ETH zürich

UQLab User Manual
The Input Module

C. Lataniotis, E. Torre, 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, E. Torre, S. Marelli, B. Sudret, UQLab user manual – The Input module, Report UQLab-V2.0-102, Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland, 2022

BIBTEX entry

```
@TechReport{UQdoc_20_102,
author = {Lataniotis, C. and Torre, E. and Marelli, S. and Sudret, B.},
title = {{UQLab user manual -- The Input module}},
institution = {Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich,
Switzerland},
year = {2022},
note = {Report UQLab-V2.0-102}
}
```

Document Data Sheet

Document Ref.	UQLAB-V2.0-102
Title:	UQLAB user manual – The Input module
Authors:	C. Lataniotis, E. Torre, S. Marelli, B. Sudret Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland
Date:	01/02/2022

Doc. Version	Date	Comments
V2.0	01/02/2022	UQLAB V2.0 release
V1.4	01/02/2021	UQLAB V1.4 release • New distribution: Rayleigh
V1.3	19/09/2019	UQLAB V1.3 release • Added C- and D- vine copulas • Multiple copulas through block-independence • Added support for Rosenblatt transform
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, Isoprobabilistic Transforms, Sampling

Contents

1 Theory **1**

1.1 Introduction . . . 1
1.2 Representation of univariate distributions . . . 1
1.2.1 Parametric distributions . . . 1
1.2.2 Kernel density estimates . . . 2
1.2.3 Truncated distributions . . . 2
1.2.4 Marginals inferred from data . . . 2
1.3 Representation of dependencies through copulas . . . 3
1.3.1 Sklar's transformations for densities and conditional distributions . . . 3
1.3.2 Some properties of copulas . . . 4
1.4 Copulas currently available in UQLAB . . . 4
1.4.1 Independent copula . . . 4
1.4.2 Gaussian Copula . . . 5
1.4.3 Pair copulas . . . 5
1.4.4 Vine copulas . . . 6
1.4.5 Copulas inferred from data . . . 10
1.5 Sampling random vectors . . . 10
1.5.1 Independence copula: probability integral transform . . . 10
1.5.2 Gaussian copula: Nataf transform . . . 11
1.5.3 Other copulas: Rosenblatt transform . . . 12

2 Usage **13**

2.1 Defining an INPUT object . . . 13
2.1.1 Defining the input marginals . . . 13
2.1.2 Defining the input copula . . . 15
2.1.3 Independent subgroups of random variables . . . 19
2.1.4 Defining standard inputs . . . 22
2.1.5 Input summary . . . 22
2.2 Drawing samples from a distribution . . . 23
2.2.1 Selecting an INPUT object and specifying the sampling method . . . 23
2.3 Switching between input objects . . . 25
2.4 Enrichment of an experimental design with new samples . . . 25

2.5 Performing an isoprobabilistic transform . . . 26
2.6 Adding bounds . . . 27
2.7 Defining and using custom marginals . . . 28
2.7.1 Advanced options . . . 29
2.8 Constant variables . . . 30

3 Reference List **31**
3.1 Creating an INPUT object: uq_createInput . . . 33
3.2 Getting samples from an INPUT object: uq_getSample . . . 37
3.3 Printing/Visualizing an INPUT object . . . 38
3.3.1 Printing information: uq_print . . . 38
3.3.2 Graphical visualization: uq_display . . . 38
3.4 Enriching an existing sample set . . . 39
3.4.1 Enriching a Latin Hypercube: uq_enrichLHS . . . 39
3.4.2 Enriching a Sobol sequence: uq_enrichSobol . . . 40
3.4.3 Enriching a Halton sequence: uq_enrichHalton . . . 40
3.4.4 Pseudo-LHS enrichment: uq_LHSify . . . 41
3.5 Sub-sampling an existing sample set: uq_subsample . . . 42
3.6 Transforming samples between spaces . . . 43
3.6.1 uq_GeneralIsopTransform . . . 43
3.6.2 uq_IsopTransform . . . 43
3.6.3 uq_NatafTransform . . . 44
3.6.4 uq_invNatafTransform . . . 44
3.6.5 uq_RosenblattTransform . . . 45
3.6.6 uq_invRosenblattTransform . . . 45
3.7 Convenience functions to define marginals and copulas . . . 46
3.7.1 uq_Marginals . . . 46
3.7.2 uq_StdNormalMarginals . . . 46
3.7.3 uq_StdUniformMarginals . . . 47
3.7.4 uq_KernelMarginals . . . 47
3.7.5 uq_GaussianCopula . . . 48
3.7.6 uq_PairCopula . . . 48
3.7.7 uq_VineCopula . . . 49
3.8 Additional functions . . . 50
3.8.1 uq_sampleU . . . 50
3.8.2 uq_MarginalFields . . . 51
3.8.3 uq_estimateMoments . . . 52
3.8.4 uq_setDefaultSampling . . . 53
3.8.5 uq_CopulaSummary . . . 53

4 Appendices **55**
A List of univariate distributions supported in UQLAB 55
A.1 Uniform distribution 55
A.2 Gaussian (Normal) 55
A.3 Lognormal distribution 57
A.4 Gumbel distribution 58
A.5 Gumbel-min distribution 59
A.6 Weibull distribution 60
A.7 Gamma distribution 61
A.8 Exponential distribution 62
A.9 Beta distribution 63
A.10 Triangular distribution 64
A.11 Logistic distribution 65
A.12 Laplace distribution 66
A.13 Rayleigh distribution 67
References 69

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 univariate and 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 dependence structure (copula). The marginals and the copula can thus be defined and specified separately, as detailed in the next sections.

1.2 Representation of univariate distributions

A random variable X can be represented by its univariate cumulative distribution function (CDF) $F_X(x) = \mathbb{P}(X \leq x)$. One then write $X \sim F_X$. The derivative f_X of a continuous CDF F_X, if existing, is called the probability density function (PDF) of X.

1.2.1 Parametric distributions

UQLAB supports a number of continuous, differentiable 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 Appendix A, a more extensive overview of each distribution and its properties (*e.g.*, PDF, cumulative distribution function, support, moments, parameters) are given for reference.

1.2.2 Kernel density estimates

Kernel density estimation (KDE), also called kernel smoothing, is a non-parametric method to infer a univariate distribution directly from data. Given a set $\mathcal{X} = \{x^{(1)}, \ldots, x^{(n)}\}$ of n independent identically distributed (i.i.d) observations from an unknown PDF $f_{\boldsymbol{X}}$, the KDE estimate of $f_{\boldsymbol{X}}$ is defined as

$$\hat{f}_{\mathcal{X}}(x) = \frac{1}{nw} \sum_{h=1}^{n} k\left(\frac{|x - x^{(h)}|}{w}\right). \tag{1.1}$$

where $k(\cdot)$ is a non-negative function (kernel), which decays with the distance between the point x where the distribution is evaluated and the point $x^{(h)} \in \mathcal{X}$ where the kernel is centered. The parameter w, called the bandwidth, dictates how fast $\hat{f}_{\mathcal{X}}$ decays with $|x - x^{(h)}|$. A common choice for the kernel is the standard Gaussian distribution.

1.2.3 Truncated distributions

A truncated distribution results from restricting the support of a probability distribution. Assume that a random variable X follows some distribution with CDF F_X. Then, the CDF and inverse CDF ("quantile function") of the distribution truncated in $[a, b]$ are defined as $F_{X|[a,b]}(x) = F_X(x|a \leq X \leq b)$ and $F^{-1}_{X|[a,b]}(x) = F^{-1}_{X|[a,b]}(x|a \leq X \leq b)$, respectively, and are given by

$$F_{X|[a,b]}(X) = \begin{cases} 0 & , \quad X \leq a \\ \frac{F_X(X) - F_X(a)}{F_X(b) - F_X(a)} & , \quad a < X < b \\ 1 & , \quad X \geq b \end{cases} \tag{1.2}$$

and

$$F^{-1}_{X|[a,b]}(u) = \begin{cases} a & , \quad u = 0 \\ F_X^{-1}\left(F_X(a) + u\left(F_X(b) - F_X(a)\right)\right) & , \quad 0 < u < 1 \\ b & , \quad u = 1 \end{cases}. \tag{1.3}$$

A practical example of how truncated distributions are defined in UQLAB is given in Section 2.6.

1.2.4 Marginals inferred from data

UQLAB provides the possibility to infer marginals from data based on different goodness-of-fit criteria. More details about the theory underlying this topic and its usage in UQLAB are

provided in the UQLAB User Manual – Statistical inference.

1.3 Representation of dependencies through copulas

An M-dimensional copula (or M-copula for brevity) is formally defined as a multivariate distribution over $[0,1]^M$ with uniform marginals in the unit interval. At the basis of the copula formalism lies Sklar's theorem, which states that, for any M-variate distribution $F_{\boldsymbol{X}}$ of an M-dimensional input random vector $\boldsymbol{X} = \{X_1, \ldots, X_M\}$ with marginals $F_{X_1}, \ldots, F_{X_M}$, there exists a copula C such that

$$F_{\boldsymbol{X}}(\boldsymbol{x}) = C(F_{X_1}(x_1), F_{X_2}(x_2), ..., F_{X_M}(x_M)). \tag{1.4}$$

Thus, the copula C is the function that links the marginals F_{X_i} of a random vector to its joint CDF $F_{\boldsymbol{X}}$. C itself does not depend on the F_{X_i}, and purely describes the statistical interactions among the components X_i of $\boldsymbol{X}$, that is, their *dependence structure*. Besides, C is unique if all F_{X_i} are continuous. The converse also holds: for any M-copula C and any set of marginals $F_{X_1}, \ldots, F_{X_M}$, the function $F_{\boldsymbol{X}}$ given by (1.4) is a joint CDF with marginals $F_{X_1}, \ldots, F_{X_M}$. This property is exploited in UQLAB to define any joint distribution by specifying separately its marginals and copula. This framework is ideal for uncertainty quantification. 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 compound information about the joint distribution 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) and Joe (2015) for more details and relevant literature.

1.3.1 Sklar's transformations for densities and conditional distributions

Relations analogous to (1.4) hold for probability densities and conditional distributions. From (1.4) the joint PDF of $\boldsymbol{X}$ is obtained by differentiation:

$$f_{\boldsymbol{X}}(\boldsymbol{x}) = c\left(F_{X_1}(x_1), \ldots, F_{X_M}(x_M)\right) \prod_{i=1}^{M} f_{X_i}(x_i), \tag{1.5}$$

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

$$c(u_1, \ldots, u_M) = \frac{\partial^M C(u_1, \ldots, u_M)}{\partial u_1 \ldots \partial u_M}. \tag{1.6}$$

For conditional densities

$$\begin{aligned} f_{X_1|X_2,\ldots,X_M}(x_1|x_2, \ldots, x_M) &= \frac{f_{\boldsymbol{X}}(\boldsymbol{x})}{\prod_{i=2}^{M} f_{X_i}(x_i)} \\ &= c\left(F_{X_1}(x_1), \ldots, F_{X_M}(x_M)\right) f_{X_1}(x_1). \end{aligned} \tag{1.7}$$

1.3.2 Some properties of copulas

As mentioned in Section 1.3, a copula describes the dependence between the random variables it couples, independent of their marginals. The following properties of copulas hold:

- Given two random vectors $\boldsymbol{X}$ and $\boldsymbol{X}'$ such that $X_i' = \alpha_i(X_i)$, where $\alpha_i(\cdot)$ is a monotone increasing transformation for each $i = 1, \ldots, M$, then it is easy to prove that $C_{\boldsymbol{X}} = C_{\boldsymbol{X}'}$;
- In particular, the random vector $\boldsymbol{U}$ such that $U_i = F_{X_i}(X_i)$ has copula $C_{\boldsymbol{U}} \equiv C_{\boldsymbol{X}}$;
- Since $\boldsymbol{U}$ has standard uniform marginals, $C_{\boldsymbol{X}}$ is also its joint CDF;
- Thus, $C_{\boldsymbol{X}}$ can be interpreted as the joint distribution of transformations of the original random variables into random variables with standard uniform distribution.

Besides, a useful property is the following: the copula $C_{\boldsymbol{X}}$ of the random vector $\boldsymbol{X} = (\boldsymbol{X}_1, \boldsymbol{X}_2)$ such that $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ are mutually independent is given by

$$C_{\boldsymbol{X}} \equiv C_{\boldsymbol{X}_1} \cdot C_{\boldsymbol{X}_2}. \tag{1.8}$$

(An analogous expression holds for the copula density $c_{\boldsymbol{X}}$ of $\boldsymbol{X}$). By recursion, the formula trivially generalizes to any number of independent subgroups. This makes it possible to easily express the joint copula density/distribution between independent groups of random variables as the product of the individual copula densities/distributions.

1.4 Copulas currently available in UQLAB

UQLAB currently supports a variety of copula families to represent dependence between two or more random variables. The currently supported families are briefly characterized below.

1.4.1 Independent copula

$$C(u_1, \ldots, u_M) = \prod_{i=1}^{M} u_i \tag{1.9}$$

By substituting Eq. (1.9) in (1.4), it is clear that it corresponds to the case in which the components of the input random vector are independent. Indeed:

$$F_{\boldsymbol{X}}(\boldsymbol{x}) = C(F_{X_1}(x_1), \ldots, F_{X_M}(x_M)) = \prod_{i=1}^{M} F_{X_i}(x_i). \tag{1.10}$$

Consequently, the independent copula density is simply $c(u_1, \ldots, u_M) = 1$. A graphical representation of the independence copula PDF in dimension $M = 2$ is given for reference in Figure 1, left panel.

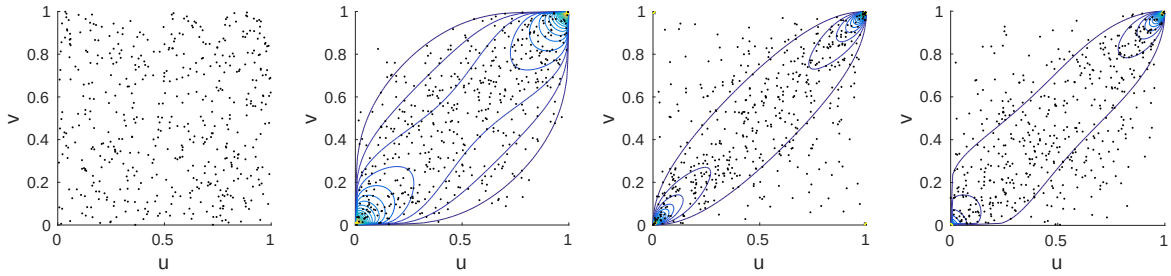

Figure 1: Scatter plot and PDF contour plot of different pair copulas. From left to right: independence copula, Gaussian copula with parameter $\rho = 0.7$, t- copula with parameters $\rho = 0.7, \nu = 2$, and Gumbel copula with parameter $\tau = 1.5$.

1.4.2 Gaussian Copula

$$C(u_1, \ldots, u_M; \mathbf{R}) = \Phi_M \left(\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_M); \mathbf{R}\right) \tag{1.11}$$

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 in dimension $M = 2$ is given for reference in Figure 1, second panel.

1.4.3 Pair copulas

UQLAB supports a variety of parametric pair copulas to represent dependencies among $M = 2$ random variables. These copula families cover different aspects of bivariate dependence, such as upper or lower tail dependence, asymmetric tails, and others. While some of these families admit an extension to the multivariate case ($M > 2$), this extension is typically non-unique and is not supported in UQLAB. Flexible multivariate families in UQLAB are instead implemented in terms of vine copulas (see Section 1.4.4).

A list of pair copulas supported in UQLAB is provided in Table 1. The densities of some of them, as well as samples drawn these densities, are illustrated in Figure 1. For each supported pair copula density, UQLAB. provides its rotations by $90°$, $180°$, and $270°$. A pair copula rotated by $90°$ or $270°$ transforms positive dependence into negative and vice versa; a copula rotated by $180°$ can be used to switch the upper and lower tails. These rotations are

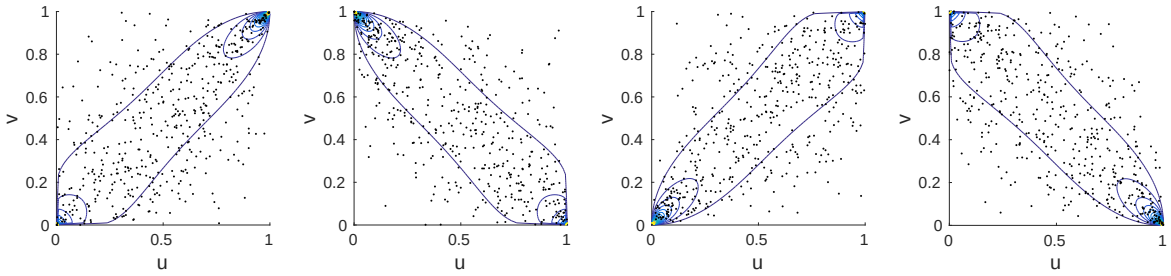

Figure 2: Rotated Gumbel copula. From left to right: rotation by 0, 90, 180, and 270 degrees. The copula parameter is $\rho = 2$.

defined by the following equations:

$$\begin{aligned} C^{(90)}(u,v) &= v - C(1-u,v), \\ C^{(180)}(u,v) &= u + v - 1 + C(1-u,1-v), \\ C^{(270)}(u,v) &= u - C(u,1-v). \end{aligned} \tag{1.12}$$

(Note that $C^{(90)}$ and $C^{(270)}$ are obtained by flipping the copula density c around the vertical and horizontal axis, respectively; some references provide the formulas for actual rotations: $C^{(90)}(u,v) = v - C(v,1-u)$, $C^{(270)}(u,v) = u - C(1-v,u)$).

An example of rotated Gumbel pair copula is provided in Figure 2. The non-rotated Gumbel copula assigns positive dependence as well as upper tail dependence to the random variables it couples. The latter is visible as an accumulation of points in the upper right corner of the unit square. The rotated versions implement negative dependence (rotations by $90°$ and $270°$) or lower tail dependence ($180°$).

1.4.4 Vine copulas

Flexible parametric families of multivariate copulas ($M > 2$) can be obtained by the product of (possibly conditional) pair copulas. This construction, known as pair copula or vine construction, was first introduced by Bedford and Cooke (2002) and later popularized by Aas et al. (2009). To get an intuition why this construction is possible, let us first recall the chain rule of probability which, in its version for probability densities, states that

$$f_{\boldsymbol{X}}(\boldsymbol{x}) = \prod_{j=1}^{M} f_{j|j+1,\dots,M}(x_j|x_{j+1},\dots,x_M), \tag{1.13}$$

where $f_{j|j+1,\dots,M}$ is the conditional PDF of X_j given $X_{j+1},\dots,X_M$. Recalling (1.5),

$$c\left(F_{X_1}(x_1),\dots,F_{X_M}(x_M)\right)\prod_{i=1}^{M} f_{X_i}(x_i) = \prod_{j=1}^{M} f_{j|j+1,\dots,M}(x_j|x_{j+1},\dots,x_M). \tag{1.14}$$

Each conditional PDF on the right hand side of (1.14) can in turn be written in terms of a pair copula density, that is a pair copula differentiated with respect to one of its arguments.

Table 1: **Pair copula families supported in UQLAB.** From left to right: copula name, CDF, and parameter range. (a) Φ is the univariate standard normal distribution, and $\Phi_{2;\theta}$ is the bivariate normal distribution with zero means, unit variances and correlation parameter θ. (b) t_ν is the univariate t distribution with ν degrees of freedom, and $t_{\nu,\theta}$ is the bivariate t distribution with ν degrees of freedom and correlation parameter θ.

Name	$C(u,v;\boldsymbol{\theta})$	Parameter range
Independence	uv	
Clayton	$(u^{-\theta}+v^{-\theta}-1)^{-1/\theta}$	$\theta>0$
Frank	$-\frac{1}{\theta}\log\left(\frac{1-e^{-\theta}-(1-e^{-\theta u})(1-e^{-\theta v})}{1-e^{-\theta}}\right)$	$\theta\in\mathbb{R}\backslash\{0\}$
Gaussian	$\Phi_{2;\theta}\left(\Phi^{-1}(u),\Phi^{-1}(v)\right)$ $^{(a)}$	$\theta\in(-1,1)$
Gumbel	$\exp\left(-((-\log u)^\theta+(-\log v)^\theta)^{1/\theta}\right)$	$\theta\in[1,+\inf)$
t-	$t_{2;\nu,\theta}\left(t_\nu^{-1}(u),t_\nu^{-1}(v)\right)$ $^{(b)}$	$\nu>1$, $\theta\in(-1,1)$

Table 2: **Some properties of pair copulas supported in UQLAB.** Kendall's tau, tail dependence coefficients, subfamilies of pair copulas that obtain for specific parameter values. See also Torre et al. (2019).

Name	τ_K	λ_l	λ_u	Special cases
Clayton	$\frac{\theta}{\theta+2}$	$2^{-1/\theta}$	0	—
Frank	$1+\frac{4}{\theta}(\frac{1}{\theta}\int_0^\theta t(e^t-1)^{-1}dt-1)$	0	0	—
Gaussian	$\frac{2}{\pi}\arcsin(\theta)$	0	0	—
Gumbel	$\frac{\theta-1}{\theta}$	0	$2-2^{1/\theta}$	—
t	$\frac{2}{\pi}\arcsin(\theta)$	$\lambda_l=\lambda_u=$ $^{(d)}$ $=2t_{\nu+1}\left(-\sqrt{(\nu+1)(1-\theta)/(1+\theta)}\right)$		—

To understand why, for a general subset of indices $\mathcal{A}\subset\{1,\ldots,M\}$, let $F_{i|\mathcal{A}}$, $f_{i|\mathcal{A}}$, , $C_{i|\mathcal{A}}$ and $c_{i|\mathcal{A}}$ indicate the CDF, PDF, copula distribution and copula density of the random variable X_i

conditioned on $\boldsymbol{X}_{\mathcal{A}}$, respectively. Joe (1996) showed that, for any $(j,i) \in \overline{\mathcal{A}} \times \mathcal{A}$,

$$F_{j|\mathcal{A}}(x_j|\boldsymbol{x}_{\mathcal{A}}) = \frac{\partial C_{ji|\mathcal{A}\setminus\{i\}}(u_j,\, u_i)}{\partial u_i}\Big|_{(F_{j|\mathcal{A}\setminus\{i\}}(x_j|\boldsymbol{x}_{\mathcal{A}\setminus\{i\}}),\, F_{i|\mathcal{A}\setminus\{i\}}(x_i|\boldsymbol{x}_{\mathcal{A}\setminus\{i\}}))}. \tag{1.15}$$

Differentiating both sides with respect to u_j yields

$$f_{j|\mathcal{A}}(x_j|\boldsymbol{x}_{\mathcal{A}}) = c_{ji|\mathcal{A}\setminus\{i\}}(F_{j|\mathcal{A}\setminus\{i\}}(x_j|\boldsymbol{x}_{\mathcal{A}\setminus\{i\}}),\, F_{i|\mathcal{A}\setminus\{i\}}(x_i|\boldsymbol{x}_{\mathcal{A}\setminus\{i\}})) \cdot f_{j|\mathcal{A}\setminus\{i\}}(x_j|\boldsymbol{x}_{\mathcal{A}\setminus\{i\}}). \tag{1.16}$$

In other words, a conditional PDF can be written as the product between a pair copula and a conditional PDF of lower order. The latter can be in turn be written as the product of another pair copula and a conditional PDF of yet lower order, and so on. Eventually, one can rewrite the right hand side of (1.14) as a product of (conditional) pair copulas multiplied by $\prod_i F_{X_i}(x_i)$. The latter factor appears in the left hand side of (1.14) too, and can be simplified. In conclusion, the joint copula density is expressed as a product of pair copulas:

- $M-1$ unconditional pair copulas,
- $M-2$ pair copulas conditioned on 1 variable,
- ...
- 1 pair copula conditioned on $M-2$ variables,

for a total of $M(M-1)/2$ pair copulas. Copulas expressed in this way are called vine copulas. Note that the order of the variables used for the factorization in (1.14), as well as the index i selected in (1.16), are arbitrary. Each choice corresponds to a different factorization of the joint copula density into pair copula densities. To help organizing these different constructions, Bedford and Cooke (2002) introduced two (non-exhaustive, but very broad) classes of vine, representing two different choices for the vine factorization. These two classes, called *canonical* (C-) and *drawable* (D-) vines, are supported in UQLAB and are defined as follows.

- **C-vine**

$$c(\boldsymbol{u}) = \prod_{j=1}^{M-1} \prod_{i=1}^{M-j} c_{j,j+i|\{1,\ldots,j-1\}}(u_{j|\{1,\ldots,j-1\}},\, u_{j+i|\{1,\ldots,j-1\}}), \tag{1.17}$$

  A C-vine thus expresses an M-copula as a tensor product of

  - $M-1$ pair copulas between variable X_1 and X_j, $j=2,\ldots,M$,
  - $M-2$ pair copulas between X_2 and X_j conditioned on X_1, $j=3,\ldots,M$,
  - ...
  - the pair copula between X_{M-1} and X_M, conditioned on $X_1,\ldots,X_{M-2}$.

  A graphical representation is provided in Figure 3, left panel.

- **D-Vine**

$$c(\boldsymbol{u}) = \prod_{j=1}^{M-1} \prod_{i=1}^{M-j} c_{i,i+j|\{i+1,\ldots,i+j-1\}}(u_{i|\{i+1,\ldots,i+j-1\}},\, u_{i+j|\{i+1,\ldots,i+j-1\}}). \tag{1.18}$$

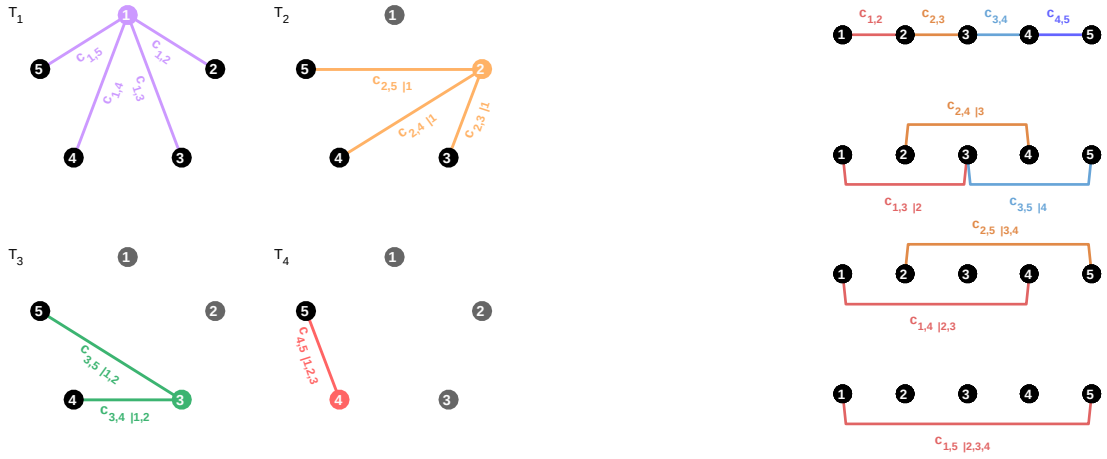

Figure 3: **Graphical representation of C- and D-vines.** The pair copulas in each tree of a 5-dimensional C-vine (left; conditioning variables are shown in grey) and of a 5-dimensional D-vine (right; conditioning variables are those between the connected nodes).

A D-vine thus expresses an M-copula as a tensor product of

- $M-1$ unconditional pair copulas between variable X_i and X_{i+1}, $i = 1, \ldots, M-1$,
- $M-2$ pair copulas between X_i and X_{i+2} conditioned on X_{i+1}, $i = 1, \ldots, M-2$,
- ...
- the pair copula between X_1 and X_M, conditioned on $X_2, \ldots, X_{M-1}$.

A graphical representation of this structure is provided in the right panel of Figure 3.

C- and D-vine constructions require in general to specify different sets of pair copulas (except when $M = 3$, in which case each D-vine is a C-vine and vice versa). Thus, one may prefer one construction or the other depending on the type of information available. Besides, the pair copulas involved in each of the two classes are fully determined by the order of the variables $X_1, \ldots, X_M$. An M-copula thus admits a factorization into $M!/2$ different C-vines and $M!/2$ different D-vines (the two classes being equivalent if $M = 3$).

Truncated vines. The pair copulas that a vine comprises can be grouped into different *trees*. The first tree consists of all independent $M-1$ unconditional pair copulas, the second tree of all $M-2$ pair copulas between random variables conditioned on another single variable, and so on until the $M-1$-th and last tree, consisting of a single pair copula between two random variables conditioned on all other variables. In equations (1.17) and (1.18), the tree index is represented by the variable j.
A vine is said to be *truncated* at level $t^* \in \{1, \ldots, M-1\}$, if all pair copulas belonging to any tree $t \geq t^*$ are the independence pair copula. The rationale for working with truncated vines is that, often, only conditional probabilities of low order can be reasonably asserted or reliably inferred from data, whereas imposing complex models for conditional probabilities

of higher order may correspond to over-interpreting or over-fitting data. Furthermore, conditional probabilities of higher order typically have a weaker effect on the joint PDF of the input random variables than those of lower order, especially if the strongest correlations are captured in earlier trees, and can thus be neglected.

1.4.5 Copulas inferred from data

In addition to user-defined parametric copulas, UQLAB supports the possibility to infer copulas from data. The theory underlying this topic and its usage in UQLAB are extensively covered in the companion UQLAB User Manual – Statistical inference.

1.5 Sampling random vectors

Various uncertainty quantification tasks require to generate data according to a specified joint distribution. For instance, resampling from a known distribution of the inputs to a system enables statistical estimation of the output by Monte-Carlo or by more sophisticated strategies.
A number of random sampling strategies (e.g. Monte Carlo sampling, latin hypercube sampling (LHS), pseudorandom sequences, etc.) are available to produce samples in the standard uniform space, that is, for $\boldsymbol{Z} \sim \mathcal{U}([0,1]^M)$. A possible strategy to sample from a different distribution $F_{\boldsymbol{X}}$ is thus to transform a sample $\boldsymbol{z}$ of $\boldsymbol{Z}$ into a sample $\boldsymbol{x}$ of $\boldsymbol{X}$, if such a transformation exists. Maps of the form (Lebrun and Dutfoy, 2009)

$$\boldsymbol{X} = \mathcal{T}(\boldsymbol{U}) \qquad s.t. \qquad X \sim F_{\boldsymbol{X}}, \quad \boldsymbol{U} \sim F_{\boldsymbol{U}}, \tag{1.19}$$

which transform a random vector $\boldsymbol{U} \sim F_{\boldsymbol{U}}$ (or a sample $\boldsymbol{u}$ thereof) into a random vector $\boldsymbol{X} \sim F_{\boldsymbol{X}}$ (or a sample $\boldsymbol{x}$ therefore) are called *isoprobabilistic transforms*. Amongst the many strategies available to generate samples distributed according to a more general distribution $F_{\boldsymbol{X}}$, UQLAB makes use of isoprobabilistic transforms. Of particular interest for generating samples of $\boldsymbol{X} \sim F_{\boldsymbol{X}}$ are isoprobabilistic transforms from $\boldsymbol{U} \sim \mathcal{U}([0,1]^M)$ to $\boldsymbol{X}$. In the following, isoprobabilistic transforms to different target distributions $F_{\boldsymbol{X}}$ will be derived. We will see that different transforms exist depending on the copula of $C_{\boldsymbol{X}}$ of $F_{\boldsymbol{X}}$.

1.5.1 Independence copula: probability integral transform

A well known result of probability theory states that, for any random variable X with continuous CDF F_X, the random variable

$$U = F_X(X) \tag{1.20}$$

is uniformly distributed in the unit interval: $U \sim \mathcal{U}([0,1])$. The map (1.20) is also known as *probability integral transform* (PIT) of X. Its inverse

$$X = F_X^{-1}(U) \tag{1.21}$$

thus maps $U \sim \mathcal{U}([0,1])$ into $X \sim F_X$. The inverse PIT provides a simple way to transform an M-variate sample $\mathcal{U} = \left\{\boldsymbol{u}^{(1)}, \ldots, \boldsymbol{u}^{(N)}\right\} \sim \mathcal{U}([0,1]^M)$ into a sample $\mathcal{X} = \left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\right\} \sim F_{\boldsymbol{X}}$, for any multivariate distribution $F_{\boldsymbol{X}}$ with independent marginals F_{X_i} – that is, with independent copula (1.9). For all $i = 1, \ldots, N$ and all $j = 1, \ldots, M$,

$$x_j^{(i)} = F_{X_j}^{-1}(u_j^{(i)}). \tag{1.22}$$

We call (1.22) the inverse PIT for random vectors.

1.5.2 Gaussian copula: Nataf transform

In the case of dependent variables, several extra steps are needed to transform a sample set distributed according to $\mathcal{U}([0,1]^M)$ into a sample set with the desired joint PDF. When the target joint PDF has Gaussian copula, a powerful tool to achieve this goal is the inverse Nataf transform.

We first recall a fundamental result of probability theory: Gaussian random variables are mutually independent if and only if they are uncorrelated. Based on this property, a Gaussian random vector $\boldsymbol{V}$ with correlation matrix $\mathbf{R}$ can be transformed into a Gaussian random vector $\boldsymbol{W}$ with uncorrelated (and therefore independent) components by

$$\boldsymbol{W} = \boldsymbol{\Gamma}^{-1}\boldsymbol{V}, \tag{1.23}$$

where $\boldsymbol{\Gamma}^{-1}$ is the inverse Choleski factor of $\mathbf{R}$, satisfying the matrix equation $\boldsymbol{\Gamma}\boldsymbol{\Gamma}^T = \mathbf{R}$.

The Nataf transform maps a random vector $\boldsymbol{X}$ with Gaussian copula $C_{\boldsymbol{X}}$ given by (1.11) and arbitrary continuous marginals F_{X_i} into $\boldsymbol{W} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_M)$, where $\mathbf{I}_M$ is the identity matrix in $\mathbb{R}^M$, in three steps:

- $\boldsymbol{X} \mapsto \boldsymbol{U} \sim C_{\boldsymbol{X}} : U_i = F_{X_i}(X_i) \sim \mathcal{U}([0,1])$ (PIT of $\boldsymbol{X}$)
- $\boldsymbol{U} \mapsto \boldsymbol{V} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) : V_i = \Phi^{-1}(U_i) \sim \mathcal{N}([0,1])$, where Φ is the standard normal CDF
- $\boldsymbol{V} \mapsto \boldsymbol{W} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_M)$ through (1.23).

The first two steps involve monotonically increasing transformations of the marginals, thus do not change the Gaussian copula of $\boldsymbol{X}$ (see Section 1.3.2). Therefore, $\boldsymbol{V}$ has standard Gaussian marginals and Gaussian copula, that is, Gaussian joint PDF. Consequently, the random vector $\boldsymbol{W}$ obtained by the last step is indeed multivariate standard normal. The inverse map, called inverse Nataf transformation, maps the standard normal space to the space with probability measure $F_{\boldsymbol{X}}$.

To further map $\boldsymbol{W}$ onto the uniform standard space, only the marginals of $\boldsymbol{W}$ need to be transformed, by

- $\boldsymbol{W} \mapsto \boldsymbol{Z} : Z_i = \Phi(W_i)$.

The inverse transformation is used to map the standard uniform space into the probability space identified by $F_{\boldsymbol{X}}$, and thus to sample from $F_{\boldsymbol{X}}$. It reads:

1. $\boldsymbol{Z} \mapsto \boldsymbol{W} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_M) : W_i = \Phi^{-1}(Z_i)$
2. $\boldsymbol{W} \mapsto \boldsymbol{V} = \boldsymbol{\Gamma}\,\boldsymbol{U} \sim \mathcal{N}(\mathbf{0}_M, \mathbf{R})$
3. $\boldsymbol{V} \mapsto \boldsymbol{U} \sim C_{\boldsymbol{X}} : U_i = \Phi(V_i)$
4. $\boldsymbol{U} \mapsto \boldsymbol{X} \sim F_{\boldsymbol{X}} : X_i = F_{X_i}^{-1}(U_i)$

1.5.3 Other copulas: Rosenblatt transform

The most general case of generating samples with an arbitrary M-dimensional copula can be addressed by making use of the inverse Rosenblatt transform. First introduced by Rosenblatt (1952), The Rosenblatt transform maps a random vector $\boldsymbol{X}$ with continuous marginals into a random vector $\boldsymbol{Z}$ with independent, $\mathcal{U}([0,1])$ components. It reads

$$\begin{cases} Z_1 = F_{X_1}(X_1) \\ Z_2 = F_{X_2|X_1}(X_2|X_1) \\ \vdots \\ Z_M = F_{X_M|X_1,\ldots,X_{M-1}}(X_M|X_1,\ldots,X_{M-1}) \end{cases} . \tag{1.24}$$

Within the copula formalism, one can perform the transform in two steps:

- $\boldsymbol{X} \mapsto \boldsymbol{U} \sim C_{\boldsymbol{X}}$, by the PIT (1.20)
- $\boldsymbol{U} \mapsto \boldsymbol{Z} : Z_i = C_{X_i|X_1,\ldots,X_{i-1}}(u_i|u_1,\ldots,u_{i-1})$.

The transformation is invertible (all conditional distributions are monotonically increasing). Its inverse can be used to generate samples of $\boldsymbol{X}$ from samples of $\boldsymbol{Z}$.
The applicability of the inverse Rosenblatt transform requires knowledge of the conditional quantile functions $C^{-1}_{X_i|X_1,\ldots,X_{i-1}}(u_i|u_1,\ldots,u_{i-1})$, $i = 1,\ldots,M$. While these are not available for all classes of parametric copulas, they are for some (at least numerically), including vine copulas (for details, see Aas et al. (2009)). UQLAB provides algorithms to sample from C- and D-vines, and uses an original implementation that capitalizes on the efficient vectorization of the numerical inversion process.

Note: while the inverse Rosenblatt transform is a valid alternative to the inverse Nataf transform for Gaussian copulas, it is computationally less efficient. The latter is therefore used by UQLAB in the presence of a Gaussian copula.

Chapter 2

Usage

2.1 Defining an INPUT object

As discussed in the theory chapter, a random vector in UQLAB is defined by its marginal distributions and the copula that defines their dependence structure. Marginals and copula can be defined independently from each other, and together fully determine the joint PDF of the input via the relation (1.4).

2.1.1 Defining the input marginals

2.1.1.1 The general case

An M-dimensional input requires the specification of M marginal distributions.
The following code defines a bivariate Gaussian random vector $\boldsymbol{X} = [X_1, X_2]^T$ with independent components (independence copula). The mean value and standard deviation of X_1 (resp. X_2) are $\mu_1 = 1$, $\sigma_1 = 1$ (resp. $\mu_2 = 2$, $\sigma_2 = 0.5$):

```
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 the copula has not been explicitly assigned: when not defined, UQLAB by default assumed it to be the independence copula (1.9). The corresponding code is (to be typed before using `uq_createInput`):

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

Also note that the first and second marginals have been specified through their parameters and moments, respectively. For every marginal, either information is accepted, but not both. When the INPUT object `myInput1` is created, all parameters and moments for all variables are computed and possible inconsistencies are checked.

2.1.1.2 Special cases of univariate distributions

Most of the available (built-in) distributions can be defined similarly to the way a Gaussian distribution was defined in Section 2.1.1.1, *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 Appendix A. Some special cases are the following:

- For the *exponential* distribution only one parameter (λ) exists. When such a distribution is defined by its parameters only one element is needed (λ). Additionally when it is defined by its moments only one element is needed again which corresponds to the mean and standard deviation (that are equal).

- For the *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 A.9.

- A marginal obtained through kernel smoothing on a data set `X` can be represented by

```
Input.Marginals.Type = 'KS';
Input.Marginals.Parameters = X;
```

  which by default uses a Gaussian kernel with bandwidth optimized on the data. The additional subfield `.Options` can be used to optionally specify the kernel type, bandwidth, and the distribution bounds. For instance, the additional lines of code

```
Input.Marginals.Options.Bandwidth = 0.1;
Input.Marginals.Options.Support = [0, 5];
```

  specify that a kernel bandwidth $w = 0.1$ must be used, and that the KDE distribution is defined within the interval $[0, 5]$.

Note: For the KDE, UQLAB relies on the MATLAB function `ksdensity` to fit distributions, the options of which are specified in `.Options`. The `.Support` defined in this field is different from the bounds (c.f. Section 2.6) for truncated distributions in Section 1.2.3. The `.Support` only applies to KDE to have distributions within a certain interval, whereas the bounds defined in Section 2.6 are used to truncate a distribution to a given range.

2.1.1.3 Marginals inferred from data

The code needed to instruct UQLAB to infer the marginals from data is explained in detail in the companion UQLAB User Manual – Statistical inference.

2.1.1.4 Convenience functions for specific marginals

A number of convenience functions are available to quickly generate data structures that define marginals of specific types:

- `uq_Marginals` can be used to conveniently generate any given number of marginals of the same parametric family and parameters (or moments);
- `uq_StdUniformMarginals` specifically generates standard uniform marginals;
- `uq_StdNormalMarginals` specifically generates standard normal marginals;
- `uq_KernelMarginals` specifically generates marginals by kernel smoothing.

For more details on these functions, see Section 3.7.

2.1.2 Defining the input copula

Let us now create another INPUT object where dependency between the marginals is introduced by imposing a copula. Below, various parametric copulas supported by UQLAB are covered. For copulas inferred from data, see the companion UQLAB User Manual – Statistical inference.

2.1.2.1 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 here taken to be $\mathbf{R} = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$ (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);
```

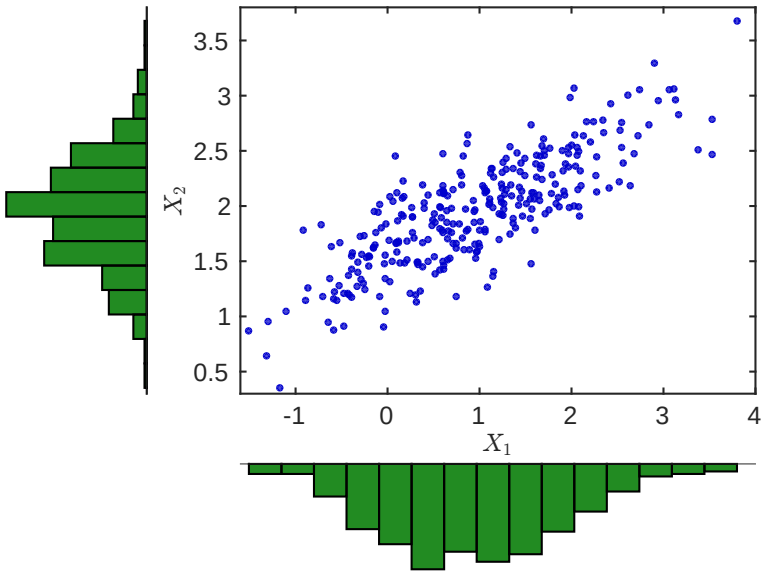

Figure 4: Samples drawn from a 2-D Gaussian distribution with a Gaussian copula (correlation coefficient $\rho_{12} = 0.8$).The sampling method is plain Monte Carlo.

The resulting samples are plotted in Figure 4. 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.

Note: The code above assumes that the fields `Input.Marginals` have been previously assigned, as in Section 2.1.1.1. In other words, after creating the INPUT `myInput1`, we replace the independence copula by the Gaussian one in the `Input.Copula` field, and create a second INPUT `myInput2`.

2.1.2.2 Pair copula

A pair copula is defined in UQLAB by

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

Additionally, the copula's `Family` and `Parameters` must be provided. As an optional argument, pair copula densities of standard parametric families can be rotated by 90, 180 or 270 degrees through the additional field `Rotation`.
The code below defines a Clayton pair copula (see Table 1), whose density is rotated by $90°$. This allows us to introduced a negative correlation between the coupled random variables X_1 and X_2. Beside, the non-rotated Clayton pair copula has lower tail dependence coefficient. Therefore, its rotated version assigns lower tail dependence between $-X_1$ and X_2, as apparent from the cloud of points extending into the lower right corner of Figure 5.

```
Input.Name = 'Input 3: Negative correlation';
Input.Copula.Type = 'Pair';
Input.Copula.Family = 'Clayton';
Input.Copula.Rotation = 90;
Input.Copula.Parameters = 1.5;
myInput3 = uq_createInput(Input);
X = uq_getSample(300);
```

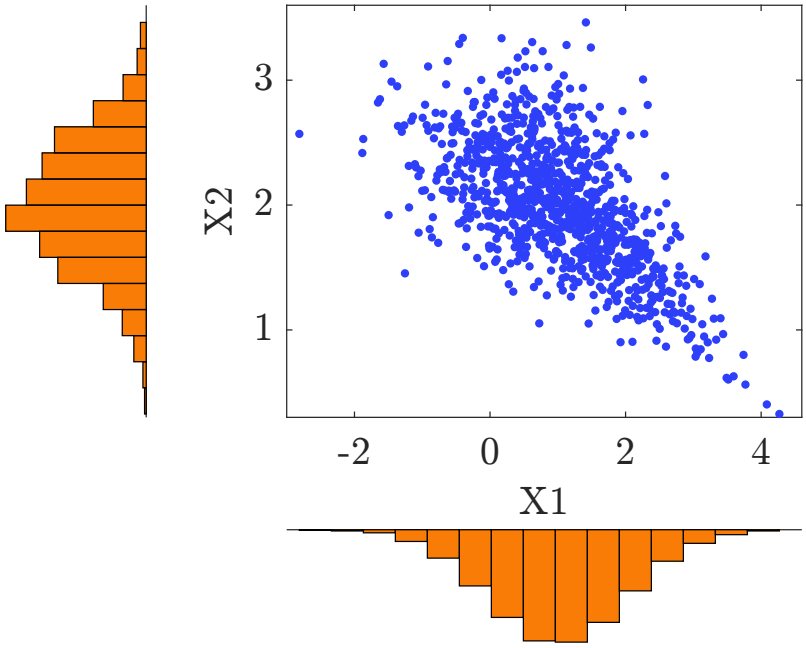

Figure 5: Monte Carlo samples drawn from a 2-D distribution with Gaussian marginals and Clayton copula ($\rho = 1.5$) rotated by $90°$.

2.1.2.3 Vine copula: C-Vine and D-Vine

Specifying a higher-dimensional copula by vine construction (see Section 1.4.4) requires to define all comprising pair copulas. The vine structure specifies which of the $M!/2$ possible orders of the variables is chosen for the construction, and thus which set of pair copulas are explicitly provided in the vine construction. UQLAB currently supports two classes of vines: canonical (C-) vines and drawable (D-) vines.

The example below constructs a 3-dimensional C-vine obtained as a tensor product of pair copula densities c_{21}, c_{23} and $c_{13|2}$, that is, vine structure $2-1-3$. The three copulas are, respectively: a `'t'`- copula with parameters $[.4, 2]$, a `'Frank'` copula with parameter 0.5, and the `'Independence'` copula. The third variable is assigned an exponential marginal. A sample set from this distribution is shown in Figure 6.

```
Input.Name = 'Input 4: 3D C-Vine' ;
```

```
Input.Marginals(1).Type = 'Gaussian';
Input.Marginals(1).Parameters = [1 1];
Input.Marginals(2).Type = 'Gaussian';
Input.Marginals(2).Moments = [2 0.5];
Input.Marginals(3).Type = 'Exponential';
Input.Marginals(3).Parameters = 1.5;
Input.Copula.Type = 'CVine';
Input.Copula.Structure = [2 1 3];
Input.Copula.Families = {'t', 'Frank', 'Independence'};
Input.Copula.Rotations = [0, 0, 0];
Input.Copula.Parameters = {[.4, 2], .5, []};
myInput4 = uq_createInput(Input);
X = uq_getSample(300);
```

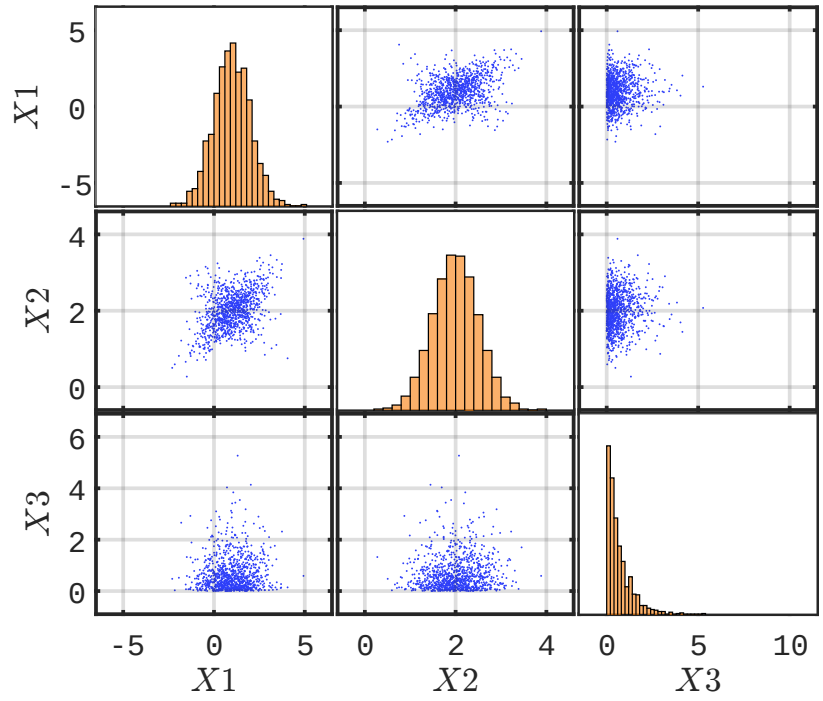

Figure 6: Monte Carlo samples drawn from a 3-D distribution with C-vine copula.

For a complete list of options to specify for a vine copula, see Table 6 and Table 8.

The set of pair copulas that a vine comprises are often not easy to keep track of, especially with higher dimension M. This information can be print on screen using the code below (for a 4-dimensional `'CVine'` with structure $[4\,1\,2\,3]$):

```
uq_CopulaSummary('CVine', [4 1 3 2]);
```

which prints the text:

```
CVine, dimension 4, structure [4 1 3 2]. Pair copulas:
Index | Pair Copula
=================================
1     | C_4,1
2     | C_4,3
```

```
3       | C_4,2
4       | C_1,3|4
5       | C_1,2|4
6       | C_3,2|4,1
```

For a detailed use of the function `uq_CopulaSummary`, see Section 3.8.5.
Additionally, a graph of the vine's constituting pair copulas and edges among them, such as those shown in Figure 3, can be visualized using the function `uq_drawVineCopula`.

2.1.2.4 Truncated vines

As discussed in Section 1.4.4, the higher-order dependencies needed for the construction of a vine copula may be difficult to assert, or to reliably infer from data. Also, when setting the vine structure such that pair copulas among more strongly correlated random variables are captured in earlier trees, higher-order dependencies play a less significant role in shaping the joint PDF and can often be neglected. This is also the approach taken by UQLAB and by most software when inferring vine copulas from data (see the UQLAB User Manual – Statistical inference, Section 1.3.3).
Representing a vine truncated at tree `t` is possible by using the additional code

```
Input.Copula.Truncation = t;
```

for any $t = 1, \ldots, M$, where $t = 1$ sets all pair copulas to the independence pair copula, and thus the vine to the independence M-copula, and $t = M$ corresponds to no truncation. Values of t lower than 1 or larger than M are also interpreted by UQLAB as no truncation.

Note: For a vine truncated at level t, only the pair copulas in the first $t-1$ trees need to be specified. The others are by definition the independence copula.

Since the j-th tree contains $M-j$ copulas, a total of $(M-1)+(M-2)+\ldots+(M-t-1) = (t-1)M - t(t+1)/2$ are to be defined (see Table 3 for the number of non-truncated pair copulas in a vine for different values of M and t). UQLAB additionally handles several special cases as follows:

- an error is thrown if not enough or too many pair copulas are specified (for instance, if $M = 10$, $t = 3$ and less than 17 or more than 45 pair copulas are provided);
- a warning is given if more pair copulas than the non-truncated ones are specified (for instance, if $M = 10$, $t = 3$ and more than 17 pair copulas are provided).

2.1.3 Independent subgroups of random variables

As mentioned in Section 1.3.2, the copula of a random vector consisting of independent subgroups of random variables is given by the product of the copulas of all subgroups (see (1.8) for the case $d = 2$). Subgroup independence can be specified in UQLAB by simply defining any number of input copulas, and the variables associated to each one. The full copula is automatically understood to be the tensor product of the individual copulas. For

Table 3: Number of non-truncated pair copulas for a vine of dimension M truncated at tree t. The last row, where $t = M$, corresponds to the no-truncation case and indicates the total number of pair copulas in the vine. *: no truncation, because $t \geq M$: all pair copulas need to be specified.

t	M							
	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9
3	3*	5	7	9	11	13	15	17
4	3*	6*	9	12	15	18	21	24
5	3*	6*	10*	14	18	22	26	30
⋮								
M	3	6	10	15	21	28	36	45

instance, the code below defines an input consisting of 5 standard normal random variables, three of which (X_1, X_4 and X_5) are coupled by a Gaussian copula, the remaining coupled by a Gumbel pair copula:

```
iOpts = struct;
iOpts.Marginals = uq_StdNormalMarginals(5);
iOpts.Copula(1).Type = 'Gaussian';
iOpts.Copula(1).RankCorr = [1 .5 -.3; .5 1 .2; -.3 .2 1];
iOpts.Copula(1).Variables = [1 4 5];
iOpts.Copula(2).Type = 'Pair';
iOpts.Copula(2).Family = 'Gumbel';
iOpts.Copula(2).Parameters = 1.5;
iOpts.Copula(2).Variables = [2 3];
myInput = uq_createInput(iOpts);
uq_print(myInput)
```

```
=============================================================
Input object name: Input 7
Dimension(M): 5

Marginals:

Index | Name | Type     |  Parameters              | Moments
-------------------------------------------------------------------------
1     | X1   | Gaussian |  0.00e+00, 1.00e+00   | 0.00e+00, 1.00e+00
2     | X2   | Gaussian |  0.00e+00, 1.00e+00   | 0.00e+00, 1.00e+00
3     | X3   | Gaussian |  0.00e+00, 1.00e+00   | 0.00e+00, 1.00e+00
4     | X4   | Gaussian |  0.00e+00, 1.00e+00   | 0.00e+00, 1.00e+00
5     | X5   | Gaussian |  0.00e+00, 1.00e+00   | 0.00e+00, 1.00e+00


Copula:

Tensor product of 2 copulas between the random vectors
X_[1 4 5], X_[2 3]

Copula 1, of X_[1 4 5]:
Type: Gaussian
Dimension: 3
Variables coupled: [1 4 5]
Parameters:
[+1.0000 +0.5176 -0.3129 ;
+0.5176 +1.0000 +0.2091 ;
-0.3129 +0.2091 +1.0000 ]

Copula 2, of X_[2 3]:
Type: Pair
Variables coupled: [2 3]
Family: Gumbel
Rotation: 0
Parameters: 1.5

=============================================================
```

All copula types can be freely combined, and there is no restriction on their individual parameters.

2.1.4 Defining standard inputs

Some multivariate distributions are often used in various types of statistical analysis. This is the case of the M-variate standard uniform (resp. standard normal) distributions, that is, distributions with $\mathcal{U}([0,1])$ (resp. $\mathcal{N}(0,1)$) marginals. To create an input object with standard uniform (resp. standard normal) marginals and copula `myCopula`, the convenience function `uq_StdUniformMarginals` (resp. `uq_StdNormalMarginals`) can be used:

```
Input.Marginals = uq_StdUniformMarginals(10);
Input.Copula = myCopula;
myInput1 = uq_createInput(Input);
```

(For details on how to specify the copula, see Section 2.1.2).
For a comprehensive list of all the available convenience functions in the INPUT module, please refer to Section 3.7.

2.1.5 Input summary

Once an INPUT object has been created, a summary of its marginals and copulas can be printed on screen via the `uq_print` command.

```
Input.Marginals(1) = uq_Marginals(1, 'Gaussian', [2 0.5]);
Input.Marginals(2) = uq_Marginals(1, 'Gaussian', [1 1]);
Input.Copula = uq_GaussianCopula([1 .8; .8, 1]);
myInput1 = uq_createInput(Input);
uq_print(myInput1)
```

```
=============================================================
Input object name: Input 1
Dimension(M): 2

Marginals:

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


Copula:

Type: Gaussian
Dimension: 2
Variables coupled: [1 2]
Parameters:
        [+1.0000 +0.8000 ;
         +0.8000 +1.0000 ]

=============================================================
```

For visual inspection of an INPUT object, the function `uq_display` can be used as follows:

```
uq_display(myInput1)
```

The result is shown in Figure 7.

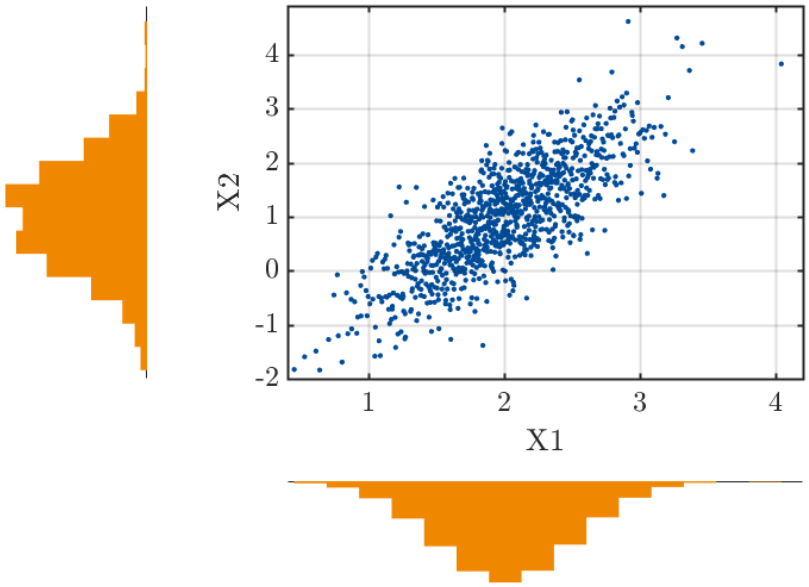

Figure 7: The output of the function `uq_display` on the INPUT object `myInput1`. By default, the samples are drawn using Monte Carlo sampling.

2.2 Drawing samples from a distribution

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 2.3. The MATLAB standard random number generator is used by default.

2.2.1 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 an INPUT object different from the last defined one:

- 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 8. For a list of all the available sampling methods refer to Table 11 in Section 3.2.

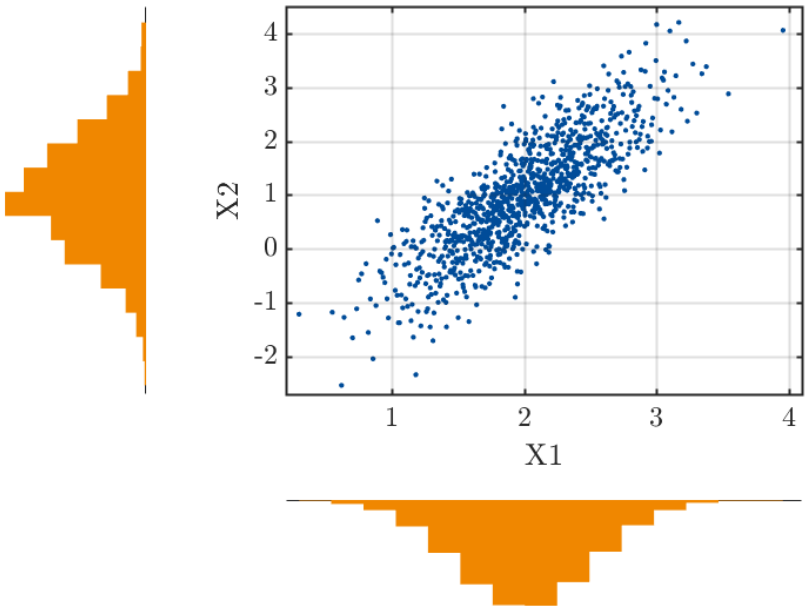

Figure 8: Samples drawn from INPUT `myInput1` using the Latin Hypercube sampling method.

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.

2.3 Switching between input objects

Suppose we want to draw two sample sets:

- 200 Monte Carlo samples from the input vector `myInput1` defined above, and having components coupled by a Gaussian copula;
- 300 LHS samples from the input vector `myInput2` defined right below, whose marginal X_1 is bounded and independent from X_2.

```
Input2.Marginals(1) = uq_Marginals(1, 'Gaussian', [2 0.5]);
Input2.Marginals(2).Bounds = [-3, 2];
Input2.Marginals(2) = uq_Marginals(1, 'Gaussian', [1 1]);
myInput2 = uq_createInput(Input2);
```

This is carried out as follows:

```
uq_selectInput(myInput1);
X2 = uq_getSample(200,'MC');
uq_selectInput(myInput2);
X3 = uq_getSample(300,'LHS');
```

The two samples are plotted in Figure 9.

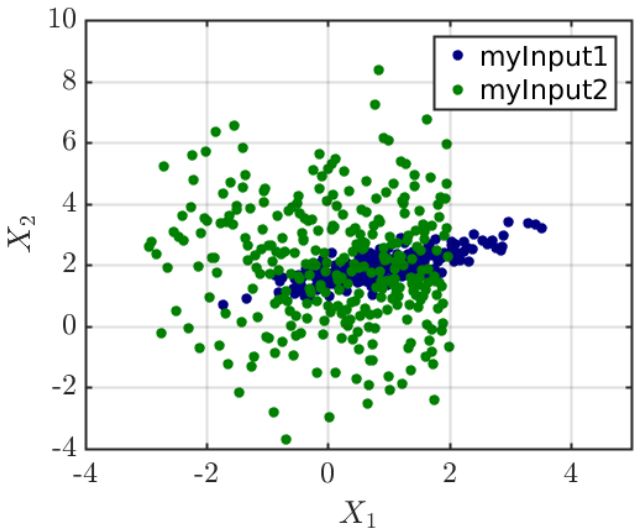

Figure 9: Samples drawn from two different cases of a 2-D Gaussian distribution. The INPUT object `myInput2` has dependent marginals and the INPUT object `myInput3` has independent marginals and bounds on $X_1 \in [-3, 2]$. The sampling method is Latin Hypercube sampling.

2.4 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 8) 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);
```

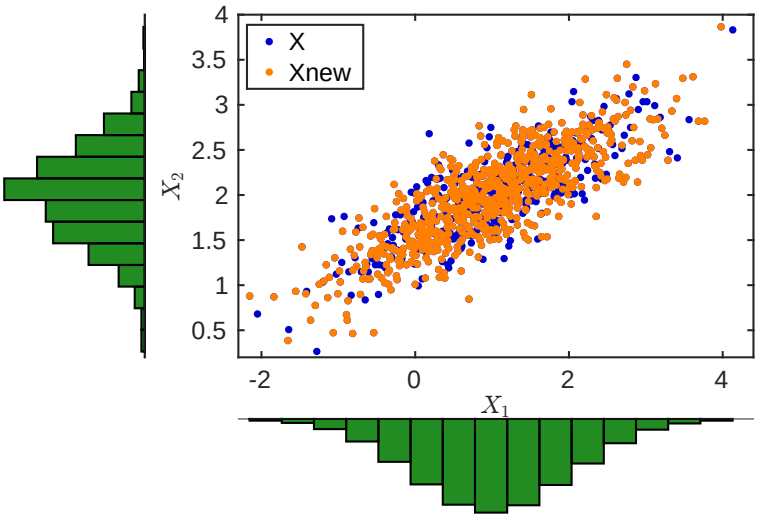

Figure 10: Enrichment of the LHS samples drawn from a 2-D Gaussian distribution with a Gaussian copula using `uq_enrichLHS`. The initial sample set is shown in Figure 8.

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.

2.5 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).
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 11.

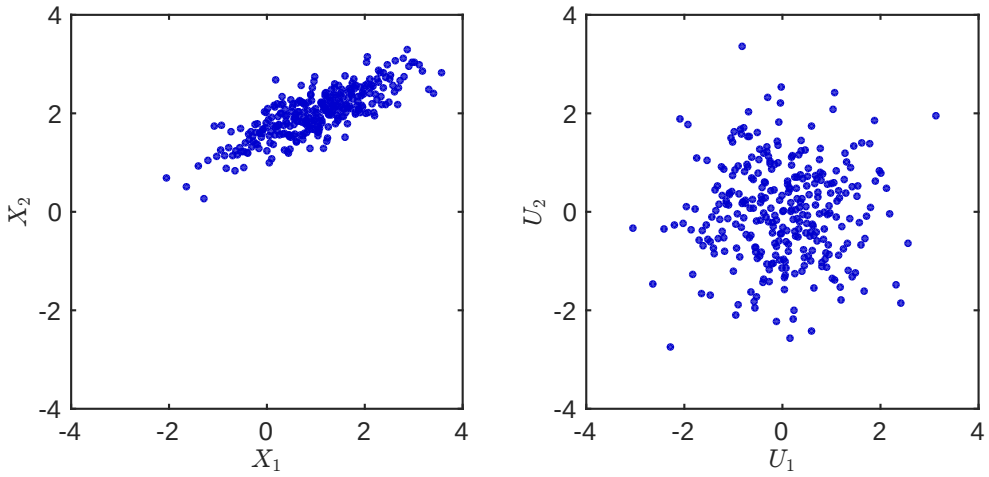

Figure 11: Isoprobabilistic transform from the physical space (2D Gaussian, left) to the standard normal space (right).

2.6 Adding bounds

A marginal distribution truncated (oa.k.a bounded) in an interval $[a, b]$ which is contained in the distribution's domain can be specified by

```
Input.Marginals(1).Bounds = [a b];
```

For instance, the code below defines 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 12.

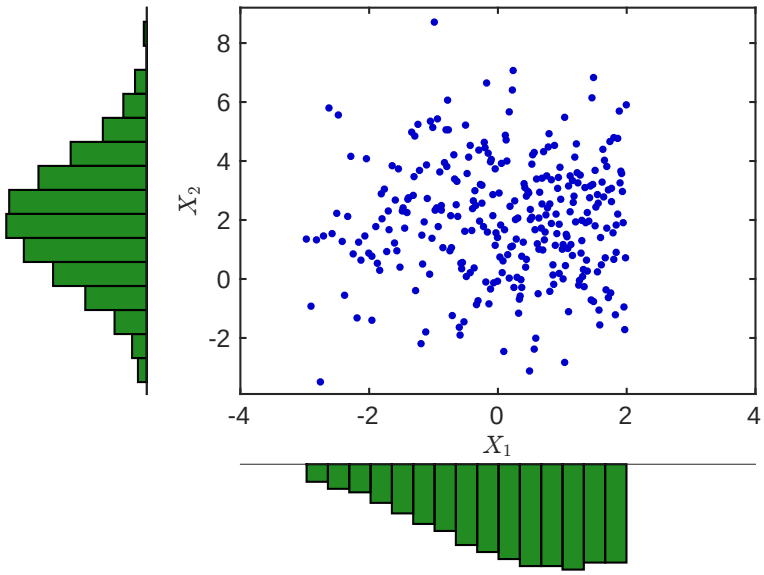

Figure 12: Samples drawn from a 2-D Gaussian distribution with independent marginals and bounds on $X_1 \in [-3, 2]$. The sampling method is Latin Hypercube sampling.

2.7 Defining and using custom marginals

The built-in probability distributions can be found in Table 5 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 ()within 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 the following 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)
```

The input arguments `X` and `F` of these functions 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);
```

2.7.1 Advanced options

On top of the three functions defined above, the user may define an additional function that computes the distribution moments from its natural parameters as followa

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

where `parameters` is an array of arbitrary size containing the distribution parameters and `moments` is the vector $[\mu, \sigma]$ containing 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 3.8.3). 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 must be specified in case the user wants to define the distribution based on its moments, rather than on its parameters specifying the distribution based on its mean and standard deviation is required. In this case, a function of the following form is required:

```
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.

2.8 Constant variables

Constants are degenerate random variables that always take the same scalar value. A constant variable can be specified as follows:

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

A random variable with zero variance is automatically interpreted as a constant. For example, the INPUT object created by the code

```
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);
```

has the second marginal set to the constant 1. The following warning message is raised:

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

Similarly, a bounded random variable with identical upper and lower bound is reverted to a constant. For example, the INPUT object created by the code

```
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);
```

has the second marginal set to the constant 0. The following warning is raised:

```
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 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: `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 4.

Table 4: `Input`

●	`.Marginals`	Table 5	The options regarding the marginals of the random vector
☐	`.Copula`	Table 6	The options regarding the copula (or copulas) 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 5.

Table 5: `Input.Marginals`

●	`.Type`	String	Type of marginal distribution
		`'Constant'`	A constant value
		`'Gaussian'`	Gaussian distribution (Section A.2)
		`'Lognormal'`	Lognormal distribution (Section A.3)
		`'Uniform'`	Uniform distribution (Section A.1)
		`'Exponential'`	Exponential distribution (Section A.8)
		`'Beta'`	Beta distribution (Section A.9)
		`'Weibull'`	Weibull distribution (Section A.6)
		`'Gumbel'`	Gumbel maximum extreme value distribution (Section A.4)
		`'GumbelMin'`	Gumbel minimum extreme value distribution (Section A.5)
		`'Gamma'`	Gamma distribution (Section A.7)
		`'Triangular'`	Triangular distribution (Section A.10)
		`'Logistic'`	Logistic distribution (Section A.11)

		`'Laplace'`	Laplace distribution (Section A.12)
		other String	A user-defined marginal type (Section 2.7)
⊕	`.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 6.

Table 6: `Input.Copula`

●	`.Type`	String default: `'Independent'`	Copula type.
		`'Independent'`	Independent copula.
		`'Gaussian'`	Gaussian copula
		`'Pair'`	Pair copula
		`'CVine'`	C-vine copula
		`'DVine'`	D-vine copula

⊞	`.Parameters`	• For `'Independent'` copula: ignored; • For `'Gaussian'` copula: $M \times M$ Double; • For `'Pair'` copula: variable length Double; • For `'C'`/`'DVine'`: cell of $M(M-1)/2$ Doubles;	• The `'Independent'` copula has no parameters • Linear correlation matrix of the `'Gaussian'` copula • Array of `'Pair'` copula parameter(s) • Parameters of each pair copula in the `'CVine'`/`'DVine'`
⊞	`.RankCorr`	$M \times M$ Double	Spearman correlation matrix (`'Gaussian'` copula only)

Additional mandatory and optional options accepted when `Copula.Type='Pair'` (resp. `Copula.Type='CVine'` or `'DVine'`) are listed in Table 7 (resp. Table 8).

Table 7: `Input.Copula: additional fields when Input.Copula.Type='Pair'`

●	`.Family`	String	Pair copula family
		`'Independent'` or `'Independence'`	Independence pair copula
		`'Gaussian'`	Gaussian pair copula
		`'Clayton'`	Clayton pair copula
		`'Gumbel'`	Gumbel pair copula
		`'Frank'`	Frank pair copula
		`'t'`	t- pair copula
□	`.Rotation`	Double default: 0 Either 0, 90, 180 or 270	Rotation of copula density; see (1.13)

Table 8: `Input.Copula: additional fields when Input.Copula.Type='CVine'/'DVine'`

●	`.Structure`	Array of M integers	Vine structure. Can be any permutation of $[1, 2, \ldots, M]$
●	`.Families`	Cell of Strings	Families of pair copulas composing the vine. Possible values are listed in Table 7 (see `.Family`)
□	`.Rotations`	Array of Doubles default: $[0, 0, \ldots, 0]$ Entries must takes values 0, 90, 180 or 270	Rotation of each pair copula density; see (1.13)
□	`.Truncation`	Integer default: M	Vine truncation level

NOTE: Instead of manually specifying the structures `Input.Marginals` and `Input.Copula`, it is possible to obtain them through convenience functions described in Section 3.7. Beside, the following functions may be useful to properly specify suitable copulas:

- `uq_PairCopulaParameterRange(family)`: provided a pair copula family as a char (*e.g.*, `'Gumbel'`), this function returns the range of admissible values for each parameter of the copula;
- `uq_CopulaSummary`: for a copula of vine type, prints the pairs of variables coupled by each pair copula in a specified vine, and the corresponding conditioning variables. For more details, see Section 3.8.5.

Output

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

Table 9: `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 5 for contents
`.Copula`	Struct	Information regarding the copula, see Table 6 for contents
`.Internal`	Struct	Internal fields that are only of interest for scientific developers

The structure `Sampling` contains the following fields :

Table 10: `myInput.Sampling`

`.DefaultMethod`	String Table 11	The default sampling method
`.Method`	String Table 11	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 11: `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 12.

Table 12: 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

`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 13.

Table 13: 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 5 and `Input.Copula` from Table 6 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 5 (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 5 (resp. Table 6) 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 5 (resp. Table 6) 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`.

3.6.5 uq_RosenblattTransform

Syntax

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

Input

The `uq_RosenblattTransform` function allows one to transform a sample set `X` (of size $N \times M$) into the standard uniform space (space independent variables with uniform distribution in $[0,1]$). The original space is defined by the structures `XMarginals` and `XCopula`. The guidelines for specifying `XMarginals` (resp. `XCopula`) can be found in Table 5 (resp. Table 6) in Section 3.1.

Output

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

3.6.6 uq_invRosenblattTransform

Syntax

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

Input

The `uq_invRosenblattTransform` function allows one to transform a sample set `U` (of size $N \times M$) drawn from a standard uniform 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 5 (resp. Table 6) in Section 3.1.

Output

`X = uq_invRosenblattTransform( ... )` 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 Convenience functions to define marginals and copulas

3.7.1 uq_Marginals

The uq_Marginals function generates a structure describing M marginal distributions all of the specified type and parameters (or moments). uq_Marginals allows one to quickly define a multivariate INPUT with identical marginals, as in the following examples:

```
% Generate an Input object with 3 Gumbel marginals of parameters
% 1 and 2, and independence copula
iOpts.Marginals = uq_Marginals(3, 'Gumbel', [1, 2]);
myInput = uq_createInput(iOpts);

% Generate a trivariate input with Gumbel marginals of moments
% 2 and 2, and independence copula
iOpts.Marginals = uq_Marginals(3, 'Gumbel', {}, [2, 2]);
myInput = uq_createInput(iOpts);
```

Syntax

```
myMarginals = uq_Marginals(Dimension, Type, Parameters, Moments)
```

Input

Dimension: integer describing the number of standard normal marginal.
Type : char, name of distribution family (common to each marginal)
Parameters / Moments: array, distribution parameters / moments. Only one can be specified, not both. Set the other to {}.

Output

Structure myMarginals with elements myMarginals(i), $i = 1, \ldots, M$, each having the following fields

```
myMarginals(i).Type = 'Gaussian'
myMarginals(i).Parameters = [0,1]
```

3.7.2 uq_StdNormalMarginals

The uq_StdNormalMarginals function generates a structure describing M standard normal marginal distributions. It is a special case of uq_Marginals with Type='Gaussian' and Parameters=[0 1]. uq_StdNormalMarginals allows one to quickly define a multivariate standard normal INPUT by the following code:

```
iOpts.Marginals = uq_StdNormalMarginals(M);
myInput = uq_createInput(iOpts);
```

Syntax

```
myMarginals = uq_StdNormalMarginals(M)
```

Input

`M`: integer describing the number of standard normal marginal.

Output

Structure `myMarginals` with elements `myMarginals(i)`, $i = 1, \ldots, M$, each having the following fields

```
myMarginals(i).Type = 'Gaussian'
myMarginals(i).Parameters = [0,1]
```

3.7.3 uq_StdUniformMarginals

The `uq_StdUniformMarginals` function generates a structure describing M standard uniform distributions, that is, uniform distributions in the interval $[0, 1]$. The syntax is the same as for `uq_StdNormalMarginals` (see above). The generated marginals, however, are of the type `'Uniform'`.

3.7.4 uq_KernelMarginals

Syntax

```
myInput = uq_KernelMarginals(X, bounds, bandwidth)
```

Creates a structure that describes the marginal distributions of the data points in `X` as kernel density estimates.

Input

`X` : matrix of input data.

Each column `X_i=X(:,ii)` represents data from input variable X_i, $i = 1, \ldots, M$.

`bounds`: positive float or $M \times 2$ Double, optional.

If specified, the marginals are bounded. Can be:

- a positive float: defined $dX_j = \max(X_j) - \min(X_j)$, each j-th marginal is bounded in $[\min(X_j) - \texttt{bounds} * dX_j,\ \max(Xj) + \texttt{bounds} * dX_j]$
- an $M \times 2$ Double: marginal `jj` is bounded in `bounds(jj,:)`
- an cell of `M` elements: marginal `jj` is bounded in `bounds`

`bandwidth`: float, cell, or array, optional.
The Gaussian kernel bandwidth. Can be:

- a positive float: the kernel bandwidth for each dimension
- an array/cell of M positive floats: the bandwiths of each dimension

Output

`Marginals`: structure
Structure that contains M marginals obtained by kernel smoothing.

3.7.5 uq_GaussianCopula

Usage

```
Copula = uq_GaussianCopula(C, corrType)
```
Convenience function to create a data structure describing a Gaussian copula with specified correlation matrix.

Input

`C`: $M \times M$ double
correlation matrix of the M-variate Gaussian copula

`corrType` : char, optional
the type of correlation provided in `C`. Either `'Linear'` (equivalently `'Pearson'`), `'Spearman'`, or `'Kendall'`.
Default: `'Linear'`

Output

`Copula`: struct
Structure that contains the specified Gaussian copula, with the fields `.Type`, `.Dimension`, `.Parameters`, `.RankCorr`, `.RhoS`, `.TauK`, `.CholR`

3.7.6 uq_PairCopula

Usage

```
Copula = uq_PairCopula(family, theta, rotation)
```
Convenience function to create a data structure describing the specified pair copula. The structure is used in copula-related operations in UQLAB, for instance as part of an INPUTobject.

Input

`family`: char

the pair copula family

`theta` : array of doubles

the copula parameters

`rotation`: double, optional
the rotation of the copula density. Can be: 0, 90, 180, 270.
Default: 0 (no rotation).
Note: rotations 90 and 270 are obtained by flipping the copula density around the lines $u_1 = 0.5$ and $u_2 = 0.5$, respectively.

Output

`Copula`: struct
Structure that contains the specified pair copula, with the fields `.Type`, `.Family`, `.Dimension`, `.Parameters`, `.Rotation`

3.7.7 uq_VineCopula

Usage

```
Copula = uq_VineCopula(type, structure, families, thetas, rotations, truncation)
```

Convenience function to create a data structure describing the specified vine copula.

Input

`type`: char

either `'CVine'` or `'DVine'`

`structure` : array of M integers

array with all and only integers 1 to M, in any order.

`families`: cell of $M(M-1)/2$ chars

the families of pair copulas comprising the vine

`thetas`: cell of $M(M-1)/2$ arrays

each j-th element in the cell is the (list of) parameter(s) of the j-th pair copula

`rotations`: $1 \times M(M-1)/2$ Double, optional
the rotation of each pair copula density (0, 90, 180, or 270)
Default: 0 (no rotation)

`truncation`: Integer, optional

the vine truncation. Pair copulas in the vine with higher conditioning order that the vine truncation are set to be the independence pair copula.
Default: $M-1$ (no truncation)

Note: Vine copulas truncated at tree $t < M-1$ need the specification of only $P = \sum_{j=1}^{t}(M-j)$ pair copulas. Consequently, the input parameters `families`, `thetas` and `rotations` need to contain only P elements for truncated vines. Additional ones are ignored.

Output

`Copula`: struct

Structure that describes the specified vine copula

3.8 Additional functions

3.8.1 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 14).

Table 14: `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 space-filling properties of the sample set. 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.

3.8.2 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 5 (Section 3.1).

Output

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

3.8.3 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 5 (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 15.

Table 15: 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 14 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.

3.8.4 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 11.

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

3.8.5 uq_CopulaSummary

Syntax

This function has to different uses, with syntax either
`uq_CopulaSummary(Copula)`,
OR
`uq_CopulaSummary(VineType, Structure)`.
It prints a summary of the specified copula structure (first syntax) or of the pair copulas in a vine with the specified type and structure.

Input

`Copula`: struct

A structure describing a copula

OR

`VineType`: char

vine type (either `'CVine'` or `'DVine'`)

`Structure`: array

vine structure (order of the variables in the vine construction)

Output

None. The function prints a message on screen.

Chapter 4

Appendices

A List of univariate distributions supported in UQLab

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.

A.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.

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

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.

A.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,

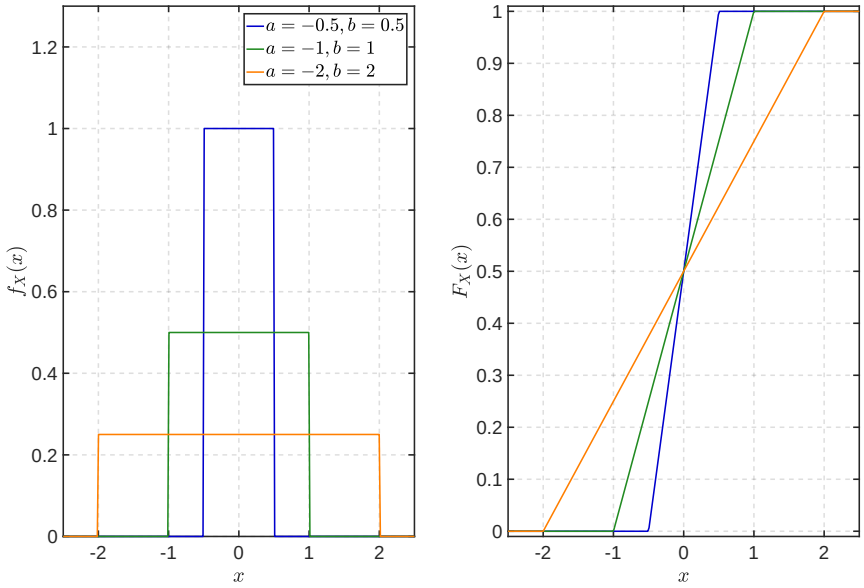

Figure 13: PDF and CDF of uniform distributions for various parameter values.

noise terms, etc.

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

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

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \tag{1}$$

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:

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

All normal distributions can be represented as linear transforms of standard normal distribu-

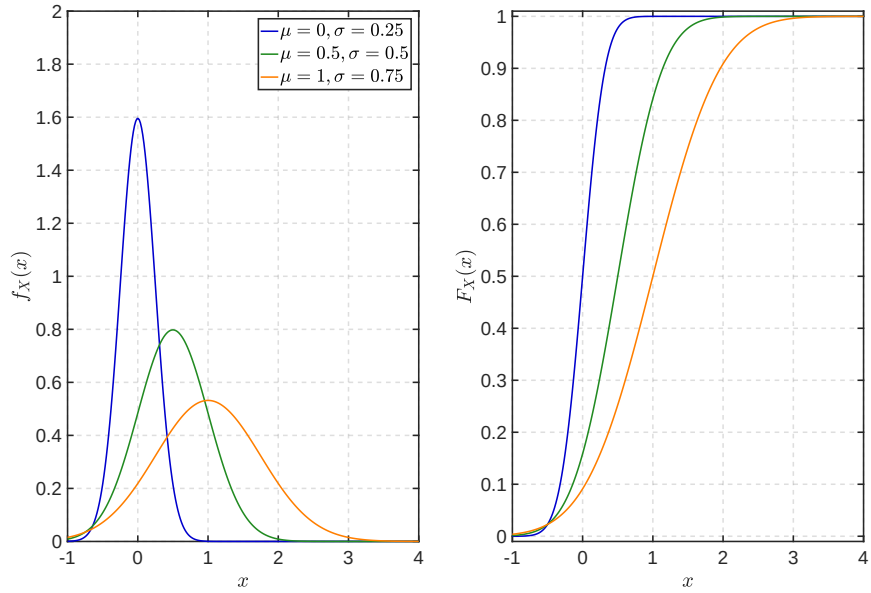

Figure 14: PDF and CDF of Gaussian distributions for various parameter values.

tions as follows:

$$\mathcal{N}(\mu, \sigma) = \mu + \sigma\mathcal{N}(0, 1) \tag{3}$$

A.3 Lognormal distribution

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

$$X \sim \mathcal{LN}(\lambda, \zeta) \Leftrightarrow \ln(X) \sim \mathcal{N}(\lambda, \zeta) \tag{4}$$

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

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).

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 lognor-

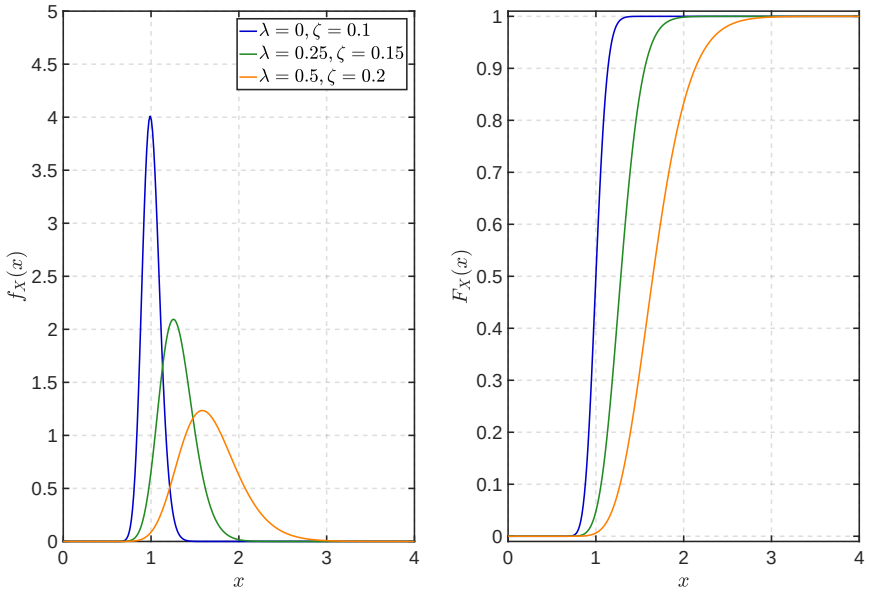

Figure 15: PDF and CDF of lognormal distributions for various parameter values.

mally distributed variables is that their products and ratios are also lognormally distributed.

A.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* 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.

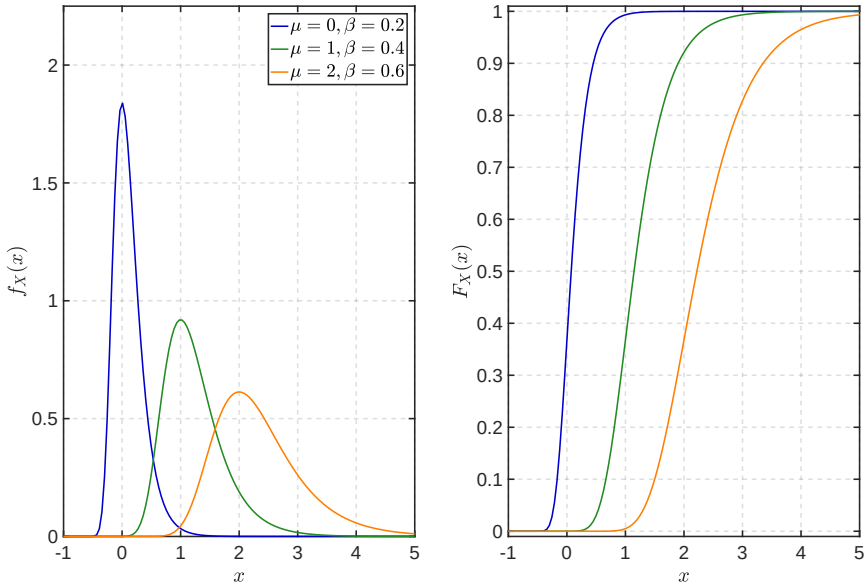

Figure 16: PDF and CDF of Gumbel maximum extreme value distributions for various parameter values.

A.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.

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 16 and 17).

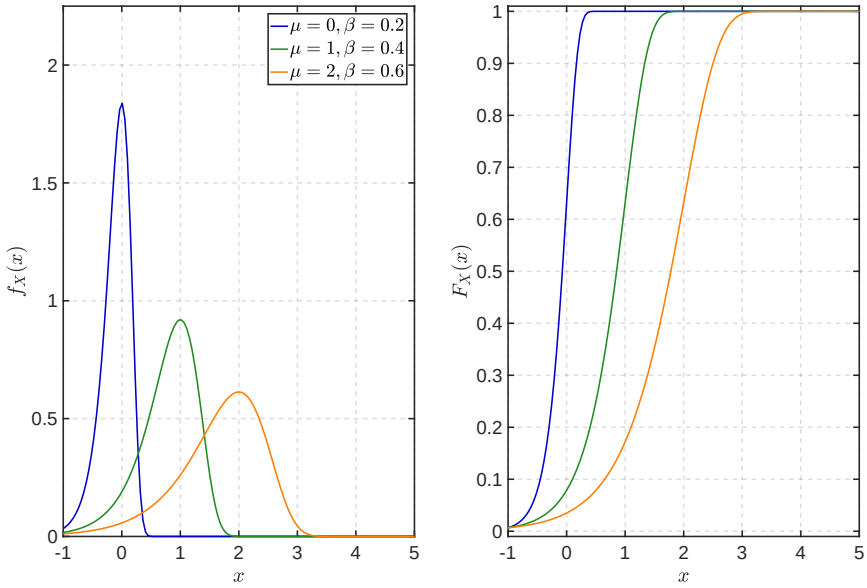

Figure 17: PDF and CDF of Gumbel-min extreme value distributions for various parameter values.

A.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).

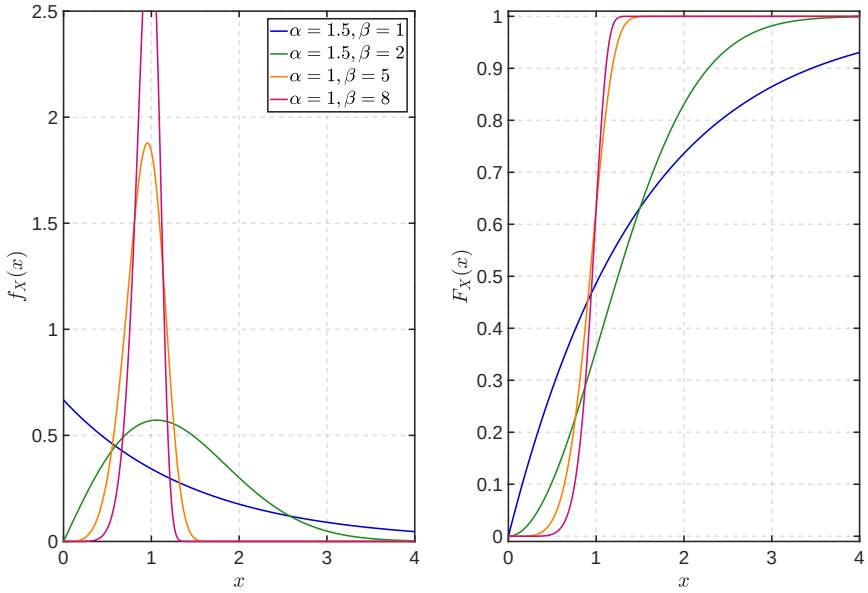

Figure 18: PDF and CDF of Weibull distributions for various parameter values.

A.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 $\Gamma(\lambda, k)$.

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

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

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt \tag{5}$$

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

$$\gamma(k, x) = \int_0^x t^{k-1} e^{-t} dt \tag{6}$$

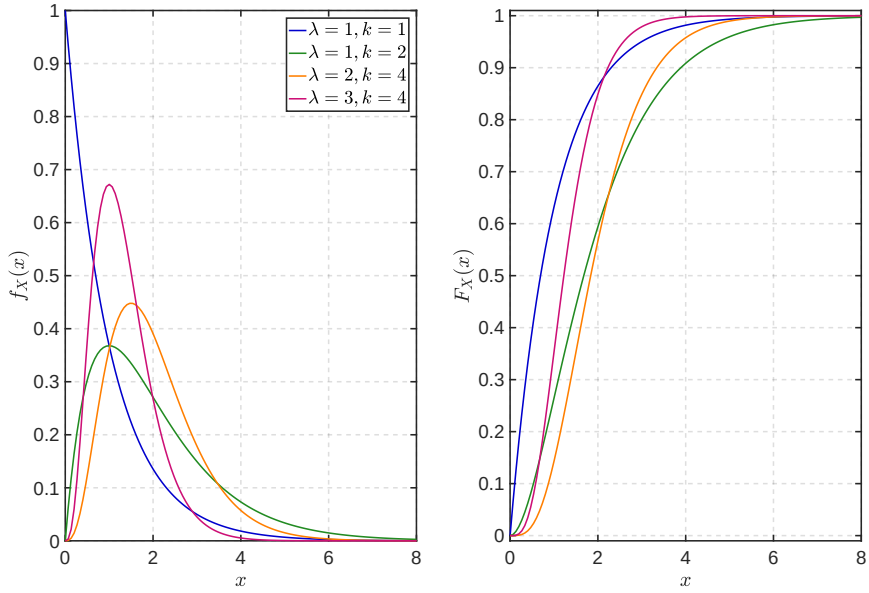

Figure 19: PDF and CDF of Gamma distributions for various parameter values.

A special case of Gamma distribution is $X \sim \Gamma(\lambda, 1)$, which corresponds to the exponential distribution (see Section A.8).

A.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).

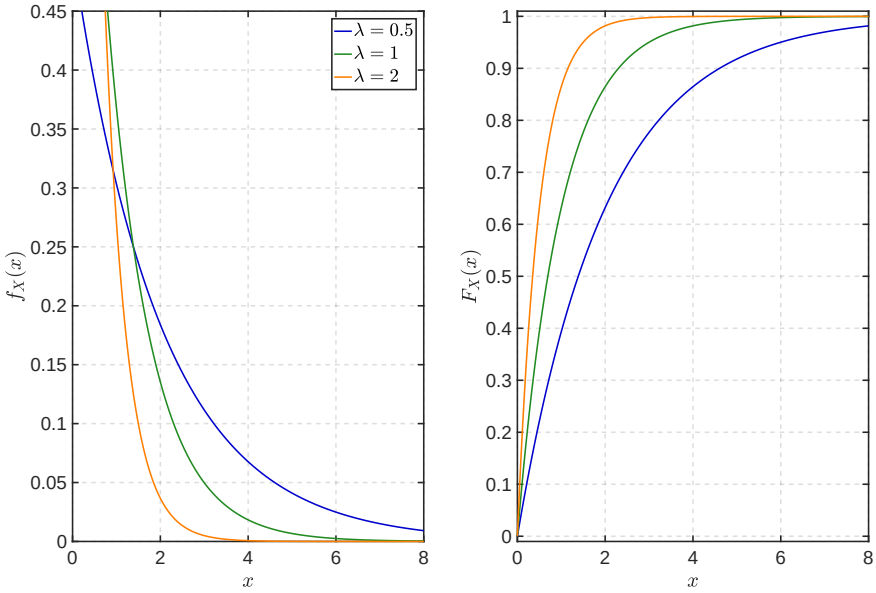

Figure 20: PDF and CDF of exponential distributions for various parameter values.

A.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:

$$B(r,s) = \int_0^1 t^{r-1}(1-t)^{s-1}dt = \frac{\Gamma(r)\Gamma(s)}{\Gamma(r+s)} \tag{7}$$

The support of the distribution 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;

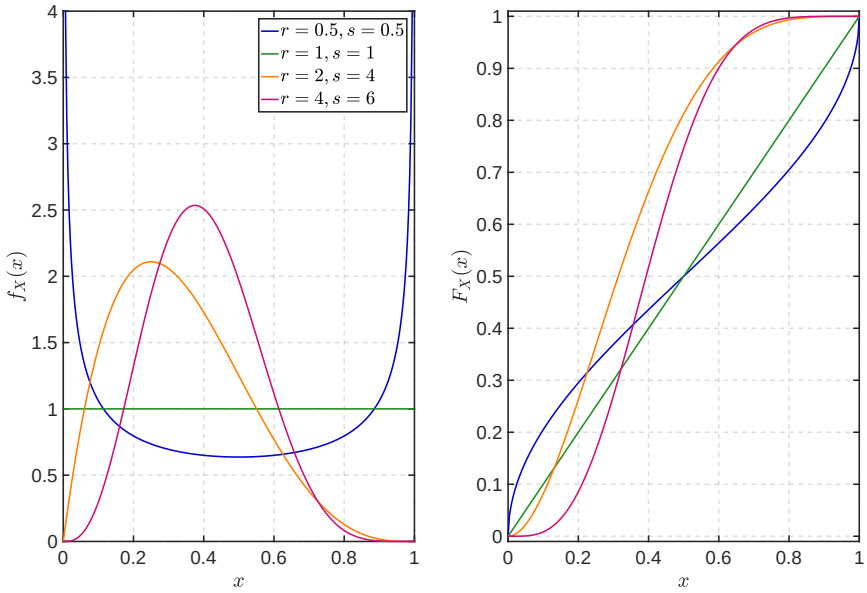

Figure 21: PDF and CDF of Beta distributions for various parameter values ($\mathcal{D}_X = [0, 1]$).

- if $r, s < 1$ the PDF is maximum at the boundaries.

A.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 a and maximum b and an "inspired guess" c of the modal value.

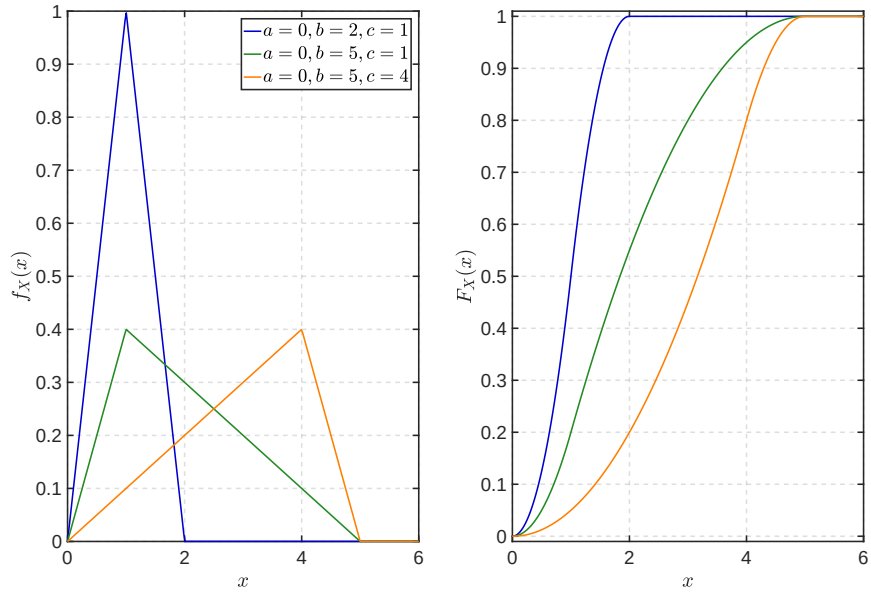

Figure 22: PDF and CDF of triangular distributions for various parameter values.

A.11 Logistic distribution

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

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

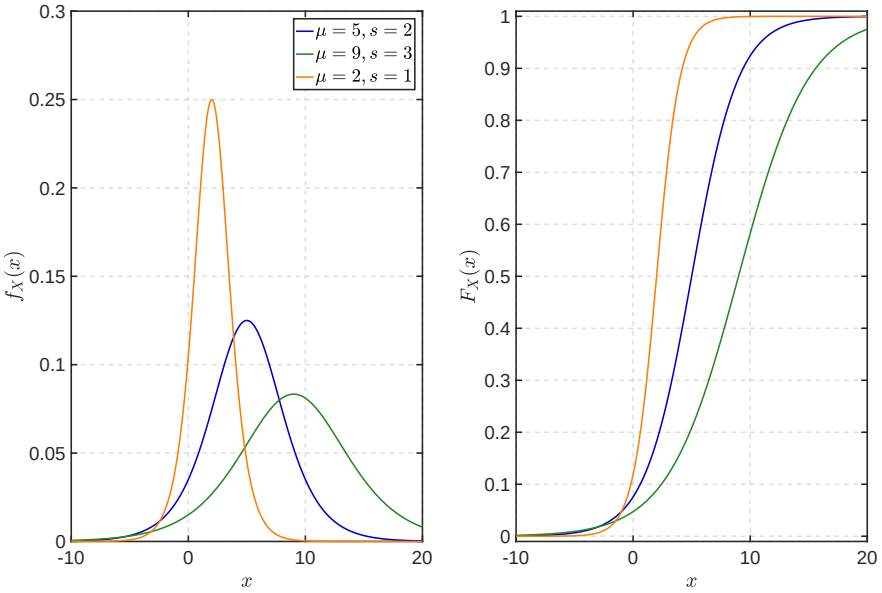

Figure 23: PDF and CDF of logistic distributions for various parameter values.

A.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.

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

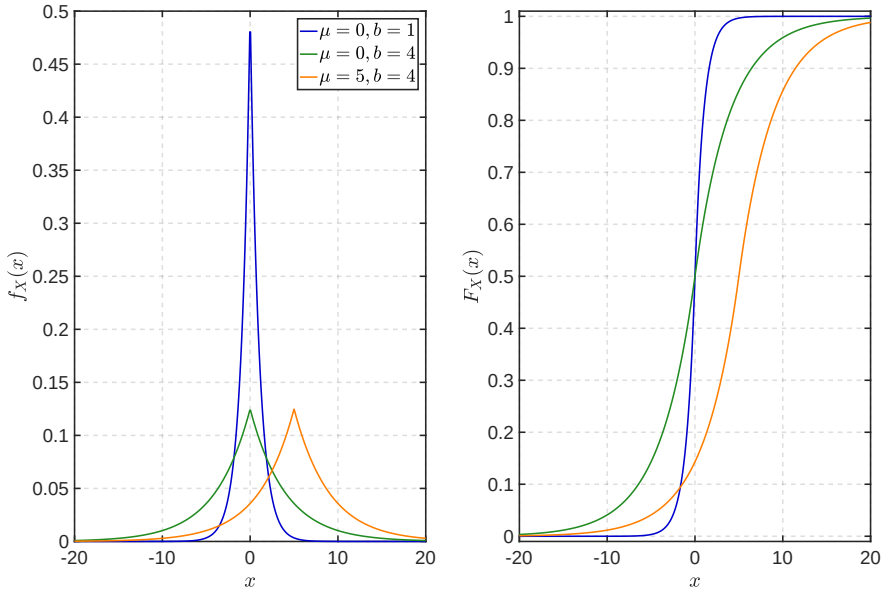

Figure 24: PDF and CDF of Laplace distributions for various parameter values.

A.13 Rayleigh distribution

A spacial case of the Weibull distribution, the Rayleigh distribution is commonly used to describe the envelope of the channel response of a radio signal (a.k.a. Rayleigh fading). Besides, it is also widely used in engineering to model the wind speed.

$$
\begin{aligned}
&\text{Notation} && : X \sim \mathcal{R}(\sigma) \\
&\text{Parameters} && : \sigma > 0 \\
&\text{Support} && : \mathcal{D}_X = [0, +\infty) \\
&\text{PDF} && : f_X(x) = \frac{x}{\sigma^2} e^{-x^2/\left(2\sigma^2\right)} \\
&\text{CDF} && : F_X(x) = 1 - e^{-x^2/\left(2\sigma^2\right)} \\
&\text{Moments} && : \mu_X = \sqrt{\frac{\pi}{2}}\sigma \\
& && \quad \sigma_X = \sqrt{\frac{4-\pi}{2}}\sigma
\end{aligned}
$$

where $\sigma > 0$ is the scale parameter.

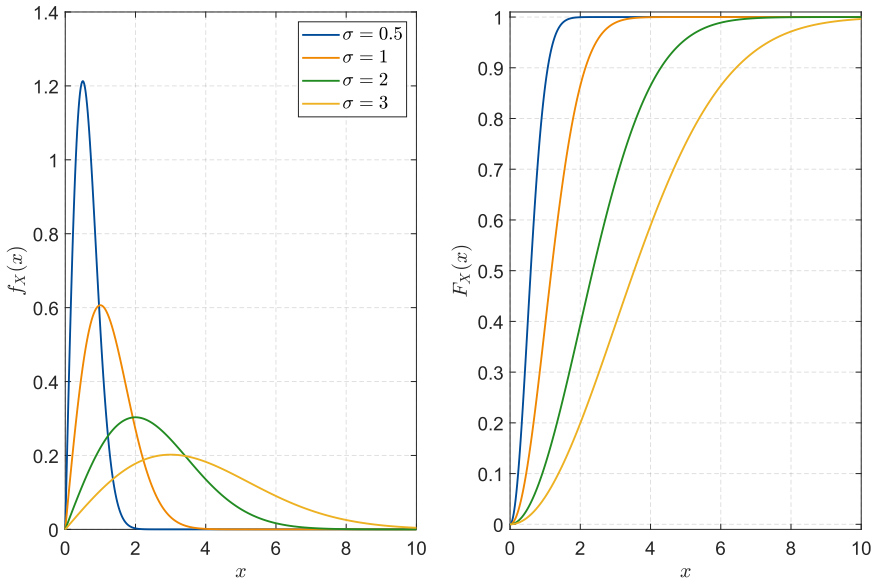

Figure 25: PDF and CDF of Rayleigh distributions for various parameter values.

References

Aas, K., C. Czado, A. Frigessi, and H. Bakkend (2009). Pair-copula constructions of multiple dependence. *Insurance: Mathematics and Economics 44*(2), 182–198. 6, 12

Bedford, T. and R. M. Cooke (2002). Vines – a new graphical model for dependent random variables. *The Annals of Statistics 30*(4), 1031–1068. 6, 8

Joe, H. (1996). Families of m-variate distributions with given margins and $m(m-1)/2$ bivariate dependence parameters. In L. Rüschendorf, B. Schweizer, and M. D. Taylor (Eds.), *Distributions with fixed marginals and related topics*, Volume 28 of *Lecture Notes–Monograph Series*, pp. 120–141. Institute of Mathematical Statistics. 8

Joe, H. (2015). *Dependence modeling with copulas*. CRC Press. 3

Lebrun, R. and A. Dutfoy (2009). A generalization of the Nataf transformation to distributions with elliptical copula. *Probabilistic Engineering Mechanics 24*(2), 172–178. 10

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. 24

Nelsen, R. B. (2006). *An Introduction to Copulas*. Secaucus, NJ, USA: Springer-Verlag New York, Inc. 1, 3, 5

Rosenblatt, M. (1952). Remarks on a multivariate transformation. *The Annals of Mathematical Statistics 23*, 470–472. 12

Saporta, G. (2006). *Probabilités, analyse des données et statistique*. Editions Technip. 52

Torre, E., S. Marelli, P. Embrechts, and B. Sudret (2019). A general framework for data-driven uncertainty quantification under complex input dependencies using vine copulas. *Probabilistic Engineering Mechanics 55*, 1–16. 7