(c-a)} & \text{if } x \in (a, c) \\ \frac{2}{b-a} & \text{if } x = c \\ \frac{2(b-x)}{(b-a)(c-a)} & \text{if } x \in (c, b) \\ 0 & \text{if } x \geq b \end{cases}\\
&\text{CDF} && : F_X(x) = \begin{cases} 0 & \text{if } x \leq a \\ \frac{(x-a)^2}{(b-a)(c-a)} & \text{if } x \in (a, c] \\ 1 - \frac{(b-x)^2}{(b-a)(b-c)} & \text{if } x \in (c, b) \\ 1 & \text{if } x \geq b \end{cases}\\
&\text{Moments} && : \mu_X = \frac{a+b+c}{3}\\
& && \sigma_X = \frac{\sqrt{a^2+b^2+c^2-ab-ac-bc}}{3\sqrt{2}}
\end{aligned}
$$

The triangular distribution is typically used as a subjective description of a population for which there is only limited sample data. It is based on knowledge of the minimum and maximum and an "inspired guess" of the modal value.

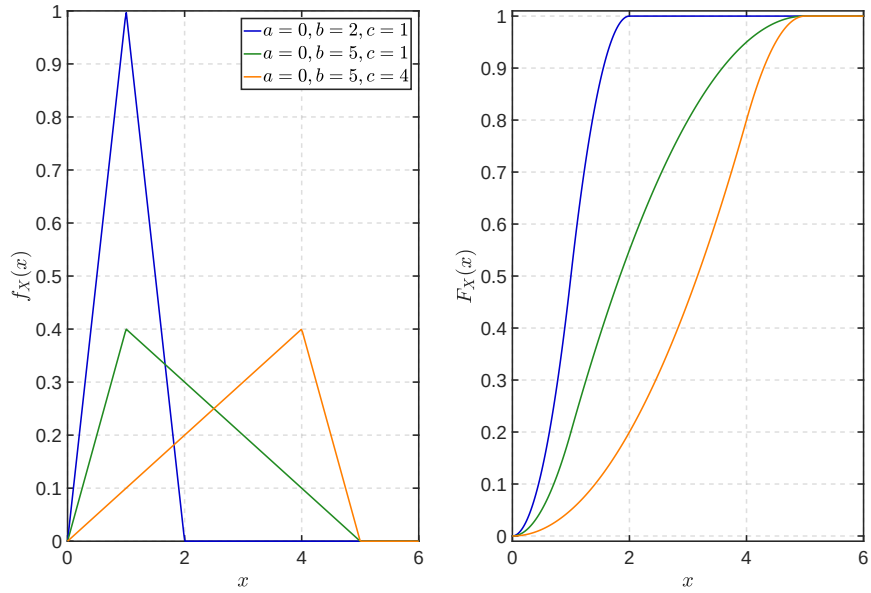

Figure 10: PDF and CDF of triangular distributions for various parameter values.

1.2.11 Logistic distribution

The logistic distribution resembles the normal distribution in shape but has heavier tails (higher kurtosis).

$$
\begin{aligned}
&\text{Notation} && : X \sim P(\mu, s) \\
&\text{Parameters} && : \mu \in \mathbb{R}\ ,\ s > 0 \\
&\text{Support} && : \mathcal{D}_X = \mathbb{R} \\
&\text{PDF} && : f_X(x) = \frac{e^{-\frac{x-\mu}{s}}}{s(1+e^{-\frac{x-\mu}{s}})^2} \\
&\text{CDF} && : F_X(x) = \frac{1}{1+e^{-\frac{x-\mu}{s}}} \\
&\text{Moments} && : \mu_X = \mu \\
& && \quad \sigma_X = \frac{s\pi}{\sqrt{3}}
\end{aligned}
$$

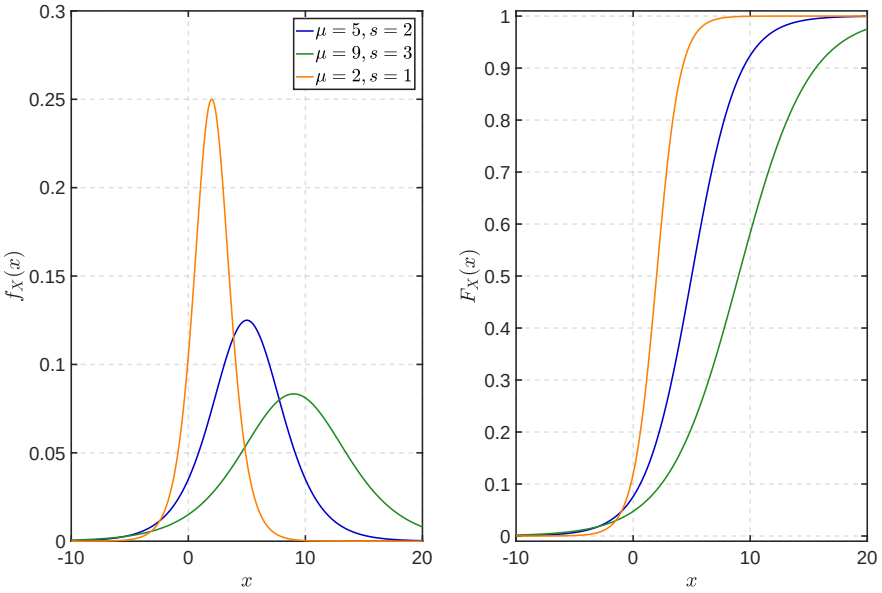

Figure 11: PDF and CDF of logistic distributions for various parameter values.

1.2.12 Laplace distribution

Laplace distribution is also known as double exponential distribution, because it can be thought of as two exponential distributions (with an additional location parameter) spliced together back-to-back.

$$
\begin{aligned}
&\text{Notation} && : X \sim \mathcal{L}(\mu, b) \\
&\text{Parameters} && : \mu \in \mathbb{R}\ ,\ b > 0 \\
&\text{Support} && : \mathcal{D}_X = \mathbb{R} \\
&\text{PDF} && : f_X(x) = \frac{1}{2b} \exp\left(- \frac{|x-\mu|}{b} \right) \\
&\text{CDF} && : F_X(x) = \begin{cases} \frac{1}{2} \exp\left(\frac{x-\mu}{b} \right) & \text{if } x < \mu \\ 1 - \frac{1}{2} \exp\left(- \frac{x-\mu}{b} \right) & \text{if } x \geq \mu \end{cases} \\
&\text{Moments} && : \mu_X = \mu \\
& && \quad \sigma_X = b\sqrt{2}
\end{aligned}
$$

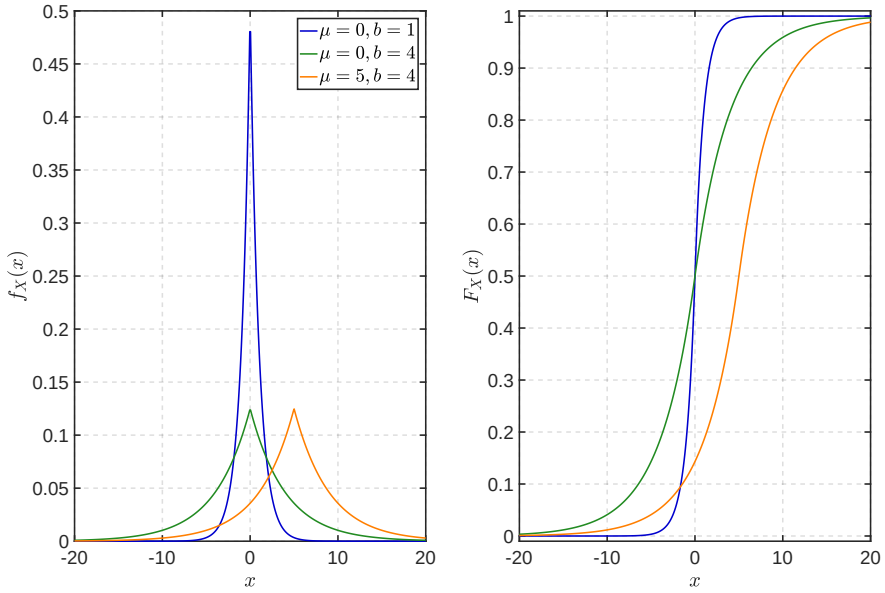

Figure 12: PDF and CDF of Laplace distributions for various parameter values.

1.3 Truncated distributions

A truncated distribution results from restricting the domain of some probability distribution. Assume that a random variable X follows some distribution with CDF $F_X(X)$. Of interest in this section is the derivation of the CDF, F'_X and inverse CDF, F'^{-1}_X of the random variable X after limiting the support to $X \in [a, b]$.
The derivation of these quantities is given below:

$$F'_X(X) = \begin{cases} 0 & , \quad X \leq a \\ \frac{F_X(X) - F_X(a)}{F_X(b) - F_X(a)} & , \quad a < X < b \\ 1 & , \quad X \geq b \end{cases} \tag{1.8}$$

$$F'^{-1}_X(u) = \begin{cases} a & , \quad u = 0 \\ F_X^{-1}\left(F_X(a) + u\left(F_X(b) - F_X(a)\right)\right) & , \quad 0 < u < 1 \\ b & , \quad u = 1 \end{cases} \tag{1.9}$$

A practical example for specifying truncated distributions is given in Section 2.4.

1.4 Representation of random vectors and joint PDFs

1.4.1 Marginals and copula

A natural extension of the univariate random variables case to the multivariate random vectors case is the introduction of multivariate joint-PDFs. Within the UQLAB framework, joint CDFs are represented by means of the copula formalism (Nelsen, 2006). Copulas are a powerful tool to provide a simple representation of multivariate distributions by separating the univariate distributions of each component of a random vector (marginals) from their depen-

dence structure (copula). At the basis of the copula formalism lies Sklar's theorem:

$$F_{\boldsymbol{X}}(\boldsymbol{x}) = C(F_{X_1}(x_1), F_{X_2}(x_2), ..., F_{X_M}(x_M)) \tag{1.10}$$

where $\boldsymbol{X} = \{X_1, \ldots, X_M\}$ is the M-dimensional input random vector with joint CDF $F_{\boldsymbol{X}}(\boldsymbol{x})$, $F_{X_i}(x_i)$ is the i-th input marginal and $C(\cdot)$ is a function that describes the dependence structure between the M input variables. From (1.10) the joint PDF is obtained by differentiation:

$$f_{\boldsymbol{X}(\boldsymbol{x})} = \prod_{i=1}^{M} f_{X_i}(x_i) \cdot c\left(F_{X_1}(x_1), \ldots, F_{X_M}(x_M)\right) \tag{1.11}$$

where $c(\cdot)$ is the copula density function obtained as

$$c(u_1, \ldots, u_M) = \frac{\partial^M C(u_1, \ldots, u_M)}{\partial u_1, \ldots, \partial u_M}. \tag{1.12}$$

Sklar's theorem states that any joint PDF $F_{\boldsymbol{X}}$ can be expressed in terms of its marginal distributions $F_{X_i}(x_i)$ and some additional function, the copula, that represents their dependence. This framework is ideal for uncertainty quantification in scientific applications. Indeed it is often the case that marginal distributions as well as some form of correlation measure between variables are known or inferred from available data, but no clear information about the correlation structure is readily available.

A detailed description of the copula formalism and of the plethora of available copulas and copula families is outside the scope of this manual. The reader is referred to Nelsen (2006) for more details and relevant literature.

1.4.2 Copulas currently available in UQLAB

The UQLAB framework currently supports two different copulas to represent dependence between variables, namely the independent, and Gaussian copulas. They are characterized as follows:

1. **Independent copula:**

$$C(u_1, \ldots, u_M) = \prod_{i=1}^{M} u_i \tag{1.13}$$

By substituting Eq. (1.13) in (1.10), it is clear that it corresponds to the case in which the components of the input random vector are independent. Indeed:

$$F_{\boldsymbol{X}}(\boldsymbol{x}) = C(F_{X_1}(x_1), \ldots, F_{X_M}(x_M)) = \prod_{i=1}^{M} F_{X_i}(x_i), \tag{1.14}$$

consequently the independent copula density is simply $c(u_1, \ldots, u_M) = 1$. A graphical representation of the uniform copula density is given for reference in Figure 13.

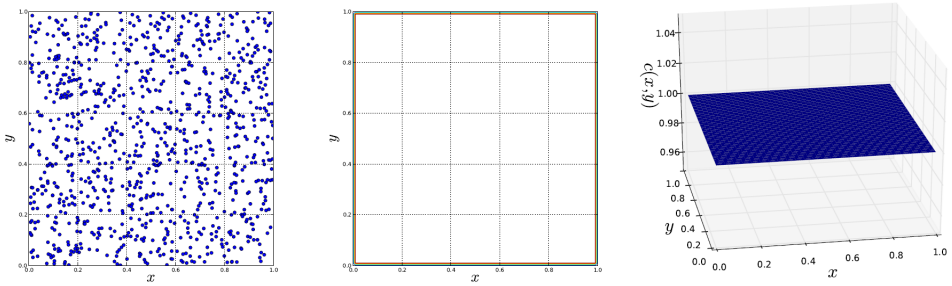

Figure 13: Independent copula: scatterplot (a), contour plot (b) and 3D-view (c) of the copula density $c(u_1, u_2)$.

2. **Gaussian copula:**

$$C(u_1, \ldots, u_M; \mathbf{R}) = \Phi_M \left(\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_M); \mathbf{R}\right) \tag{1.15}$$

where $\mathbf{R}$ is the linear correlation matrix *of the multivariate Gaussian distribution associated with the Gaussian copula*, $\Phi_M(\boldsymbol{u}; \mathbf{R})$ is the cumulative distribution function of an M-variate Gaussian distribution with mean $\mathbf{0}$ and correlation matrix $\mathbf{R}$ and $\Phi^{-1}(u_i)$ is the inverse cumulative distribution function of the standard normal distribution.

The Gaussian copula is one of the most commonly used copulas to parametrize the dependence structure of random vectors with known marginals. If all the marginals as well as the copula are Gaussian, the resulting joint PDF is also a multivariate normal distribution with correlation matrix $\mathbf{R}$. Another important property of the Gaussian copula is that a random vector with Gaussian copula and diagonal correlation matrix $\mathbf{R}$ has independent components.

The Gaussian copula is asymptotically independent in both upper and lower tails. This means that no matter how high the parameter correlation coefficient R_{ij} is, there will be no tail dependence (see Nelsen (2006) for details).

A graphical representation of the Gaussian copula is given for reference in Figure 14.

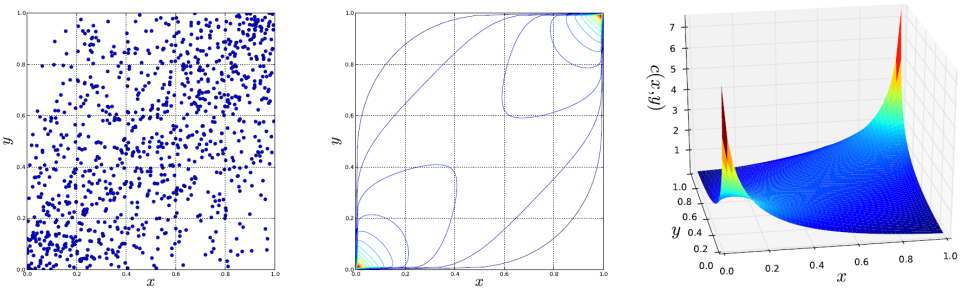

Figure 14: Gaussian copula: scatter plot (a), contour plot (b) and 3D-view (c) of the copula density $c(u_1, u_2)$.

1.5 Sampling random vectors

In most uncertainty quantification applications, sampling the input vectors is as important as properly represent them. However, most of the existing random sampling strategies (e.g. Monte Carlo sampling, latin hypercube sampling (LHS), pseudorandom sequences, etc.) produce samples in the unit hypercube distributed according to $\boldsymbol{X} \sim \mathcal{U}([0,1]^M)$. Amongst the many strategies available to generate samples distributed according to a specific PDF $F_{\boldsymbol{X}}$, UQLAB approaches the problem in terms of *isoprobabilistic transforms*. An isoprobabilistic transform is a map of the form (Lebrun and Dutfoy, 2009):

$$\boldsymbol{X} = \mathcal{T}(\boldsymbol{U}) \qquad s.t. \qquad X \sim F_{\boldsymbol{X}}, \quad \boldsymbol{U} \sim F_{\boldsymbol{U}} \tag{1.16}$$

or, in other words, it is a change of variables that transforms a sample of random vector $\boldsymbol{U} \sim F_{\boldsymbol{U}}$ into a sample of random vector $X \sim F_{\boldsymbol{X}}$. Of particular interest for sampling purposes is the transform between the unit hypercube $\boldsymbol{U} \sim \mathcal{U}([0,1]^M)$, and any other random vector with joint distribution $F_{\boldsymbol{X}}$.
In the following, isoprobabilistic transforms will be derived for the purposes of sampling independent random vectors as well as random vectors with Gaussian copula.

1.5.1 Isoprobabilistic transform of independent marginals

Consider a sampling of size N of the unit hypercube $\mathcal{Z} = \left\{\boldsymbol{u}^{(1)}, \ldots, \boldsymbol{u}^{(N)}\right\} \sim \mathcal{U}([0,1]^M)$. Due to the independence between the components of the random vector $\boldsymbol{X} \sim F_{\boldsymbol{X}}$, a simple isoprobabilistic transform can be used to transform $\mathcal{Z}$ into $\mathcal{X} = \left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\right\} \sim F_{\boldsymbol{X}}$:

$$x_j^{(i)} = F_{X_j}^{-1}(u_j^{(i)}) \tag{1.17}$$

where $F_{X_j}^{-1}$ denotes the inverse CDF of the j-th marginal of the random vector $\boldsymbol{X}$.

1.5.2 Generalized Nataf transform

In the case of dependent variables, several extra steps are needed to properly transform a sample from the unit hypercube to the desired joint PDF. A powerful tool is given by the generalized Nataf transform (Lebrun and Dutfoy, 2009). Given Sklar's theorem in Eq. (1.10), it follows that that a sample $\mathcal{Z}$ from a random vector $\boldsymbol{X} \sim F_{\boldsymbol{X}}(\boldsymbol{x})$ can be obtained from a sample of the underlying copula $\mathcal{C} = \left\{\boldsymbol{c}^{(1)}, \ldots, \boldsymbol{c}^{(N)}\right\}$ with a component-by-component isoprobabilistic transform similar to that in Eq. (1.17):

$$x_j^{(i)} = F_{X_j}^{-1}(c_j^{(i)}) \tag{1.18}$$

where $F_{X_j}^{-1}$ denotes the inverse CDF of the j-th marginal of the random vector $\boldsymbol{X}$. Moreover, the underlying copula distribution is a multivariate Gaussian. An important property of the Gaussian copula is that it is *elliptical*, which means that an isoprobabilistic transform exist that maps samples from an elliptical copula with correlation matrix $\mathbf{R}$ to the same copula

with identity correlation matrix $\boldsymbol{I}$ (uncorrelated components). Such transform has the form:

$$\boldsymbol{U} = \boldsymbol{\Gamma}^{-1}\boldsymbol{V} \tag{1.19}$$

where $\boldsymbol{U}$ is a sample from the elliptical copula with identity correlation matrix, $\boldsymbol{V}$ is a sample from the same copula but with correlation matrix $\mathbf{R}$ and $\boldsymbol{\Gamma}^{-1}$ is the inverse Cholesky factor of $\mathbf{R} = \boldsymbol{\Gamma}\boldsymbol{\Gamma}^{\mathsf{T}}$. Note that this property is well known for multivariate Gaussian distributions. With these ingredients it is possible to define the so-called *generalized Nataf transform*:

$$\boldsymbol{U} = \mathcal{T}^{GN}(\boldsymbol{X}), \tag{1.20}$$

where $\boldsymbol{U} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ (standard normal space) for the Gaussian copula, as follows:

1. $\boldsymbol{X} \mapsto \boldsymbol{W} = [F_{X_1}(X_1), \ldots, F_{X_M}(X_M)]^T$
2. $\boldsymbol{W} \mapsto \boldsymbol{V} = \left[E^{-1}(W_1), ..., E^{-1}(W_M)\right]^T$
3. $\boldsymbol{V} \mapsto \boldsymbol{U} = \boldsymbol{\Gamma}^{-1}\,\boldsymbol{V}$

where $\boldsymbol{W}$ is a sample of the copula with correlation matrix $\mathbf{R}$, $\boldsymbol{V}$ is a sample of the multivariate elliptical distribution with correlation matrix $\mathbf{R}$ that generates the copula , $E^{-1} = \Phi^{-1}$ for the Gaussian copula and $\boldsymbol{\Gamma}^{-1}$ is the inverse Cholesky factor of $\mathbf{R}$. The Nataf transform first transforms any sample of the input random vector into a sample of the underlying copula. It then transforms the copula into the desired multivariate standard elliptical distribution with correlation matrix $\mathbf{R}$ and finally decorrelates it via Eq. (1.19).

1.5.3 Sampling multivariate distributions with the inverse generalized Nataf transform

An important property of the transform in Section 1.5.2 is that it is invertible. It then becomes clear how it can be used for efficient sampling of multivariate distributions in conjunction with random number generators that generate samples in the uniform unit hypercube $\boldsymbol{Z} = \mathcal{U}([0,1]^M)$. Given a sample of the uniform unit hypercube $\mathcal{Z} = \left\{\boldsymbol{z^{(1)}}, \ldots, \boldsymbol{z^{(N)}}\right\}$, one can obtain a sample from the desired random vector $\boldsymbol{X} \sim F_{\boldsymbol{X}}(\boldsymbol{x})$ with arbitrary marginals $F_{X_1}(x_1), \ldots, F_{X_M}(x_M)$ and elliptic copula $C(\boldsymbol{U})$ as follows:

1. $\boldsymbol{Z} \mapsto \boldsymbol{U} \sim F_E(\boldsymbol{u})$ [Generate samples from the standard elliptical distribution]
2. $\boldsymbol{U} \mapsto \boldsymbol{V} = \boldsymbol{\Gamma}^{-1}\,\boldsymbol{U}$ [Correlate the samples]
3. $\boldsymbol{V} \mapsto \boldsymbol{W} = [E(V_1), ..., E(V_M)]^T$ [Transform into a sample of the underlying copula]
4. $\boldsymbol{W} \mapsto \boldsymbol{X} = \left[F_{X_1}^{-1}(X_1), \ldots, F_{X_M}^{-1}(X_M)\right]^T$ [Transform into $F_{\boldsymbol{X}}(\boldsymbol{x})$]

The implementation of the first step of this algorithm depends on the actual copula chosen. In case of Gaussian copula it can be easily achieved directly with Eq. (1.17).

Chapter 2

Usage

2.1 Drawing samples from a distribution

2.1.1 Introductory example

Let us consider a Gaussian vector $\boldsymbol{X} = [X_1, X_2]^T$ with independent components. The mean value and standard deviation of X_1 (resp. X_2) are $\mu_1 = 1$, $\sigma_1 = 1$ (resp. $\mu_2 = 2$, $\sigma_2 = 0.5$). In UQLAB an INPUT object is created by defining the list of marginal distributions and the copula that connects these marginals to form the joint distribution.

```
uqlab;
Input.Marginals(1).Type = 'Gaussian';
Input.Marginals(1).Parameters = [1 1];
Input.Marginals(2).Type = 'Gaussian';
Input.Marginals(2).Moments = [2 0.5];
myInput1 = uq_createInput(Input);
```

Note that each marginal distribution can be defined either from its parameters or moments (*but not from both*). By default the input variables are assumed independent in UQLAB. In this case there is no need to define a copula. This can be however done explicitly by typing (before using `uq_createInput`):

```
Input.Copula.Type = 'Independent';
```

When the INPUT object `myInput1` is created, all the parameters and moments for all variables are computed and all possible inconsistencies are checked. In case the parameters and moments are specified for a given marginal. an error is returned and the script aborts.
Once the INPUT object `myInput1` has been created, a report can be produced as follows:

```
uq_print(myInput1)
-----------------------------------------
Input object name: Input 1
Dimension(M): 2

Marginals:
Index | Name | Type     | Parameters           | Moments
=========================================================================
1     | X1   | Gaussian | 1.000e+00, 1.000e+00 | 1.000e+00, 1.000e+00
2     | X2   | Gaussian | 2.000e+00, 5.000e-01 | 2.000e+00, 5.000e-01
```

```
Copula:
Type: Independent
-----------------------------------------
```

For visual inspecting an INPUT object the function `uq_display` can be used as follows:

```
uq_display(myInput1)
```

The result is shown in Figure 15.

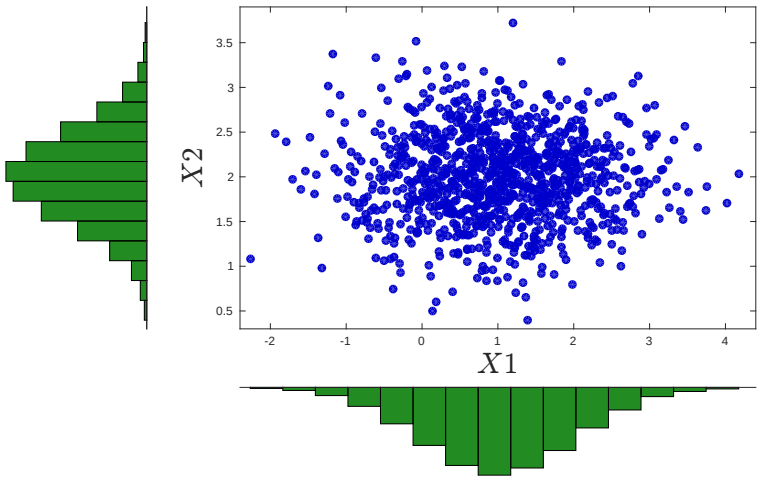

Figure 15: The output of the function `uq_display` on the INPUT object `myInput1`.

It is possible to draw samples from the INPUT object `myInput1` as follows:

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

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`. Methods to handle different INPUT objects in the workspace are described in Section 2.5.
The MATLAB standard random number generator is used by default. The result is shown in Figure 16.

2.1.2 Special cases of distributions

Most of the available (built-in) distributions can be defined similarly to the way a Gaussian distribution was defined in Section 2.1.1, *i.e.*either by defining the two parameters of the distribution or its moments (mean and standand deviation). The meaning of the parameters is as described in Section 1.2. Some special cases are the following:

- For *exponential* distribution only one parameter (λ) exists. When such distribution is defined by its parameters only one element is needed (λ). Additionally when it is

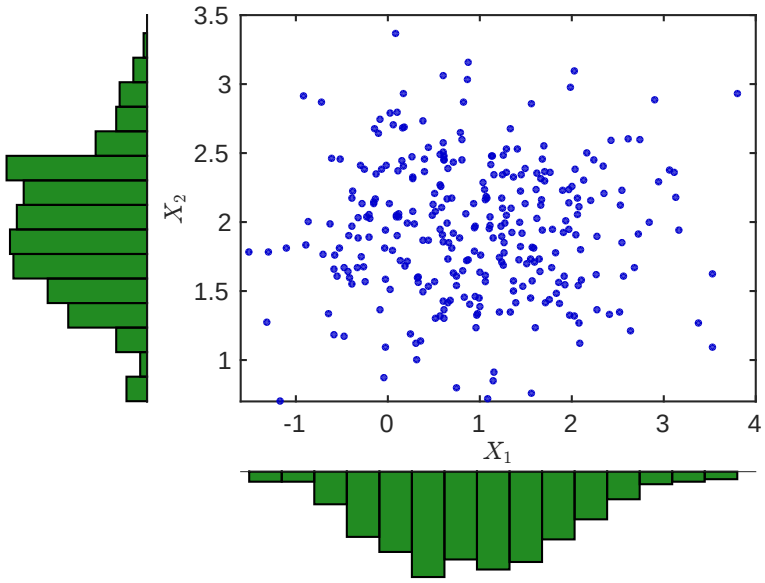

Figure 16: Samples drawn from a 2-D Gaussian distribution with independent marginals. The sampling method is plain Monte Carlo.

defined by its moments only one element is needed again which corresponds to the mean and standard deviation (that are equal).

- For *beta* distribution there is the possibility of using four parameters when a custom support $[a, b]$ needs to be defined. For example, in order to define an element of an input vector that follows a beta distribution with parameters $[r, s] = [1, 2]$ and support $[a, b] = [0.5, 1.5]$ we do the following:

```
Input.Marginals.Type = 'Beta';
Input.Marginals.Parameters = [1, 2, 0.5, 1.5];
```

Similarly, we can define a beta distribution with moments $[\mu, \sigma] = [0.8, 0.2]$ and support $[a, b] = [0.5, 1.5]$ as follows:

```
Input.Marginals.Type = 'Beta';
Input.Marginals.Moments = [0.8, 0.2, 0.5, 1.5];
```

In this case the two parameters r, s are computed according to the equations in Section 1.2.9.

2.1.3 Using a copula

Let us now create another INPUT object where dependency between the marginals is introduced by imposing a Gaussian copula. As discussed in Section 1.4.2, the Gaussian copula takes as parameter the linear correlation matrix $\mathbf{R}$ of the copula, which is assumed to be

equal to $\mathbf{R} = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$ (linear correlation coefficient $\rho_{12} = 0.8$).

```
Input.Name = 'Input 2: Dependent marginals' ;
Input.Copula.Type = 'Gaussian';
Input.Copula.Parameters = [   1   0.8  ; 0.8   1 ];
myInput2 = uq_createInput(Input);
X = uq_getSample(300);
```

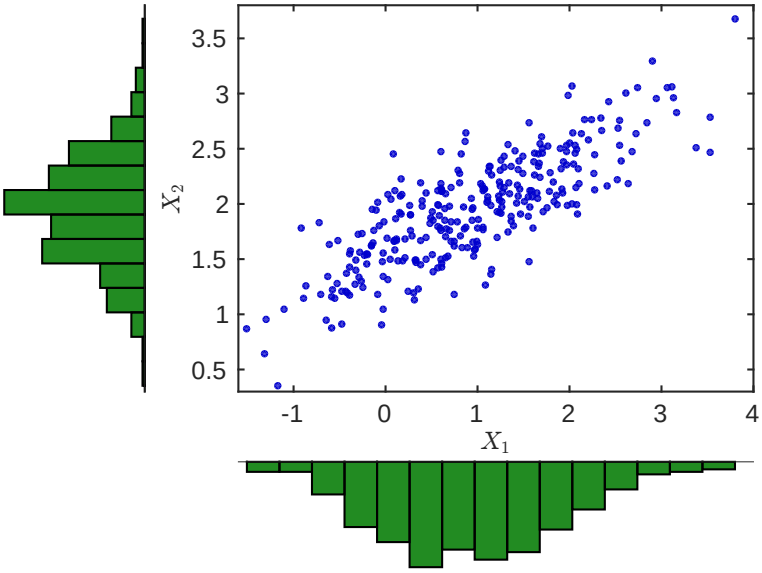

Figure 17: Samples drawn from a 2-D Gaussian distribution with a Gaussian copula (correlation coefficient $\rho_{12} = 0.8$).The sampling method is plain Monte Carlo.

The resulting samples are plotted in Figure 17. By default the sampling method that is used is Monte Carlo. Also notice that we can assign custom names to an INPUT object, *e.g.* `'Input 2: Dependent marginals'` in the present case.

2.1.4 Selecting an INPUT object and specifying the sampling method

When handling several INPUT objects in parallel, the last one that has been defined is used by default for sampling. There are two ways to use a different INPUT object than the last for sampling:

- Using the function `uq_selectInput`. For example, drawing 300 samples from the INPUT object `myInput1` can be achieved as follows:

```
uq_selectInput(myInput1);
X = uq_getSample(300);
```

- Specifying the INPUT object directly in `uq_getSample`. For example, drawing 300 samples from the INPUT object `myInput1` can be achieved as follows:

```
X = uq_getSample(myInput1, 300);
```

When using `uq_getSample` in order to obtain samples from a random vector, various sampling methods can be used. For instance Latin Hypercube Sampling (McKay et al., 1979) can be used to sample from the INPUT object `myInput2` as follows:

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

The result is shown in Figure 18. For a list of all the available sampling methods refer to Table 6 in Section 3.2.

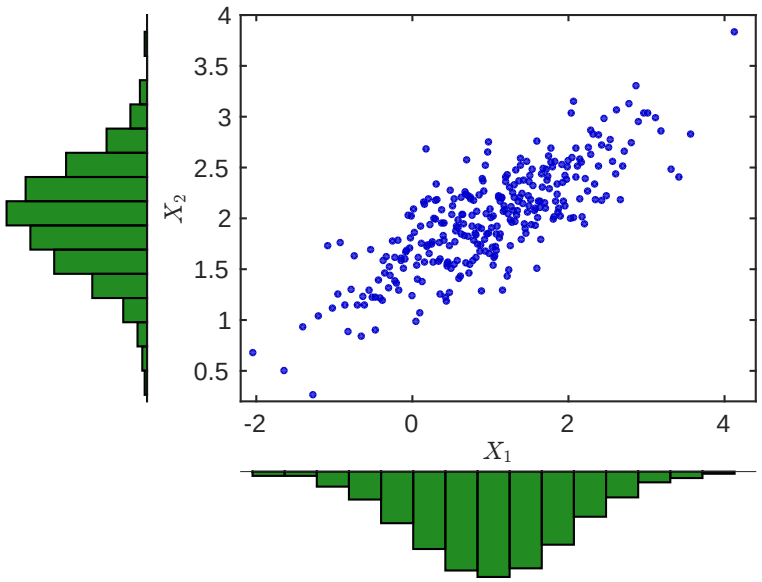

Figure 18: Samples drawn from a 2-D Gaussian distribution with a Gaussian copula. The Latin Hypercube sampling method is used.

Advanced options

Instead of using the `'LHS'` argument in `uq_getSample` one could also change the default sampling method of the INPUT object `myInput2` as follows:

```
uq_selectInput(myInput2);
uq_setDefaultSampling('LHS');
```

Note that this does not affect other INPUT objects, *i.e.*the sampling method for `myInput1` is still plain Monte Carlo.

2.2 Enrichment of an experimental design with new samples

Enrichment is a functionality that can be used when there is an already existing sample set (called *experimental design* in the context of metamodelling) to which more points need to be added. Starting from where the previous section (Figure 18) left off, assume that we want to add 700 additional points to the Latin Hypercube (the already existing 300 samples are stored in `X`):

```
Xnew = uq_enrichLHS(X, 700);
```

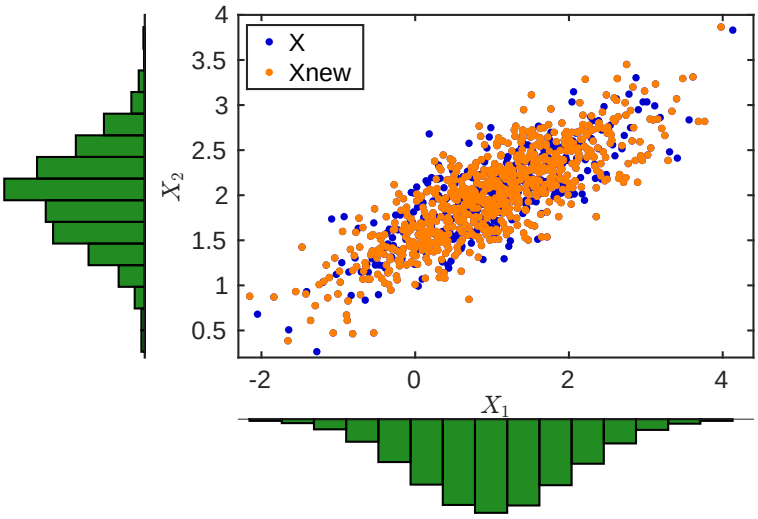

Figure 19: Enrichment of the LHS samples drawn from a 2-D Gaussian distribution with a Gaussian copula using `uq_enrichLHS`. The initial sample set is shown in Figure 18.

Note: `uq_enrichLHS` only returns the new sample points! The full set can be retrieved by `Xfull = [X;Xnew]`.

However, `uq_enrichLHS` should *only* be used when the initial sample set is generated by Latin Hypercube sampling. If the initial sample set `X` was generated, *e.g.* using plain Monte Carlo or if its origin is unknown (*e.g.* because it has been produced by another software) then the function `uq_LHSify` must be used. This function enriches the existing sample set in such a way that it forms a pseudo-Latin Hypercube sampling as a whole. This can be done as follows:

```
Xnew = uq_LHSify(X, 700);
Xfull = [X;Xnew];
```

For enriching an experimental design that was generated by Sobol or Halton sampling the functions `uq_enrichSobol` and `uq_enrichHalton` can be used with a similar syntax. For more information about the available sample enrichment functions see Section 3.4.

2.3 Performing an isoprobabilistic transform

Isoprobabilistic transforms are implemented in a general fashion in UQLAB. Assume that we want to map some existing sample `X` to the standard normal space (*e.g.* for solving a reliability problem, see Section 1 of UQLAB User Manual – Structural reliability (Rare event estimation)).

This can be carried out by defining the marginals (here standard normal) and copula (here independent) of the target vector U.

```
UMarginals(1).Type = 'Gaussian';
UMarginals(1).Parameters = [0,1];
UMarginals(2).Type = 'Gaussian';
UMarginals(2).Parameters = [0,1];
UCopula.Type = 'Independent';
```

Then the transformed samples are obtained by:

```
U = uq_GeneralIsopTransform(X,...
    myInput2.Marginals, myInput2.Copula, UMarginals, UCopula);
```

The original and transformed samples are shown in Figure 20.

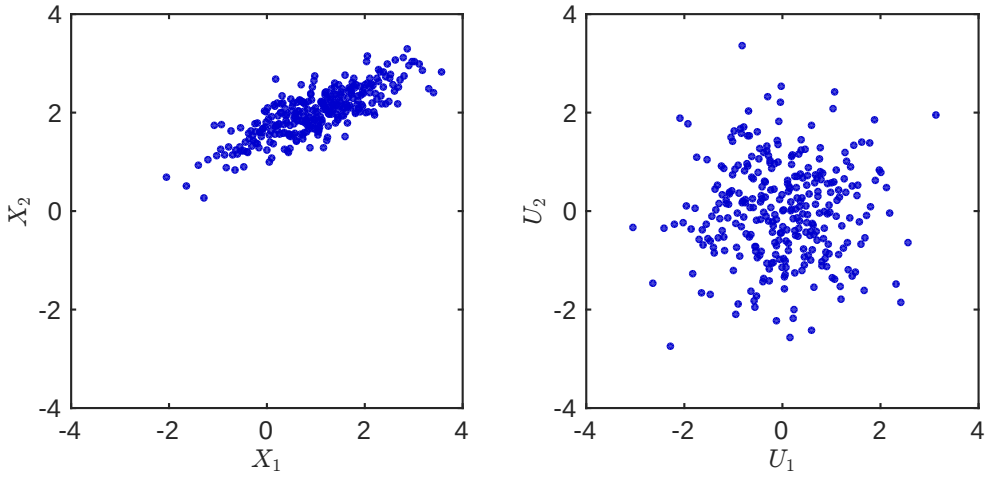

Figure 20: An isoprobabilistic transform from some physical space (2D Gaussian) to the standard normal space.

2.4 Adding bounds

Suppose we want to draw samples from an uncorrelated 2D Gaussian distribution with X_1 (resp. X_2) having mean value $\mu_1 = 1$ (resp. $\mu_2 = 3$) and standard deviation $\sigma_1 = 2$ (resp. $\sigma_2 = 2$). Additionally X_1 is bounded in $[-3, 2]$:

```
Input.Marginals(1).Type = 'Gaussian';
Input.Marginals(1).Parameters = [1 2];
Input.Marginals(1).Bounds = [-3 2];
Input.Marginals(2).Type = 'Gaussian';
```

```
Input.Marginals(2).Moments = [2 2];
myInput3 = uq_createInput(Input);
```

Now we can draw 300 samples using Latin Hypercube Sampling as follows:

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

The result is shown in Figure 21.

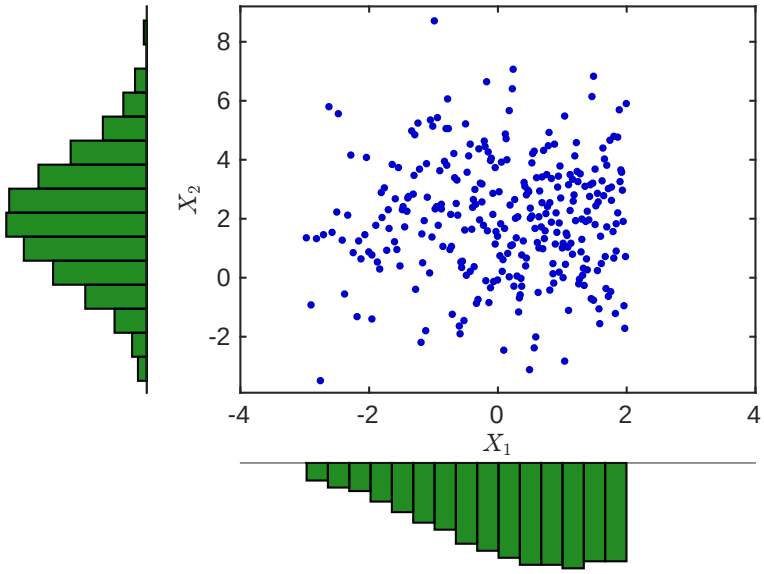

Figure 21: Samples drawn from a 2-D Gaussian distribution with independent marginals and bounds on $X_1 \in [-3, 2]$. The sampling method is Latin Hypercube sampling.

2.5 Switching between input objects

Suppose we want to draw two sample sets:

- 200 Monte Carlo samples from the input vector `myInput2` with correlated components.
- 300 LHS samples from the input vector `myInput3` with bounded marginal X_1.

This is carried out as follows:

```
uq_selectInput(myInput2);
X2 = uq_getSample(200,'MC');
uq_selectInput(myInput3);
X3 = uq_getSample(300,'LHS');
```

The two samples are plotted in Figure 22.

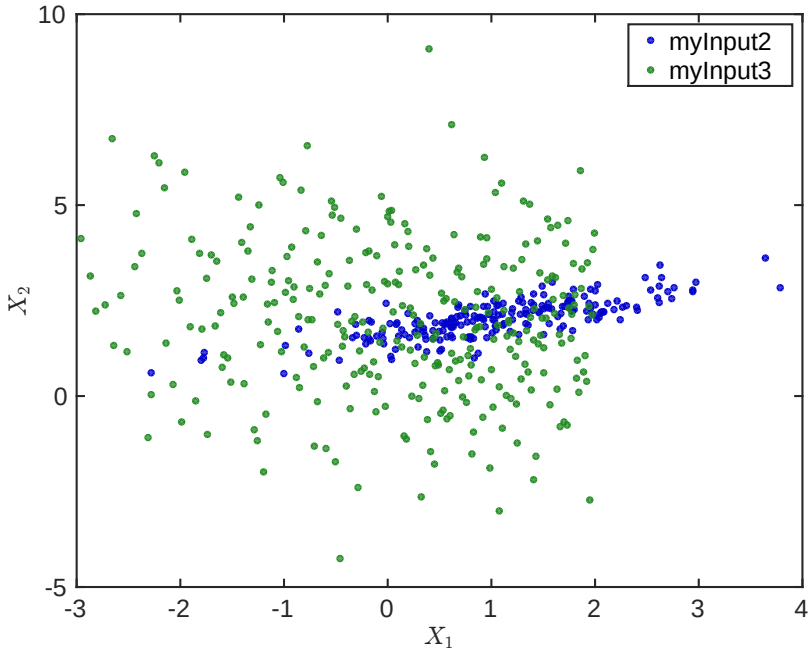

Figure 22: Samples drawn from two different cases of a 2-D Gaussian distribution. The INPUT object `myInput2` has dependent marginals and the INPUT object `myInput3` has independent marginals and bounds on $X_1 \in [-3, 2]$. The sampling method is Latin Hypercube sampling.

2.6 Defining and using custom marginals

The built-in probability distributions can be found in Table 2 of the Reference List (Section 3) and a brief description of each in Section 1.2. New distributions can be defined by providing three files to a folder that is available to the MATLAB path, each corresponding to the PDF, CDF and inverse CDF respectively. In order to define a new distribution some naming convention has to be followed. The functions that correspond *e.g.*to a distribution called `myDistribution` should be named as follows:

- PDF function: `uq_myDistribution_pdf`
- CDF function: `uq_myDistribution_cdf`
- inverse CDF function: `uq_myDistribution_invcdf`

The definition of each function should look like:

```
function f = uq_myDistribution_pdf(X, parameters)

function F = uq_myDistribution_cdf(X, parameters)

function X = uq_myDistribution_invcdf(F, parameters)
```

In these functions, `X` and `F` are vectors of length N and `parameters` is a vector of doubles of arbitrary length.

In order to use the custom distribution `myDistribution` we then create an INPUT object as follows:

```
Input.Marginals(1).Type = 'myDistribution';
Input.Marginals(1).Parameters = ...
%etc
myInput = uq_createInput(Input);
```

2.6.1 Advanced options

In order to fully specify a custom probability distribution a function that calculates the mean and standard deviation of the distribution given its parameters is required. The following naming convention should be used in order to specify such function:

```
function moments = uq_myDistribution_PtoM(parameters)
```

where `parameters` is an arbitrary variable that contains the distribution's parameters and `moments` is a vector equal to $[\mu, \sigma]$, that is the mean and standard deviation of the distribution for the given parameter values.

Note: In case the function `uq_myDistribution_PtoM` is not provided the moments of the distribution are estimated numerically. For more information refer to the `uq_estimateMoments` function reference (Section 3.7.3). The default values of the options of `uq_estimateMoments` are used for estimating the moments.

In addition, the function that calculates the parameters of a user-defined probability distribution given its moments can be specified in case the ability of specifying the distribution based on its mean and standard deviation is required. In that case a function of the following form needs to be created:

```
function parameters = uq_myDistribution_MtoP(moments)
```

where `moments` is a vector equal to $[\mu, \sigma]$, that is the mean and standard deviation of the distribution and `parameters` is a vector that contains the distribution's parameters that correspond to the given moments.

2.7 Constant variables

Constants form a special variable type that simply corresponds to some constant scalar value. A constant variable can be specified as follows:

```
Input.Marginals.Type = 'Constant';
Input.Marginals.Parameters = 1;
```

In addition, a random variable may be reverted to constant in case its variance is zero. For example, after creating the following INPUT object:

```
Input.Marginals(1).Type = 'Gaussian';
```

```
Input.Marginals(1).Parameters = [0 1];
Input.Marginals(2).Type = 'Gaussian';
Input.Marginals(2).Parameters = [1 0];
myInput = uq_createInput(Input);
```

The following warning message is returned:

```
Warning: Marginal(2).Type changed from Gaussian to constant because
the variance was zero.
```

Similarly a random variable is reverted to constant in case bounds have been specified with upper and lower bound being identical. For example, after creating the following INPUT object:

```
Input.Marginals(1).Type = 'Gaussian';
Input.Marginals(1).Parameters = [0 1];
Input.Marginals(2).Type = 'Gaussian';
Input.Marginals(2).Parameters = [0 1];
Input.Marginals(2).Bounds = [0 0];
myInput = uq_createInput(Input);
```

The following warning message is returned:

```
Warning: Marginal(2).Type changed from Gaussian to constant because
the upper and lower bounds were identical.
```

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 group configuration options and output quantities semantically. Due to the complexity of the algorithms implemented, it is not uncommon to employ nested structures to fine-tune inputs/outputs. Throughout this reference guide, we adopt a table-based description of the configuration structures.
The simplest case is given when a field of the structure is a simple value/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 correspond to name, 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, we provide a link to a table that describes the available options, 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/doubles` mean a real number represented in double precision (resp. a set of such real numbers).

3.1 Creating an INPUT object: `uq_createInput`

Syntax

```
myInput = uq_createInput(Input)
```

Input

The `struct` variable `Input` contains the description of the marginal distributions of the input parameters as well as their dependence through the copula function.
The content of the `Input` structure is listed in Table 1.

Table 1: `Input`

●	`.Marginals`	Table 2	The options regarding the marginals of the random vector
☐	`.Copula`	Table 3	The options regarding the copula of the random vector
☐	`.Name`	String	The name of the object. If not set by the user, a unique string is automatically assigned to it, *e.g.* `'Input 1'`.

The options that can be defined under `Input.Marginals` are given in Table 2.

Table 2: `Input.Marginals`

●	`.Type`	String	Type of marginal distribution
		`'Constant'`	A constant value
		`'Gaussian'`	Gaussian distribution (Section 1.2.2)
		`'Lognormal'`	Lognormal distribution (Section 1.2.3)
		`'Uniform'`	Uniform distribution (Section 1.2.1)
		`'Exponential'`	Exponential distribution (Section 1.2.8)
		`'Beta'`	Beta distribution (Section 1.2.9)
		`'Weibull'`	Weibull distribution (Section 1.2.6)
		`'Gumbel'`	Gumbel maximum extreme value distribution (Section 1.2.4)
		`'GumbelMin'`	Gumbel minimum extreme value distribution (Section 1.2.5)
		`'Gamma'`	Gamma distribution (Section 1.2.7)
		`'Triangular'`	Triangular distribution (Section 1.2.10)

		`'Logistic'`	Logistic distribution (Section 1.2.11)
		`'Laplace'`	Laplace distribution (Section 1.2.12)
		other String	A user-defined marginal type (Section 2.6)
⊕	`.Moments`	variable length Double	• Mean and standard deviation($[\mu, \sigma]$) of the marginal distribution • In case of constants just the constant value is needed. • For Beta distribution $[\mu, \sigma, a, b]$ can be used for defining a custom support, otherwise it is assumed that $[a, b] = [0, 1]$ • For exponential distribution a single element in `.Moments` is needed (both mean and standard deviation)
⊕	`.Parameters`	variable length Double	• The parameters of the marginal distribution as defined in Section 1.2 • In case of constants just the constant value is needed. • For Beta distribution either the parameters $[r, s]$ can be set (then the support is assumed to be $[a, b] = [0, 1]$) or all $[r, s, a, b]$ can be set for defining a custom support $[a, b]$.
□	`.Bounds`	1×2 Double default: `[-inf, inf]`	$[X_{min}, X_{max}]$ admissible value

Note: Only one of the two fields `Input.Marginals.Parameters` or `Input.Marginals.Moments` can be set by the user for each element of `Input.Marginals`. If both fields are specified, an error is returned.

The options that can be defined under `Input.Copula` are listed in Table 3.

Table 3: `Input.Copula`

●	`.Type`	String default: `'Independent'`	Copula type.
		`'Independent'`	Independent copula.
		`'Gaussian`	Gaussian copula

⊞	`.Parameters`	$M \times M$ Double	• For a Gaussian copula it corresponds to the linear correlation matrix • It is not taken into account when `Input.Copula.Type` has value `'Independent'`
⊞	`.RankCorr`	$M \times M$ Double	Spearman correlation matrix (Gaussian copula only)

Output

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

Table 4: `myInput = uq_createInput(...)`		
`.Name`	String	The name of the INPUT object
`.Sampling`	Struct	Information regarding the default and lastly used sampling method. See `Sampling` for contents
`.Marginals`	$M \times 1$ Struct	Information regarding the marginals, see Table 2 for contents
`.Copula`	Struct	Information regarding the copula, see Table 3 for contents
`.Internal`	Struct	Internal fields that are only of interest for scientific developers. For more information refer to the Scientific Developer's Guide of the Input module

The structure `Sampling` contains the following fields :

Table 5: `myInput.Sampling`		
`.DefaultMethod`	String Table 6	The default sampling method
`.Method`	String Table 6	The current sampling method

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, U] = uq_getSample(...)
```

Description

`X = uq_getSample(N)` returns N samples of a random vector defined in the currently selected INPUT module using the default sampling method. If not set otherwise by the user, the default sampling method is `'MC'` (Monte Carlo). The user can call the function `uq_setDefaultSampling` to change the default sampling method.

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

`X = uq_getSample( myInput,N,method)` returns N samples of a random 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 6: `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 7.

Table 7: Available options of `uq_getSample`

Name	Value	Description
	`method = 'LHS'`	
`'iterations'`	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

`X = uq_getSample(...)` returns an N-by-M matrix of the samples of the selected random vector in the physical space (where M is the dimension of the random vector).

`[X, U] = uq_getSample(...)` additionally returns an N-by-M matrix (`U`) of the samples in the uniform space.

3.3 Printing/Visualizing an INPUT object

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

3.3.1 Printing information: `uq_print`

Syntax

```
uq_print(myInput);
```

Description

`uq_print(myInput)` prints a report about the INPUT object `myInput` (type, name, parameters and moments of each marginal and brief information about the copula).

3.3.2 Graphical visualization: `uq_display`

Syntax

```
uq_display(myInput);
uq_display(myInput, idx);
uq_display(myInput, idx, 'marginals');
```

Description

`uq_display(myInput)` produces a sample set from the INPUT object `myInput`. Then it produces $M \times M$ plots in a single figure. Each plot, indexed by $i, j = 1, \ldots, M$ corresponds to the scatter plot between the i-th and j-th element of the random vector. The diagonal elements (*i.e.*when $i = j$) contain the histogram of the corresponding element of the random vector. If $M = 1$ it produces plots of the PDF and CDF of the random variable instead.

`uq_display(myInput, idx)` only takes into account the input indices that are contained in the vector `idx`. In particular, if `idx` is a single integer, this command produces plots of the PDF and CDF of that particular component.

`uq_display(myInput, idx, 'marginals')` only takes into account the input indices that are contained in the vector `idx` and regardless of its size, it produces plots of the PDF and CDF of each component.

Examples

`uq_display(myInput, [1 3])` will display the plots only for the first and third element of the random vector that is described by `myInput`.

3.4 Enriching an existing sample set

3.4.1 Enriching a Latin Hypercube: `uq_enrichLHS`

Syntax

```
X1 = uq_enrichLHS(X0, N)
X1 = uq_enrichLHS(X0, N, myInput)
[X1,  U1] = uq_enrichLHS(...)
```

Input

`X1 = uq_enrichLHS(X0, N)` enriches the experimental design `X0` defined in the currently selected INPUT object with `N` new samples so that the enriched sample set forms a (pseudo-) Latin Hypercube sampling.

> **Note:** The initial sample set is expected to form a Latin Hypercube, i.e. it should be generated using Latin Hypercube sampling! If that is not the case, the function `uq_LHSify` should be used instead (see Section 3.4.4).

`X1 = uq_enrichLHS(X0, N, myInput)` enriches the experimental design `X0` defined in the INPUT object `myInput` with `N` new samples so that the enriched sample set forms a Latin Hypercube.

Output

`X1 = uq_enrichLHS(...)` returns an N-by-M matrix of the *new* samples of the selected random vector in the physical space.

> **Note:** `X1` only contains the new samples in the physical space. The enriched sample set is obtained by `X = [X0;X1]`.

`[X1, U1] = uq_enrichLHS(...)` additionally returns an N-by-M matrix (`U1`) of the *new* samples in the uniform space.

3.4.2 Enriching a Sobol sequence: `uq_enrichSobol`

Syntax

```
X1 = uq_enrichSobol(X0, N)
X1 = uq_enrichSobol(X0, N, myInput)
[X1,  U1] = uq_enrichSobol(...)
```

Input

`X1 = uq_enrichSobol(X0, N)` enriches the experimental design `X0` defined in the currently selected `INPUT` object with `N` new samples that correspond to the next elements of the Sobol sequence.

> **Note:** The initial sample set is expected to be generated using Sobol sampling.

`X1 = uq_enrichSobol(X0, N, myInput)` enriches the experimental design `X0` defined in the `INPUT` object `myInput` with `N` new samples that correspond to the next elements of the Sobol sequence.

Output

`X1 = uq_enrichSobol(...)` returns an N-by-M matrix of the *new* samples of the selected random vector in the physical space.

> **Note:** `X1` only contains the new samples in the physical space. The enriched sample set is obtained by `X = [X0;X1]`.

`[X1, U1] = uq_enrichSobol(...)` additionally returns an N-by-M matrix (`U1`) of the *new* samples in the uniform space.

3.4.3 Enriching a Halton sequence: `uq_enrichHalton`

Syntax

```
X1 = uq_enrichHalton(X0, N)
X1 = uq_enrichHalton(X0, N, myInput)
[X1,  U1] = uq_enrichHalton(...)
```

Input

`X1 = uq_enrichHalton(X0, N)` enriches the experimental design `X0` defined in the currently selected `INPUT` object with `N` new samples that correspond to the next elements of the Halton sequence.

> **Note:** The initial sample set is expected to be generated using Halton sampling.

`X1 = uq_enrichHalton(X0, N, myInput)` enriches the experimental design `X0` defined in the `INPUT` object `myInput` with `N` new samples that correspond to the next elements of the Halton sequence.

Output

`X1 = uq_enrichHalton(...)` returns an N-by-M matrix of the *new* samples of the selected random vector in the physical space.

> **Note:** `X1` only contains the new samples in the physical space. The enriched sample set is obtained by `X = [X0;X1]`.

`[X1, U1] = uq_enrichHalton(...)` additionally returns an N-by-M matrix (`U1`) of the *new* samples in the uniform space.

3.4.4 Pseudo-LHS enrichment: `uq_LHSify`

Syntax

```
X1 = uq_LHSify(X0, N)
X1 = uq_LHSify(X0, N, myInput)
[X1,  U1] = uq_LHSify(...)
```

Input

`X1 = uq_LHSify(X0, N)` enriches the experimental design `X0` defined in the currently selected `INPUT` object with `N` new samples so that the enriched sample set forms a pseudo-Latin Hypercube.
`X1 = uq_LHSify(X0, N, myInput)` enriches the experimental design `X0` defined in the `INPUT` object `myInput` with `N` new samples so that the enriched sample set forms a pseudo-Latin Hypercube.

Output

`X1 = uq_LHSify(...)` returns an N-by-M matrix of the *new* samples of the selected random vector in the physical space.

> **Note:** `X1` only contains the new samples in the physical space. The enriched sample set is obtained by `X = [X0;X1]`.

`[X1, U1] = uq_LHSify(...)` additionally returns an N-by-M matrix (`U1`) of the *new* samples in the uniform space.

3.5 Sub-sampling an existing sample set: `uq_subsample`

Syntax

```
X1 = uq_subsample(X0, N1, method)
X1 = uq_subsample(X0, N1, method, Name, Value)
X1 = uq_subsample(...)
[X1,  idx] = uq_subsample(...)
```

Input

`X1 = uq_subsample(X0, N1, method)` reduces the sample size of the experimental design `X0` (`N0`-by-M matrix) from `N0` to `N1` samples using the approach specified in `method`. The following values are possible for `method`:

- `'random'`: The subset of `N1` samples out of `N0` is selected randomly
- `'k-means'`: The subset of `N1` samples out of `N0` is selected by first performing k-means clustering with $k =$ `N1`. Subsequently, the `N1` samples closest to the cluster centroids are selected.

`X1 = uq_subsample(X0, N1, method, Name, Value)` allows for fine-tuning various parameters of the subsampling algorithm by specifying `Name` and `Value` pairs of options. The available options are summarised in Table 8.

Table 8: Available options of `uq_subsample(..., Name, Value)`

`Name`	`Value`	Description
The following options are taken into account when `method = 'kmeans'`		
`'Distance_kmeans'`	String default: `'sqeuclidean'`	The distance measure that is used in k-means clustering. The available options can be found in the documentation of the built-in MATLAB function `kmeans` (option `'Distance'`).
`'Distance_nn'`	String default: `'euclidean'`	The distance measure that is used in nearest neighbour search for determining the samples closest to the k-means centroids. The available options can be found in the documentation of the built-in MATLAB function `knnsearch` (option `'Distance'`).

Output

`X1 = uq_subsample(...)` returns an `N1`-by-M matrix that contains a subset of the samples in `X0`.
`[X1, idx] = uq_subsample(...)` additionally returns the indices of the selected samples.

3.6 Transforming samples between spaces

3.6.1 uq_GeneralIsopTransform

Syntax

```
Y = uq_GeneralIsopTransform(X, XMarginals, XCopula, YMarginals, ...
                            YCopula)
```

Input

The `uq_GeneralIsopTransform` function allows one to transform a sample set `X` (of size $N \times M$) drawn from a random vector defined by `XMarginals` and `XCopula` into samples of a random vector `Y` defined by `YMarginals` and `YCopula`.
The guidelines for specifying the structures `XMarginals`, `XCopula`, `YMarginals` and `YCopula` can be found in Section 3.1 (the syntax of structures `Input.Marginals` from Table 2 and `Input.Copula` from Table 3 are used respectively).

Output

`Y = uq_GeneralIsopTransform(...)` returns an N-by-M matrix `Y` that contains the transformed sample set.

3.6.2 uq_IsopTransform

Syntax

```
Y = uq_IsopTransform(X, XMarginals, YMarginals)
```

Input

The `uq_IsopTransform` function allows one to transform a sample set `X` (of size $N \times M$) drawn from a random vector with marginals `XMarginals` into samples of a random vector `Y` with marginals `YMarginals` assuming that the components are independent. The guidelines for specifying the structures `XMarginals` and `YMarginals` can be found in Table 2 (Section 3.1).

Note: The `.Type` and `.Parameters` fields of `X_marginals`, `Y_marginals` are necessary. In case the moments are given for some marginals the function `uq_MarginalFields` can be executed first in order to obtain the corresponding parameter values.

Output

`Y = uq_IsopTransform(...)` returns an N-by-M matrix `Y` that contains the transformed sample set.

3.6.3 uq_NatafTransform

Syntax

```
U = uq_NatafTransform( X, XMarginals, XCopula)
```

Input

The `uq_NatafTransform` function allows one to transform a sample set `X` (of size $N \times M$) into the standard normal space (space of zero mean, unit variance independent normal variables). The space is defined by the structures `XMarginals` and `XCopula`. The guidelines for specifying `XMarginals` (resp. `XCopula`) can be found in Table 2 (resp. Table 3) in Section 3.1.

Output

`U = uq_NatafTransform(...)` returns an N-by-M matrix `U` that contains N samples from M independent standard normal variables resulting from the Nataf transform of `X`.

3.6.4 uq_invNatafTransform

Syntax

```
X = uq_invNatafTransform( U, XMarginals, XCopula)
```

Input

The `uq_invNatafTransform` function allows one to transform a sample set `U` (of size $N \times M$) drawn from a standard normal random vector into samples of a random vector `X` defined by `XMarginals` and `XCopula`. The guidelines for specifying `XMarginals` (resp. `XCopula`) can be found in Table 2 (resp. Table 3) in Section 3.1.

Output

`X = uq_invNatafTransform( ... )` returns an N-by-M matrix `X` that contains N samples of size M, each row being a realization of the random vector defined by `XMarginals` and `XCopula`.

3.7 Additional functions

3.7.1 uq_sampleU

Syntax

```
U = uq_sampleU(N, M)
U = uq_sampleU(N, M, options)
```

Input

`U = uq_sampleU(N, M)` returns `N` samples of a random vector having M independent, uniform marginals over $[0, 1]$ (i.e. uniform sample over the unit hypercube of dimension M).

`U = uq_sampleU(N, M, options)` returns `N` samples of a random vector having M independent uniform marginals over $[0, 1]$, with sampling options defined in the structure `options` (Table 9).

Table 9: `options` of `uq_sample_u`

☐	`.Method`	String default: `'MC'`	Sampling method
		`'MC'`	Monte Carlo
		`'LHS'`	Latin Hypercube
		`'Sobol'`	Sobol series
		`'Halton'`	Halton series
☐	`.LHSiterations`	Integer default: 5	This option is taken into account when `.Method = 'LHS'`. It refers to the maximum number of iterations to perform in an attempt to improve the design. For more information refer to the MATLAB function `lhsdesign`.
☐	`.SobolGen`	`sobolset` object	Sobol sequence point set
☐	`.HaltonGen`	`haltonset` object	Halton sequence point set

Note: For customizing Sobol or Halton sampling, one can set the relevant fields of `options.SobolGen` or `options.HaltonGen` objects. For more details refer to the MATLAB documentation of `sobolset` and `haltonset` respectively.

Output

`U = uq_sampleU(...)` returns an N-by-M matrix of samples of the unit hypercube.

3.7.2 uq_MarginalFields

Syntax

```
uq_MarginalFields(marginals)
updated_marginals = uq_MarginalFields( ... )
```

Input

The uq_MarginalFields function computes moments of marginals from parameters and vice versa. More precisely uq_MarginalFields computes for each marginal contained in marginals:

- the moments marginals.Moments if the parameters marginals.Parameters are available
- the parameters marginals.Parameters if the moments marginals.Moments are available.

The input variable marginals is a structure as described in Table 2 (Section 3.1).

Output

updated_marginals = uq_MarginalFields(...) returns an updated structure having both the Parameters and Moments fields filled.

3.7.3 uq_estimateMoments

Syntax

```
uq_estimateMoments(marginal)
uq_estimateMoments(marginal, Name, Value)
moments = uq_estimateMoments(...)
[moments, exit_flag] = uq_estimateMoments(...)
[moments, exit_flag, convergence] = uq_estimateMoments(...)
```

Input

The uq_estimateMoments function computes the moments (mean and standard deviation) of a distribution numerically. By default the moments are estimated by numerical integration. In case poor convergence of the integrator is observed the moments are re-estimated using a sampling-based scheme. The sample-based method measures convergence by means of the COV of the moments estimators (Saporta, 2006).

moments = uq_estimateMoments(marginal) calculates the moments of the distribution that is described by the marginal structure as described in Table 2 (Section 3.1).

`moments = uq_estimateMoments(marginal, Name, Value)` allows for fine-tuning various parameters of the moments estimation algorithm by specifying `Name`, and `Value` pairs of options. The available options are summarized in Table 10.

Table 10: Available options of `uq_estimateMoments`

`Name`	`Value` (default)	Description
`'method'`	String (`'Integral'`)	Method for estimating the moments. Currently supported values are `'MC'` for sample-based and `'Integral'` for integral-based estimation.
The following options are taken into account when `'method'= 'MC'`		
`'N0'`	Double (10^6)	Initial sample size
`'Nstep'`	Double (10^5)	Number of additional samples per iteration
`'targetCOV'`	Double (0.01)	The target Coefficient of Variation of the moments estimates
`'maxiter'`	Double (30)	Maximum number of iterations
`'sampling'`	String (`'Sobol'`)	Method for generating samples. See Table 9 for available options.
`'verbose'`	Logical (`false`)	If set to `true` convergence information are printed after each iteration otherwise nothing is printed

Output

`moments = uq_estimateMoments(marginal)` returns a 2×1 vector that contains the mean and standard deviation of the distribution.

`[moments, exit_flag] = uq_estimateMoments(marginal)` additionally returns a boolean variable `exit_flag` that is `true` if the algorithm converged to the specified target Coefficient of Variation or `false` otherwise. This extra output will only be available in case the sampling-based method (`method = 'MC'`) was used.

`[moments, exit_flag, convergence] = uq_estimateMoments(marginal)` additionally returns a $N_i \times 3$ matrix, where N_i is the number of iterations. Each row contains the iteration number and the corresponding estimate of the mean and standard deviation at that iteration. This extra output will only be available in case the sampling-based method (`method = 'MC'`) was used.

3.7.4 `uq_setDefaultSampling`

Syntax

```
uq_setDefaultSampling(method)
uq_setDefaultSampling(myInput, method)
success = uq_setDefaultSampling(...)
```

Description

`uq_setDefaultSampling(method)` sets the default sampling method of the currently selected INPUT object to `method`. The accepted values of `method` can be found in Table 6.

`uq_setDefaultSampling(myInput, method)` sets the default sampling method of the INPUT object `myInput` to `method`.

References

Lebrun, R. and A. Dutfoy (2009). A generalization of the Nataf transformation to distributions with elliptical copula. *Probabilistic Engineering Mechanics 24*(2), 172–178. 18

McKay, M. D., R. J. Beckman, and W. J. Conover (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. 25

Nelsen, R. B. (2006). *An Introduction to Copulas*. Secaucus, NJ, USA: Springer-Verlag New York, Inc. 15, 16, 17

Saporta, G. (2006). *Probabilités, analyse des données et statistique*. Editions Technip. 49