Documentation of the ERA Distribution Classes

Engineering Risk Analysis Group, Technische Universität München.
Arcisstr. 21, 80333 Munich, Germany.

Contributors (in alphabetical order):
Max Ehre, Sebastian Geyer, Antonios Kamariotis, Iason Papaioannou, Luca Sardi,
Daniel Straub, Felipe Uribe

January 2022

1 Introduction

This document presents four MATLAB® classes for defining joint probability distributions and performing selected operations on these distributions. The classes were developed in view of applications to reliability analysis and uncertainty quantification, but can also be used in other contexts.

The first class, termed ERADist, defines the marginal distributons among a selected set of probability distribution types. The second class, called `ERANataf` (section 4), defines the joint distribution through a Gaussian copula by means of the Nataf isoprobabilistic transformation. The third class, called `ERACond` (section 5), allows the definition of conditional distribution objects for the use within the scope of the fourth class, which is called `ERARosen`. `ERARosen` (section 6) is based on the Rosenblatt transform and enables the definition of joint distributions by combining different marginal and conditional distributions. Both, `ERANataf` and `ERARosen`, allow the transformation of the joint distribution from and to standard normal space.

This documentation is not meant to provide an exhaustive description of the implemented MATLAB® classes and methods. Instead it presents briefly the motivation, theoretical background and their definition. Together with the provided MATLAB® example scripts, the user should get a basic understanding on how the different classes can be used. To get more information about a specific class or method it is possible to call the commands `help class` and `help class\method` within MATLAB® , where `class` is one of the four class names (presented in this documentation) and `method` one of the methods of the respective class.
Since all the functionalities were implemented within the object-oriented programming paradigm in the next section the basic principles of this programming paradigm within the MATLAB® framework are explained.

2 Object-oriented programming in MATLAB®

One can usually implement simple numerical methods using built-in functions in MATLAB® . However, with the increase in magnitude and complexity of the tasks, functions become more complex and difficult to operate. Object-oriented techniques can simplify the programming of numerical methods that may involve specialized data structures or a large number of functions that interact between the different data structures.

Object-oriented programming not only allows the definition of a certain data structure, but also the definition of different types of operations (functions) that can be applied to the data. In this way, the data structure becomes an object that includes both data and functions. In

particular, a *class* is a template that describes objects with common characteristics; one must identify and establish all the objects that are manipulated and how they relate to each other.

MATLAB® admits the creation of programs using objects and ordinary functions [9]. In general, a MATLAB® class (command `classdef`), should contain the following main blocks:

- `properties` block: Defines all the properties that are associated with a class. Attributes and default values of all properties are defined.
- `methods` block: Defines functions associated with the class and their attributes. The first method, called the constructor, must have the same name as the class itself .

3 The `ERADist` Class

3.1 Motivation

The specification of probability distributions is paramount in statistical and uncertainty analysis. This task is not only essential for the definition of probability density functions (PDFs) and cumulative distribution functions (CDFs), but also for the generation of samples and computation of statistics of quantities of interest. `ERADist` is the basic class for defining marginal distributions, which is the basic for defining joint distributions with the other classes described later.

The *Statistics and Machine Learning Toolbox* [8] by MATLAB® provides an environment for statistical computations. However, it does not provide much support with operations on joint distributions, which are central to reliability analysis and uncertainty quantification. Additionally, some basic functionalities, which make these tasks easier, are also not provided by The *Statistics and Machine Learning Toolbox*. For example, marginal distributions have to be defined by their parameters, but often it is rather the the moments of the distribution that are provided for its characterization. Consequently, the `ERADist` class has been designed with the purpose of generating several types of probability distribution that can be defined either by their parameters, moments or experimental data to which the distribution is fitted. Furthermore, the different methods of the `ERADist` class are used as auxiliary functions for different methods of the `ERANataf` and `ERARosen` class.

3.2 Definition

3.2.1 'Properties' block

The properties block contains three elements:

- `Name`: specifies the distribution type
- `Par`: specifies the parameters of the distribution
- `ID`: name by which the distribution can be identified

3.2.2 'Methods' block

The methods block is defined by seven functions:

- `Obj = ERADist(name,opt,val)` constructor method to define the object
- `Mean = mean(Obj)` returns the mean value
- `Standarddeviation = std(Obj)` returns the standard deviation
- `CDF = cdf(Obj,x)` returns the value of the CDF at `x`
- `InverseCDF = icdf(Obj,y)` returns the value of the inverse CDF at `y`
- `PDF = pdf(Obj,x)` returns the value of the PDF at `x`
- `Random = random(Obj,m,n)` generates $m \times n$ random samples

To get a better understanding on how an `ERADist` object can be defined, in the following the different inputs of the constructor method are explained.
The input argument `name` corresponds to the type of probability distribution that the user wants to create. The user can choose from 21 supported probability distributions that are listed in Table 1. To get more information about the supported distributions and their properties please refer to the appendix A.
The input argument `opt` can be chosen as:

- `PAR`: If the distribution should be defined by its parameters (Table 2).
- `MOM`: If the distribution should be defined by its first moments (Table 3).
- `DATA`: If the distribution should be defined by fitting the parameters to a given data set (Table 4).

The argument `val` requires the numeric value of the given `opt`, i.e., the value of the parameters if `PAR` is chosen, the value of the moments if `MOM` is chosen, or the data vector if `DATA` is chosen. In the case that the user wants to use the `ERADist` object as part of an `ERARosen` object the additional input argument `id` can be given. In this case the `ERADist` object must be defined as follows:

```
Obj = ERADist(name,opt,val,id)
```

The scope of the additional input is to be able to identify the corresponding node when plotting the Bayesian network which defines the `ERARosen` object with the method `plotGraph`. The input `id` must thereby be given as a character array.
In order to facilitate the definition of the `ERADist` class, the inputs of `name` and `opt` are case insensitive.

Table 1: `ERADist` distribution names.

`name`	Distribution
'`beta`'	Beta distribution
'`binomial`'	Binomial distribution
'`chisquare`'	Chi-Squared distribution
'`exponential`'	Exponential distribution
'`frechet`'	Frechet distribution
'`gamma`'	Gamma distribution
'`geometric`'	Geometric distribution
'`GEV`'	Generalized extreme value distribution (to model maxima)
'`GEVMin`'	Generalized extreme value distribution (to model minima)
'`gumbel`'	Gumbel distribution (to model maxima)
'`gumbelMin`'	Gumbel distribution (to model minima)
'`lognormal`'	Log-Normal distribution
'`negativebinomial`'	Negative binomial distribution
'`normal`'	Normal distribution
'`pareto`'	Pareto distribution
'`poisson`'	Poisson distribution
'`rayleigh`'	Rayleigh distribution
'`standardnormal`'	Standard normal distribution
'`truncatednormal`'	Truncated normal distribution
'`uniform`'	Uniform distribution
'`weibull`'	Weibull distribution

Table 2: Definition of the distributions by their parameters (`PAR`).

Distribution	Defined by parameters
Beta	`Obj = ERADist('beta','PAR',[r,s,lower,upper])`
Binomial	`Obj = ERADist('binomial','PAR',[n,p])`
Chi-squared	`Obj = ERADist('chisquare','PAR',[k])`
Exponential	`Obj = ERADist('exponential','PAR',[lambda])`
Fréchet	`Obj = ERADist('frechet','PAR',[a_n,k])`
Gamma	`Obj = ERADist('gamma','PAR',[lambda,k])`
Geometric	`Obj = ERADist('geometric','PAR',[p])`
GEV (to model maxima)	`Obj = ERADist('GEV','PAR',[beta,alpha,epsilon])`
GEV (to model minima)	`Obj = ERADist('GEVMin','PAR',[beta,alpha,epsilon])`
Gumbel (to model maxima)	`Obj = ERADist('gumbel','PAR',[a_n,b_n])`
Gumbel (to model minima)	`Obj = ERADist('gumbelMin','PAR',[a_n,b_n])`
Log-normal	`Obj = ERADist('lognormal','PAR',[mu_lnx,sig_lnx])`
Negative binomial	`Obj = ERADist('negativebinomial','PAR',[k,p])`
Normal	`Obj = ERADist('normal','PAR',[mean,std])`
Pareto	`Obj = ERADist('pareto','PAR',[x_m,alpha])`
Poisson	`Obj = ERADist('poisson','PAR',[v,t])` `Obj = ERADist('poisson','PAR',[lambda])`
Rayleigh	`Obj = ERADist('rayleigh','PAR',[alpha])`
Standard normal	`Obj = ERADist('standardnormal','PAR',[])`
Truncated normal	`Obj = ERADist('truncatednormal','PAR',[mu_n,sig_n,a,b])`
Uniform	`Obj = ERADist('uniform','PAR',[lower,upper])`
Weibull	`Obj = ERADist('weibull','PAR',[a_n,k])`

Table 3: Definition of the distributions by their moments (`MOM`).

Distribution	Defined by moments
Beta	`Obj = ERADist('beta','MOM',[mean,std,lower,upper])`
Binomial	`Obj = ERADist('binomial','MOM',[mean,std])`
Chi-squared	`Obj = ERADist('chisquare','MOM',[mean])`
Exponential	`Obj = ERADist('exponential','MOM',[mean])`
Fréchet	`Obj = ERADist('frechet','MOM',[mean,std])`
Gamma	`Obj = ERADist('gamma','MOM',[mean,std])`
Geometric	`Obj = ERADist('geometric','MOM',[mean])`
GEV (to model maxima)	`Obj = ERADist('GEV','MOM',[mean,std,beta])`
GEV (to model minima)	`Obj = ERADist('GEVMin','MOM',[mean,std,beta])`
Gumbel (to model maxima)	`Obj = ERADist('gumbel','MOM',[mean,std])`
Gumbel (to model minima)	`Obj = ERADist('gumbelMin','MOM',[mean,std])`
Log-normal	`Obj = ERADist('lognormal','MOM',[mean,std])`
Negative binomial	`Obj = ERADist('negativebinomial','MOM',[mean,std])`
Normal	`Obj = ERADist('normal','MOM',[mean,std])`
Pareto	`Obj = ERADist('pareto','MOM',[mean,std])`
Poisson	`Obj = ERADist('poisson','MOM',[mean,t])` `Obj = ERADist('poisson','MOM',[mean])`
Rayleigh	`Obj = ERADist('rayleigh','MOM',[mean])`
Standard normal	`Obj = ERADist('standardnormal','MOM',[])`
Truncated normal	`Obj = ERADist('truncatednormal','MOM',[mean,std,a,b])`
Uniform	`Obj = ERADist('uniform','MOM',[lower,upper])`
Weibull	`Obj = ERADist('weibull','MOM',[mean,std])`

Table 4: Definition of the distributions by data (`DATA`).

Distribution	Defined by data
Beta	`Obj = ERADist('beta','DATA',{[X],[lower,upper]})`
Binomial	`Obj = ERADist('binomial','DATA',{[X],n})`
Chi-squared	`Obj = ERADist('chisquare','DATA',[X])`
Exponential	`Obj = ERADist('exponential','DATA',[X])`
Fréchet	`Obj = ERADist('frechet','DATA',[X])`
Gamma	`Obj = ERADist('gamma','DATA',[X])`
Geometric	`Obj = ERADist('geometric','DATA',[X])`
GEV (to model maxima)	`Obj = ERADist('GEV','DATA',[X])`
GEV (to model minima)	`Obj = ERADist('GEVMin','DATA',[X])`
Gumbel (to model maxima)	`Obj = ERADist('gumbel','DATA',[X])`
Gumbel (to model minima)	`Obj = ERADist('gumbelMin','DATA',[X])`
Log-normal	`Obj = ERADist('lognormal','DATA',[X])`
Negative binomial	`Obj = ERADist('negativebinomial','DATA',[X])`
Normal	`Obj = ERADist('normal','DATA',[X])`
Pareto	`Obj = ERADist('pareto','DATA',[X])`
Poisson	`Obj = ERADist('poisson','DATA',{[X],t})` `Obj = ERADist('poisson','DATA',[X])`
Rayleigh	`Obj = ERADist('rayleigh','DATA',[X])`
Truncated normal	`Obj = ERADist('truncatednormal','DATA',{[X],[a,b]})`
Uniform	`Obj = ERADist('uniform','DATA',[X])`
Weibull	`Obj = ERADist('weibull','DATA',[X])`

4 The `ERANataf` Class

4.1 Motivation and Theoretical Background

For several methods in probabilistic assessment and reliability analysis, it is beneficial to transform the basic random variables to an equivalent independent standard normal random variable space. This allows to take advantage of the numerous properties of the normal distribution. Depending on the information available on the distributions of random vectors, there exist several

possible transformations from the space of physical variables to the standardized variable space, e.g. Nataf transformation, Rosenblatt transformation (see section 6), Hermite polynomials transformation, among others [4].

The Nataf transformation [10] describes the joint probability density function of random variables based on their individual marginal distributions and coefficients of correlation using a Gaussian copula. The basic assumption of the underlying distribution model is that the random variables derived from a marginal transformation to standard normal variables will follow the multivariate normal distribution.

Consider a set of n correlated random variables $\mathbf{X} = [X_1, ..., X_n]$ with known marginal CDFs $F_{X_1}(x_1), ..., F_{X_n}(x_n)$ and linear correlation matrix $\mathbf{R}_0$ with components,

$$\rho'_{ij} = \mathrm{E}\left[\left(\frac{X_i - \mu_{X_i}}{\sigma_{X_i}}\right)\left(\frac{X_j - \mu_{X_j}}{\sigma_{X_j}}\right)\right],$$

where μ_{X_i} and σ_{X_i} are the mean and standard deviation of X_i. The *Nataf transformation* $T : \mathbf{X} \to \mathbf{U}$ is the composition of two functions $T = T_2 \circ T_1$,

- First, an associated set of correlated standard normal random variables $\hat{\mathbf{U}} = [\hat{U}_1, ..., \hat{U}_n]$ (with correlation matrix $\mathbf{R}_0$) can be obtained by applying:

  $$T_1 : X_i \longrightarrow \hat{U}_i = \Phi^{-1}\left(F_{X_i}(x_i)\right) \qquad i = 1, ..., n;$$

  where, Φ is the standard normal CDF.

- Second, a set of uncorrelated variables $\mathbf{U} = [U_1, ..., U_n]$ is subsequently obtained by applying:

  $$T_2 : \hat{\mathbf{U}} \longrightarrow \mathbf{U} = \boldsymbol{\Gamma}\hat{\mathbf{U}}$$

  where, $\boldsymbol{\Gamma}$ is the square root of the inverse of the correlation matrix $\mathbf{R}_0$, e.g. a Cholesky factor of $\mathbf{R}_0^{-1}$. The correlation matrix $\mathbf{R}_0$ is called the fictive correlation matrix. In general $\mathbf{R}_0 \neq \mathbf{R}$, but they are related as:

  $$\begin{aligned}\rho_{ij} &= \mathrm{E}\left[\left(\frac{F_{X_i}^{-1}\left(\Phi(\hat{u}_i)\right) - \mu_{X_i}}{\sigma_{X_i}}\right)\left(\frac{F_{X_j}^{-1}\left(\Phi(\hat{u}_j)\right) - \mu_{X_j}}{\sigma_{X_j}}\right)\right] \\ &= \frac{1}{\sigma_{X_i}\sigma_{X_j}}\int\int\left(F_{X_i}^{-1}\left(\Phi(\hat{u}_i)\right) - \mu_{X_i}\right)\left(F_{X_j}^{-1}\left(\Phi(\hat{u}_j)\right) - \mu_{X_j}\right)\phi_2\left(\hat{u}_i, \hat{u}_j; \rho'_{ij}\right)\mathrm{d}\hat{u}_i\mathrm{d}\hat{u}_j\end{aligned}$$

  where, ϕ_2 is the two-dimensional standard normal PDF with zero means, unit standard deviations and correlation coefficient ρ'_{ij}.

  The computation of the coefficients ρ'_{ij} may be carried out using iterative procedures. However, it might be difficult since it involves the resolution of an integral equation (which is not guaranteed to have a solution, if ρ_{ij} is too close to 1 or -1), and even if each coefficient ρ'_{ij} can be computed, there is no guarantee that the resulting matrix will be positive definite [3]. When this happens, a set of semi-empirical formulas relating ρ_{ij} and ρ'_{ij} based on numerical studies for various types of random variables can be used [4]. In the `ERANataf` class the computation of the fictive correlation matrix (i.e. solution of the integral equation) is carried out numerically by a two-dimensional Gauss-Legendre quadrature integration, together with the MATLAB® `fzero` command in order to solve for the correlation coefficients.

After applying the Nataf transformation the joint PDF and CDF of the random vector $\mathbf{X}$ can be computed as [4],

$$f_{\mathbf{X}}(\mathbf{x}) = \phi_n\left(\hat{\mathbf{u}}; \mathbf{R}_0\right)\frac{f_{X_1}(x_1)f_{X_2}(x_2)\cdots f_{X_n}(x_n)}{\phi(\hat{u}_1)\phi(\hat{u}_2)\cdots\phi(\hat{u}_n)} = \phi_n\left(\hat{\mathbf{u}}; \mathbf{R}_0\right)\prod_{i=1}^{n}\frac{f_{X_i}(x_i)}{\phi(\hat{u}_i)}$$

$$F_{\mathbf{X}}(\mathbf{x}) = \Phi(\hat{\mathbf{u}}; \mathbf{R}_0)$$

The `ERANataf` class implements not only the Nataf transformation to map a given set of samples from the physical to the standard normal space (`X2U`), but also the inverse Nataf transformation to map the samples from the standard normal space to the physical space (`U2X`) and the Jacobians of the two transformations. Moreover, one can compute the joint PDF and CDF of a given set of correlated random variables, or generate random samples from the joint distribution.

4.2 Definition

4.2.1 'Properties' block

The properties block contains four elements:

- `Rho_X`: correlation matrix of the random vector $\mathbf{X}$
- `Rho_Z`: correlation matrix of the correlated normal vector $\hat{\mathbf{U}}$
- `A`: lower triangular matrix of the Cholesky decomposition of `Rho_Z`
- `Marginals`: contains all marginal distribution objects

4.2.2 'Methods' block

The methods block is defined by six functions:

- `Obj = ERANataf(M,Correlation)` constructor method to define the object
- `[U,Jac] = X2U(Nataf,X,opt)` transformation from `X` to `U` and Jacobian of this transformation
- `[X,Jac] = U2X(Nataf,U,opt)` transformation from `U` to `X` and Jacobian of this transformation
- `jointrandom = random(Nataf,n)` generates `n` random samples according to the joint distribution
- `jointpdf = pdf(Nataf,X)` returns the joint PDF at `X`
- `jointcdf = cdf(Nataf,X)` returns the joint CDF at `X`

To define the `ERANataf` object, the marginals, defined by a vector-shaped `ERADist` object, must be given as input argument `M` of the constructor method. A vector-shaped `ERADist` object can be created by stacking the single `ERADist` objects in a matrix (e.g., [dist_1,...,dist_n]). The dependence between the different marginals must be defined by a $n \times n$ correlation matrix and given as input argument `Correlation` of the constructor method.
For the other methods one must consider that every row of `X` and `U` corresponds to a single data point while the number of columns corresponds to the dimension of the joint distribution. Moreover note, the output `Jac` of the methods `X2U` and `U2X` is only given if the respective input `opt` is given as character array `'Jac'`. If that is the case, the output `Jac` corresponds to the Jacobian of the transformation of the first sample in `X` and `U`.

5 The ERACond Class

5.1 Motivation

A way to describe dependence between different random variables is to define conditional probability distributions. The density of the variable Y conditional on the variable X could be written as:

$$f_{X|Y}(x|y) = \frac{f_{X,Y}(x,y)}{f_X(x)}$$

where $f_{X,Y}(x,y)$ represents the joint PDF of the variables X and Y. It is not always possible to directly define the joint distribution of two or more random variables. However, if different marginal and conditional distributions are known, the combination of these can be used to define the joint distribution. This approach, which is different to the one represented by the Nataf distribution model (see section 4), is implemented in the `ERARosen` class (see section 6). To enable this approach, the `ERACond` class is defined, which allows the construction of conditional distributions.

5.2 Definition

5.2.1 'Properties' block

The properties block contains five elements:

- `Name`: specifies the distribution type
- `Opt`: type of parameter which defines the distribution, either 'MOM' or 'PAR'
- `Param`: specifies the parameters of the distribution according to the property `Opt` as a function handle
- `ID`: name by which the distribution can be identified

5.2.2 'Methods' block

Although the `ERACond` class contains different methods, in the following only the constructor method will be explained. The other methods are only meant to be called by the different `ERARosen` methods. The direct use is nonetheless possible, but in general not recommended and therefore not explained.
An `ERACond` object can be defined by the constructor method as follows:

```
Obj = ERACond(name,opt,param)
```

The input argument `name` corresponds to the type of probability distribution that the user wants to create. The user can choose from a total of 21 supported probability distributions that are listed in Table 1. To get more information about the supported distributions and their properties please refer to the appendix A.
The `ERACond` class, unlike the `ERADist` class, accepts only two options for the input argument `opt` , namely:

- `PAR`: If the distribution should be defined by its parameters (see Table 2 for orientation).

- `MOM`: If the distribution should be defined by its first moments (see Table 3 for orientation).

For the input argument `param` the user must give the required information, parameters or first moments according to the choice given in `opt`, as follows:

- as a cell array of function handles (and scalars) for distributions with multiple parameters.
- as a function handle for distributions with one parameter. A cell array of a single function handle also works for distributions with one parameter.

The number of variables of the function handle corresponds to the number of distributions (other `ERAcond` or `ERADist` objects) that the conditional distribution is depending on. The distributions on which the conditional distribution depends are also referred to as parents. An example for a cell array of function handles given as input `param` for distributions that have two parameters and three parents could be:

```
param = {@(x,y,z) x+y+z, 0.2*x^2}
```

An example for the required input `param` for distributions that have one parameter and two parents could be:

```
param = {@(x,y) x^2-y} OR param = @(x,y) x^2-y
```

Note, in case that one of the parameters or moments does not depend on any parents the respective matrix entry of the function handle can be simply given by a constant term.

In case the user wants to use the method `plotGraph` within the `ERARosen` class in order to visualize the Bayesian network which defines the `ERARosen` object, it can be beneficial to give the additional input argument `id` to the different `ERACond` objects that define the `ERARosen` object. In this case ,the `ERACond` objects must be defined as follows:

```
Obj = ERACond(name,opt,param,id)
```

The reason for the additional input is to be able to identify the different nodes when plotting the Bayesian network. The input `id` must thereby be given as a character array.

6 The `ERARosen` Class

It should be noted that this class is not working for MATLAB® releases older than `MATLAB R2019a`.

6.1 Motivation and Theoretical Background

As already mentioned in section 5.1, there are different ways to define joint probability distributions. The approach underlying the `ERARosen` class is the definition of a joint probability distribution by the combination of different marginal and conditional distributions through the Rosenblatt transform. In general, the joint PDF of a random vector $\mathbf{X}$ can be written through the chain rule as

$$f_{\mathbf{X}}(\mathbf{x}) = f_{X_1}(x_1) \cdot f_{X_2|X_1}(x_2|X_1 = x_1) \cdot ... \cdot f_{X_n|X_1,...,X_{n-1}}(x_n|X_1 = x_1, ..., X_{n-1} = x_{n-1})$$

Due to the successive conditioning of the different random variables, a transformation of the variables from physical to standard normal space $T : \mathbf{X} \rightarrow \mathbf{U}$ can be written as follows:

$$
\begin{aligned}
u_1 &= \Phi^{-1}(F_{X_1}(x_1)) \\
u_2 &= \Phi^{-1}(F_{X_2}(x_2|X_1 = x_1)) \\
&\vdots \\
u_n &= \Phi^{-1}(F_{X_n}(x_n|X_1 = x_1, ..., X_{n-1} = x_{n-1}))
\end{aligned}
$$

This transformation is known as Rosenblatt transformation [1,2,11]. Its inverse, the transformation $T : \mathbf{U} \rightarrow \mathbf{X}$, can be written as:

$$
\begin{aligned}
x_1 &= F_{X_1}^{-1}(\Phi(u_1)) \\
x_2 &= F_{X_2}^{-1}(\Phi(u_2)|X_1 = x_1) \\
&\vdots \\
x_n &= F_{X_n}^{-1}(\Phi(u_n)|X_1 = x_1, ..., X_{n-1} = x_{n-1})
\end{aligned}
$$

In order to facilitate the understanding, definition and visualization of the joint PDF with the `ERARosen` class, it is useful to link the Rosenblatt transformation to the concept of Bayesian networks. A Bayesian network (BN) is a probabilistic modeling tool that has originated from the field of artificial intelligence and machine learning [12]. A BN is defined by a Directed Acyclic Graph (DAG) structure and its graphical representation makes it easy to intuitively define and understand the dependency between random variables. Every marginal and conditional distribution corresponds to a node within the graph. The BN encodes conditional independence assumptions among the random variables. Specifically, from the d-separation properties underlying BNs it follows that the chain rule speficying the joint PDF can be reduced to:

$$
f_{\mathbf{X}}(\mathbf{x}) = \prod_{i=1}^{n} f_{X_i}(x_i|pa_{X_i}(x_i))
$$

where $f_{X_i}(x_i|pa_{X_i}(x_i))$ corresponds to the conditional PDF of X_i given realizations of the parents of X_i, denoted as $pa_{X_i}(x_i)$. Parents of X_i are all nodes that have links pointing towards X_i.

For the example shown in Figure 1 the joint PDF can therefore be written as:

$$
\begin{aligned}
f_{\mathbf{X}}(\mathbf{x}) =& f_A(a) \cdot f_B(b) \cdot f_{C|A,B}(c|A = a, B = b) \cdot f_{D|A,C}(d|A = a, C = c) \cdot \\
& \cdot f_{E|A,C}(e|C = c, B = b) \cdot f_{F|D}(f|D = d) \cdot f_{G|D,C,E}(g|D = d, C = c, E = e)
\end{aligned}
$$

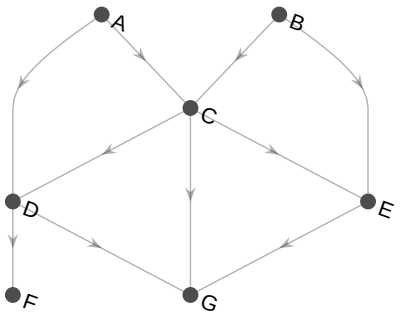

Figure 1: Example of a Directed Acyclic Graph

The variables A and B are defined as marginal distributions and therefore must be created as `ERADist` objects while all the other variables are defined as conditional distributions, hence must be created with `ERACond`.

Additional to the joint PDF and the two transformations, a sequential generation of multivariate random samples is implemented in the `ERARosen` class.

Since it is not always possible to accurately evaluate the integral for the joint CDF, no method is implemented for this purpose. However, the user can make use of the MATLAB® built-in kernel density estimation functions `mvksdensity` and `ksdensity`. More information regarding the use of this functions can be found in [6,7] or in the provided MATLAB® script `Example_Kernel_Density_Estimation.m`.

6.2 Definition

6.2.1 'Properties' block

The properties block contains four elements:

- `Dist`: cell array containing all the different marginal and conditional distributions
- `Parents`: cell array containing the indices of the parent nodes
- `Layers`: cell array containing information about the computation order
- `Adjacency`: adjacency matrix of the Directed Acyclic Graph which defines the dependency between the different distributions

6.2.2 'Methods' block

The methods block is defined by six functions:

- `Obj = ERARosen(dist,depend)` constructor method to define the object
- `U = X2U(Obj,X,opt)` transformation from `X` to `U`
- `X = U2X(Obj,U,opt)` transformation from `U` to `X`
- `jointpdf = pdf(Obj,X,opt)` returns the joint PDF at `X`
- `jointrandom = random(Obj,n)` generates `n` random samples according to the joint distribution
- `fig = plotGraph(Obj,opt)` plots the Bayesian network by which the joint distribution is defined

The input arguments of the constructor methods are defined as follows:

- `dist` must be a cell array with vector shape that contains all the marginal distributions (`ERADist` objects) and conditional distributions (`ERACond` objects) required to define the joint distribution. To create the input `dist` all the different distribution objects must be defined separately and then be stacked in a cell array (e.g., `dist = {dist_1,dist_2, ... , dist_n}`). The use of cell arrays instead of matrix objects is necessary since it is not

possible to put objects of different classes, in this case `ERADist` and `ERACond` objects, in one matrix.

- `depend` must describe the dependency between the different marginal and conditional distributions. The dependency is defined by collecting vector shaped matrices which contain the indices of the parents of the respective distributions in a cell array. The matrices within the cell array must be ordered according to the place of the corresponding distribution in the input `dist`. If a distribution is defined as a marginal distribution, and therefore has no parents, an empty matrix (`[]`) must be given for that distribution in `depend`. For conditional distributions, the order of the indices within one of the matrices corresponds to the order of the variables of the respective function handle of the according `ERACond` object. The described dependency must always lead to a Directed Acyclic Graph (DAG) structure.

For the the methods `X2U`, `U2X` and `pdf` the following apply:

- `X` must be a $n \times d$ matrix (n = number of data points, d = dimensions).
- If no error message should be given in case of the detection of an improper distribution, give opt as character array `'NaN'`.

When using the method `plotGraph`, if opt is given as `'numbering'` the nodes are named according to their order of input in `dist` (e.g., the third distribution in `dist` is named #3). If no ID was given to a certain distribution, the distribution is also named according to its position in `dist`, otherwise the property ID is taken as the name of the distribution.

7 Use of discrete marginals with `ERANataf` and `ERARosen`

The distributions models that can be used to construct objects of the `ERADist` and `ERACond` classes include a number of discrete distributions. These include the binomial, negative binomial, geometric and Poisson models, see appendix A. Discrete random variables can be used to construct joint probability distributions through application of the `ERANataf` and `ERARosen` classes. The key concept that allows the definition of discrete joint distributions based on the Gaussian copula and the product rule (Rosenblatt transform) is the extension of the definition of the inverse of a function as follows:

$$F^{-1}(u) = \inf\{x \in \mathbb{R} : F(x) \geq u\}$$

The above is known as the *generalized inverse* of the function $F(x)$. Through employing the generalized inverse of the CDF of discrete random variables, it is possible to define a unique transformation $T : \mathbf{U} \to \mathbf{X}$ based on either joint model. However, for a discrete random variable X with CDF F_X it holds that $F_X(F_X^{-1}(u)) \neq u$, which implies that the transform $T : \mathbf{X} \to \mathbf{U}$ is not unique. Therefore, if at least one of the marginal (or conditional) distributions is a discrete distribution, the method `X2U` will return an error.

We note that if the `ERANataf` class includes only discrete marginals, the joint PMF of the random vector $\mathbf{X}$ is given by the expression:

$$f_{\mathbf{X}}(\mathbf{x}) = \Delta_{x_n-1}^{x_n}\Delta_{x_{n-1}-1}^{x_n}\cdots\Delta_{x_1-1}^{x_1}\Phi(\hat{\mathbf{u}}, \mathbf{R}_0)$$

where

$$\Delta_{a_k}^{b_k}G(\mathbf{x}) = G(x_1, \ldots, x_{k-1}, b_k, x_{k+1}, \ldots, x_n) - G(x_1, \ldots, x_{k-1}, a_k, x_{k+1}, \ldots, x_n)$$

For the case where the marginals of the first m components of $\mathbf{X}$ are continuous and the remaining components are discrete, the joint PDF reads:

$$f_{\mathbf{X}}(\mathbf{x}) = \Delta_{x_n-1}^{x_n}\Delta_{x_{n-1}-1}^{x_n}\cdots\Delta_{x_{m+1}-1}^{x_{m+1}}F_{\mathbf{X}_{m+1:n}}(\mathbf{x}_{m+1:n}|\mathbf{x}_{1:m})f_{\mathbf{X}_{1:m}}(\mathbf{x}_{1:m})$$

where $f_{\mathbf{X}_{1:m}}(\mathbf{x}_{1:m})$ is obtained as explained in section 4 and $F_{\mathbf{X}_{m+1:n}}(\mathbf{x}_{m+1:n}|\mathbf{x}_{1:m})$ is the CDF of the random variable $\mathbf{X}_{m+1:n}$ conditional on $\mathbf{X}_{1:m} = \mathbf{x}_{1:m}$. As the accurate implementation of these steps involves considerable computational cost, they are currently not included. Therefore, the `jointpdf` method of the `ERANataf` class will return an error if at least one marginal is a discrete distribution.

8 Application Examples

For more information on the different classes, please refer to the 'Examples' folder. The folder contains three MATLAB® scripts which illustrate the use of the different classes which were presented in this document. An additional script illustrates how to estimate the marginal PDFs and joint CDF of a random vector defined with the `ERARosen` class using kernel density estimation with MATLAB® built-in functions.

References

[1] Ditlevsen, O. and H.O. Madsen (2005) - "Structural Reliability Methods". *Coastal, Maritime and Structural Engineering Department of Mechanical Engineering*, Technical University of Denmark.

[2] Hohenbichler, M. and R. Rackwitz (1981) - "Non-Normal Dependent Variables in Structural Reliability." *Journal of Eng. Mech., ASCE,* 107, 1227-1238.

[3] Lebrun, Régis and Anne Dutfoy (2009) - "An innovating analysis of the Nataf transformation from the copula viewpoint". *Probabilistic Engineering Mechanics* 24(3), 312-320.

[4] Lemaire, Maurice (2009) - "Structural reliability". *Wiley-ISTE* ISBN: 978-1848210820.

[5] Liu, Pei-Ling and Armen Der Kiureghian (1986) - "Multivariate distribution models with prescribed marginals and covariances". *Probabilistic Engineering Mechanics* 1(2), 105-112.

[6] MathWorks (2020) - "ksdensity". www.mathworks.com/help/stats/ksdensity.html.

[7] MathWorks (2020) - "mvksdensity". www.mathworks.com/help/stats/mvksdensity.html.

[8] MathWorks (2020) - "Statistics and Machine Learning Toolbox". www.mathworks.com/help/stats/.

[9] MathWorks (2020) - "Object-Oriented Programming". www.mathworks.com/help/matlab/object-oriented-programming.html.

[10] Nataf, A. (1962) - "Détermination des distributions de probabilités dont les marges sont données". *Comptes rendus de l'Académie des sciences* 225, 42-43.

[11] Rosenblatt, M. (1952) - "Remarks on a multivariate transformation". *Ann. Math. Stat., 23,* 470-472.

[12] Straub, Daniel (2019) - "Lecture notes in Engineering Risk Analysis". *Engineering Risk Analysis Group*, Technische Universität München.

A Supported Probability Distributions

Name	PDF/PMF and CDF	Support	Parameters	Mean	Standard deviation
Beta	$f_X(x) = \frac{(x-a)^{r-1}(b-x)^{s-1}}{B(r,s)(b-a)^{r+s-1}}$ $F_X(x) = I_{\frac{x-a}{b-a}}(r,s)$	$x \in (a,b)$	$r > 0$ $s > 0$ $(a < b) \in \mathbb{R}$	$\frac{as+br}{r+s}$	$\frac{b-a}{r+s}\sqrt{\frac{rs}{r+s+1}}$
Binomial	$p_X(x) = \binom{n}{x} p^x (1-p)^{n-x}$ $F_X(x) = \sum_{i=0}^{x} \binom{n}{i} p^i (1-p)^{n-i}$	$x \in \{0, ..., n\}$	$n \in \mathbb{N}_0$ $p \in [0,1]$	np	$\sqrt{np(1-p)}$
Chi-squared	$f_X(x) = \frac{1}{2^{\frac{k}{2}}\Gamma\left(\frac{k}{2}\right)} x^{(\frac{k}{2}-1)} \exp\left(-\frac{x}{2}\right)$ $F_X(x) = \frac{\gamma\left(\frac{k}{2}, \frac{x}{2}\right)}{\Gamma\left(\frac{k}{2}\right)}$	$x \in [0, \infty)$	$k \in \mathbb{N}_{>0}$	k	$\sqrt{2k}$
Exponential	$f_X(x) = \lambda \exp(-\lambda x)$ $F_X(x) = 1 - \exp(-\lambda x)$	$x \in [0, \infty)$	$\lambda > 0$	$\frac{1}{\lambda}$	$\frac{1}{\lambda}$

Name	PDF/PMF and CDF	Support	Parameters	Mean	Standard deviation
Fréchet	$f_X(x) = \frac{k}{a_n}\left(\frac{a_n}{x}\right)^{k+1}\exp\left(-\left(\frac{a_n}{x}\right)^k\right)$ $F_X(x) = \exp\left(-\left(\frac{a_n}{x}\right)^k\right)$	$x \in [0, \infty)$	$a_n \in (0, \infty)$ $k \in (0, \infty)$	$a_n\Gamma\left(1-\frac{1}{k}\right)$ for $k > 1$ ∞ for $k \leq 1$	$a_n\left[\Gamma\left(1-\frac{2}{k}\right) - \Gamma^2\left(1-\frac{1}{k}\right)\right]^{1/2}$ for $k > 2$ ∞ for $k \leq 2$
Gamma	$f_X(x) = \frac{\lambda^k x^{k-1}\exp(-\lambda x)}{\Gamma(k)}$ $F_X(x) = \frac{\gamma(k,\, \lambda x)}{\Gamma(k)}$	$x \in [0, \infty)$	$k > 0$ $\lambda > 0$	$\frac{k}{\lambda}$	$\sqrt{\frac{k}{\lambda^2}}$
Geometric	$p_X(x) = (1-p)^{x-1}p$ $F_X(x) = 1-(1-p)^x$	$x \in \{1, 2, 3, ...\}$	$p \in (0, 1]$	$\frac{1}{p}$	$\sqrt{\frac{1-p}{p^2}}$
GEV	$f_X(x) = \frac{1}{\alpha}(t(x))^{\beta+1}\exp(-t(x))$ $F_X(x) = \exp(-t(x))$ with $t(x) = \left(1+\beta(\frac{x-\epsilon}{\alpha})\right)^{-1/\beta}$	$x \in [\epsilon - \frac{\alpha}{\beta}, \infty)$ for $\beta > 0$ $x \in (-\infty, \epsilon - \frac{\alpha}{\beta}]$ for $\beta < 0$	$\alpha > 0$ $\beta \in \mathbb{R}$ $\epsilon \in \mathbb{R}$	$\epsilon + \alpha\frac{\Gamma(1-\beta)-1}{\beta}$ for $\beta \neq 0, \beta < 1$ ∞ for $\beta \geq 1$	$\frac{\alpha}{\beta}\sqrt{\Gamma(1-2\beta) - \Gamma(1-\beta)^2}$ for $\beta \neq 0, \beta < 1/2$ ∞ for $\beta \geq \frac{1}{2}$

Name	PDF/PMF and CDF	Support	Parameters	Mean	Standard deviation
GEV Min (mirror image of GEV around ϵ)	$f_X(x) = \frac{1}{\alpha}\left(t(x)\right)^{\beta+1}\exp(-t(x))$ $F_X(x) = 1 - \exp(-t(x))$ with $t(x) = \left(1-\beta(\frac{x-\epsilon}{\alpha})\right)^{-1/\beta}$	$x \in [\epsilon + \frac{\alpha}{\beta}, \infty)$ for $\beta < 0$ $x \in (-\infty, \epsilon + \frac{\alpha}{\beta}]$ for $\beta > 0$	$\alpha > 0$ $\beta \in \mathbb{R}$ $\epsilon \in \mathbb{R}$	$\epsilon - \alpha\frac{\Gamma(1-\beta)-1}{\beta}$ for $\beta \neq 0, \beta < 1$ ∞ for $\beta \geq 1$	$\frac{\alpha}{\beta}\sqrt{\Gamma(1-2\beta) - \Gamma(1-\beta)^2}$ for $\beta \neq 0, \beta < 1/2$ ∞ for $\beta \geq \frac{1}{2}$
Gumbel	$f_X(x) = \frac{1}{a_n}\exp(-z - \exp(-z))$ $F_X(x) = \exp(-\exp(-z))$ with $z = \frac{x - b_n}{a_n}$	$x \in (-\infty, \infty)$	$a_n > 0$ $b_n \in \mathbb{R}$	$b_n + a_n\gamma$ where $\gamma \approx 0.577216$	$\frac{\pi a_n}{\sqrt{6}}$
Gumbel Min (mirror image of Gumbel around b_n)	$f_X(x) = \frac{1}{a_n}\exp(z - \exp(z))$ $F_X(x) = 1 - \exp(-\exp(z))$ with $z = \frac{x - b_n}{a_n}$	$x \in (-\infty, \infty)$	$a_n > 0$ $b_n \in \mathbb{R}$	$b_n - a_n\gamma$ where $\gamma \approx 0.577216$	$\frac{\pi a_n}{\sqrt{6}}$
Log-normal	$f_X(x) = \frac{1}{x\sigma_{lnX}\sqrt{2\pi}}\exp\left(-\frac{(\ln(x) - \mu_{lnX})^2}{2\sigma_{lnX}^2}\right)$ $F_X(x) = \frac{1}{2} + \frac{1}{2}\,\mathrm{erf}\left[\frac{\ln x - \mu_{lnX}}{\sqrt{2}\sigma_{lnX}}\right]$	$x \in (0, \infty)$	$\mu_{lnX} \in \mathbb{R}$ $\sigma_{lnX} > 0$	$\exp\left(\mu_{lnX} + \frac{\sigma_{lnX}^2}{2}\right)$	$\exp\left(\mu_{lnX} + \frac{\sigma_{lnX}^2}{2}\right)\sqrt{\exp(\sigma_{lnX}^2) - 1}$

Name	PDF/PMF and CDF	Support	Parameters	Mean	Standard deviation
Negative binomial	$p_X(x) = \binom{x-1}{k-1}(1-p)^{x-k}p^k$ $F_X(x) = \sum_{i=k}^{x}\binom{i-1}{k-1}(1-p)^{i-k}p^k$	$x \in \{k, k+1, ...\}$	$k \in \mathbb{N}$ $p \in (0,1)$	$\frac{k}{p}$	$\sqrt{\frac{k(1-p)}{p^2}}$
Normal	$f_X(x) = \frac{1}{\sqrt{2\sigma^2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ $F_X(x) = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right]$	$x \in \mathbb{R}$	$\mu \in \mathbb{R}$ $\sigma > 0$	μ	σ
Pareto	$f_X(x) = \frac{\alpha\, x_\mathrm{m}^\alpha}{x^{\alpha+1}}$ $F_X(x) = 1 - \left(\frac{x_\mathrm{m}}{x}\right)^\alpha$	$x \in [x_m, \infty)$	$x_m > 0$ $\alpha > 0$	$\frac{\alpha\, x_\mathrm{m}}{\alpha - 1}$ for $\alpha > 1$	$\sqrt{\frac{x_\mathrm{m}^2\alpha}{(\alpha-1)^2(\alpha-2)}}$ for $\alpha > 2$
Poisson	$p_X(x) = \frac{\lambda^x \exp(-\lambda)}{x!} = \frac{(vt)^x \exp(-vt)}{x!}$ $F_X(x) = \exp(-\lambda)\sum_{i=0}^{\lfloor x \rfloor}\frac{\lambda^i}{i!}$	$x \in \{0, 1, 2, ...\}$	$\lambda > 0$ or $v > 0, \quad t > 0$	$\lambda = vt$	$\sqrt{\lambda} = \sqrt{vt}$
Rayleigh	$f_X(x) = \frac{x}{\alpha^2}\exp(-x^2/2\alpha^2)$ $F_X(x) = 1 - \exp(-x^2/2\alpha^2)$	$x \in [0, \infty)$	$\alpha > 0$	$\alpha\sqrt{\frac{\pi}{2}}$	$\sqrt{\frac{4-\pi}{2}\alpha^2}$

Name	PDF/PMF and CDF	Support	Parameters	Mean	Standard deviation
Standard normal	$\varphi(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}u^2\right)$ $\Phi(u) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{u} \exp\left(-t^2/2\right) \mathrm{d}t$	$u \in \mathbb{R}$	–	0	1
Truncated normal	$f_X(x) = \frac{\varphi\left(\frac{x-\mu_n}{\sigma_n}\right)}{\sigma_n\left(\Phi\left(\frac{b-\mu_n}{\sigma_n}\right) - \Phi\left(\frac{a-\mu_n}{\sigma_n}\right)\right)}$ $F_X(x) = \frac{\Phi\left(\frac{x-\mu_n}{\sigma_n}\right) - \Phi\left(\frac{a-\mu_n}{\sigma_n}\right)}{\Phi\left(\frac{b-\mu_n}{\sigma_n}\right) - \Phi\left(\frac{a-\mu_n}{\sigma_n}\right)}$	$x \in [a, b]$	$\mu_n \in \mathbb{R}$ $\sigma_n > 0$ $a < b$	$\int_a^b x \cdot f_X(x)\mathrm{d}x$	$\sqrt{\int_a^b x^2 \cdot f_X(x)\mathrm{d}x - \left(\int_a^b x \cdot f_X(x)\mathrm{d}x\right)^2}$
Uniform	$f_X(x) = \begin{cases} \frac{1}{b-a} & x \in [a, b] \\ 0 & \text{otherwise} \end{cases}$ $F_X(x) = \begin{cases} 0 & x < a \\ \frac{x-a}{b-a} & x \in [a, b) \\ 1 & x \geq b \end{cases}$	$x \in [a, b]$	$-\infty < a < \infty$ $-\infty < b < \infty$	$\frac{1}{2}(a + b)$	$\sqrt{\frac{1}{12}(b-a)^2}$
Weibull	$f_X(x) = \frac{k}{a_n}\left(\frac{x}{a_n}\right)^{k-1} \exp\left(-\left(\frac{x}{a_n}\right)^k\right)$ $F_X(x) = 1 - \exp\left(-\left(\frac{x}{a_n}\right)^k\right)$	$x \in [0, \infty)$	$a_n \in (0, \infty)$ $k \in (0, \infty)$	$a_n\Gamma(1 + 1/k)$	$a_n\left[\Gamma\left(1 + \frac{2}{k}\right) - \Gamma^2\left(1 + \frac{1}{k}\right)\right]^{1/2}$

In order to compute some of the previous expressions the following special functions are required:

- the error function,
$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2)\mathrm{d}t$$
- The beta function,
$$B(x,y) = \int_0^1 t^{x-1}(1-t)^{y-1}\mathrm{d}t$$
- The regularized beta function,
$$I_x(a,b) = \frac{B(x;a,b)}{B(a,b)} = \frac{\int_0^x t^{a-1}\,(1-t)^{b-1}\mathrm{d}t}{B(a,b)}$$
- The gamma function,
$$\Gamma(t) = \int_0^\infty x^{t-1}\exp(-x)\ \mathrm{d}x$$
- The lower incomplete gamma function,
$$\gamma(s,x) = \int_0^x t^{s-1}\exp(-t)\ \mathrm{d}t$$