ERADist and ERANataf classes: Documentation and Examples

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

July 28, 2016

1 Introduction and Background

The present tutorial is not meant to make a complete explanation of the functions and methods implemented in the MATLAB® codes, but to present a general overview of the structure of the programs, their use and applications.

1.1 Object-oriented programming in MATLAB®

We can usually implement simple numerical methods using built-in functions in MATLAB® . However, with the increasing 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 large numbers of functions that interact between them.

Object-oriented programming allows us to define not only the type of a data structure, but also the 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 [1]. In particular, a *class* is a template that describes objects with common characteristics; here, we must identify and establish all the objects we want to manipulate and how they relate to each other.

MATLAB® admits to create programs using objects and ordinary functions together in our codes [6]. In general, a MATLAB® class (command `classdef`), should contain the following main blocks:

- `properties` block: Defines all properties to be associated with a class. Defines attributes of all properties and default values.
- `methods` block: Defines functions associated with the class and their attributes. The first method must have the same name as the class and it is called the constructor.

In the following, two special MATLAB® classes have been designed in order to facilitate the implementation of several methods in the field of probabilistic analysis. The first class, termed `ERADist`, has as main function the generation of various types of probability distributions. The second class, called `ERANataf`, seeks to apply the Nataf isoprobabilistic transformation to a given set of correlated samples.

1.2 The `ERADist` class

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. Therefore, it is desirable to have a software toolbox that allows the use of different types of probabilistic models, in order to combine them with mathematical modeling techniques for some particular applications.

MATLAB® is a well-suited environment for statistical computations. Its statistics and machine learning toolbox provides functions to describe, analyze, and model data using statistics and machine learning [5]. However, the information that describes a given distribution not always came in terms

of its parameters. In some cases, first-order moments or point measurements 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 with their parameters, moments or experimental data.

1.3 The `ERANataf` class

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, Hermite polynomials transformation, among others (see e.g. [3]).

The Nataf transformation ([7]) 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 [2]. 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 [3].

After applying the Nataf transformation the joint PDF and CDF of the random vector $\mathbf{X}$ can be computed as [3],

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

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

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

2 Definition of the `ERADist` class

2.1 'Properties' block

The properties block contains 3 elements:

- `Name`: Specifies the name of the distribution.
- `Par`: Specifies the type of input the parameters of the distribution.
- `Dist`: Object for the marginal distribution.

2.2 'Methods' block

The methods block is defined by 7 functions:
`-Obj = ERADist(name,opt,val)` (the constructor),
`-Mean = mean(Obj)` (returns the mean value),
`-Standarddeviation = std(Obj)` (returns the standard deviation),
`-CDF = cdf(Obj,x)` (returns the value of the CDF),
`-InverseCDF = icdf(Obj,y)` (returns the value of the inverse CDF),
`-PDF = pdf(Obj,x)` (returns the value of the PDF),
`-Random = random(Obj,m,n)` (generates random numbers).

Table 1: `ERADist` distribution names.

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

It should be noted that MATLAB® already supports 27 different distributions (www.mathworks.com/help/stats/makedist.html#inputarg_distname).

In our case, the constructor `Obj = ERADist(name,opt,val)` takes the argument `name` from our list of supported probability distributions (20 distributions) given in Table 1.

The argument `opt` can be chosen as:

- `PAR`: If you want to specify the distribution by its parameters (Table 2).

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

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

- `MOM`: If you want to specify the distribution by its first moments (Table 3).

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

Distribution	Defined by moments
Binomial	`Obj = ERADist('binomial','MOM',[mean,std])`
Geometric	`Obj = ERADist('geometric','MOM',[mean])`
Negative binomial	`Obj = ERADist('negativebinomial','MOM',[mean,std])`
Poisson	`Obj = ERADist('poisson','MOM',[mean])`
Uniform	`Obj = ERADist('uniform','MOM',[lower,upper])`
Normal	`Obj = ERADist('normal','MOM',[mean,std])`
Standard normal	`Obj = ERADist('standardnormal','MOM',[])`
Log-normal	`Obj = ERADist('lognormal','MOM',[mean,std])`
Exponential	`Obj = ERADist('exponential','MOM',[mean])`
Gamma	`Obj = ERADist('gamma','MOM',[mean,std])`
Beta	`Obj = ERADist('beta','MOM',[mean,std,lower,upper])`
Gumbel (to model minima)	`Obj = ERADist('gumbel','MOM',[mean,std])`
Gumbel (to model maxima)	`Obj = ERADist('gumbelMax','MOM',[mean,std])`
Fréchet	`Obj = ERADist('frechet','MOM',[mean,std])`
Weibull	`Obj = ERADist('weibull','MOM',[mean,std])`
GEV (to model maxima)	`Obj = ERADist('GEV','MOM',[mean,std,epsilon])`
GEV (to model minima)	`Obj = ERADist('GEVMin','MOM',[mean,std,epsilon])`
Pareto	`Obj = ERADist('pareto','MOM',[mean,std])`
Rayleigh	`Obj = ERADist('rayleigh','MOM',[mean])`
Chi-squared	`Obj = ERADist('chisquare','MOM',[mean])`

- `DATA`: If you want to specify the distribution based on some data vector (this option is only available for some distributions, see Table 4).

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

Distribution	Defined by moments
Poisson	`Obj = ERADist('poisson','DATA',[X])`
Normal	`Obj = ERADist('normal','DATA',[X])`
Log-normal	`Obj = ERADist('lognormal','DATA',[X])`
Exponential	`Obj = ERADist('exponential','DATA',[X])`
Gamma	`Obj = ERADist('gamma','DATA',[X])`
Gumbel (to model minima)	`Obj = ERADist('gumbel','DATA',[X])`
Gumbel (to model maxima)	`Obj = ERADist('gumbelMax','DATA',[X])`
Weibull	`Obj = ERADist('weibull','DATA',[X])`
GEV (to model maxima)	`Obj = ERADist('GEV','DATA',[X])`
GEV (to model minima)	`Obj = ERADist('GEVMin','DATA',[X])`
Rayleigh	`Obj = ERADist('rayleigh','DATA',[X])`

Finally, the argument `val` requires the numeric value of the given `opt`, i.e., the value of the parameters if we choose `PAR`, the value of the moments if we choose `MOM`, or the data vector (as a column) if we choose `DAT`.

Please refer to Section 5 in order to know our supported probability distributions, their parameters and first-order moments (mean and standard deviation) definitions.

`gumbel` is the Gumbel distribution given by definition (used to model the minimum value), and `gumbelMax` is the mirror image of the Gumbel distribution (used to model the maximum value). Similar distinction applies to the `GEV` and `GEVMin` distributions.

3 Definition of the `ERANataf` class

3.1 'Properties' block

The properties block contains 4 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.

3.2 'Methods' block

The methods block is defined by 6 functions:
`-Obj = ERANataf(M,Correlation)` (the constructor),
`-[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)` (random numbers according to the joint distribution),
`-jointpdf = pdf(Nataf,X)` (returns the joint PDF),
`-jointcdf = cdf(Nataf,X)` (returns the joint CDF).

The constructor `Obj = ERANataf(M,Correlation)` takes the argument `M` containing the marginals of the input correlated random variables expressed as distribution objects (e.g. using the `ERADist` class); and the argument `Correlation` defines the correlation matrix of these random variables.

It is worth mentioning that the calculation 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.

4 Application examples

Please also refer to the file `input_file.m`, which contains some of the examples that will be introduced in the following.

1. Creating distribution objects: For some distributions the parameters coincide with the first-order moments (e.g. Normal distribution).

```
%%% create distribution object: Normal distribution
mu       = 4;          %% mean
sigma    = 2;          %% standard deviation
dist_obj = ERADist('normal','PAR',[mu,sigma]);
%% which is equal to
dist_obj = ERADist('normal','MOM',[mu,sigma]);
```

   However, this not apply for most of the distributions.

```
%%% create distribution object: Exponential distribution
lambda   = 2;                  %% rate parameter
dist_obj = ERADist('exponential','PAR',lambda);
fprintf('mean %g and standard deviation %g\n',dist_obj.mean,dist_obj.std);
%% which is equal to
mu       = lambda^(-1);          %% mean
sigma    = sqrt(lambda^(-2));    %% standard deviation
dist_obj = ERADist('exponential','MOM',[mu,sigma]);
fprintf('mean %g and standard deviation %g\n',dist_obj.mean,dist_obj.std);
```

2. Plotting distributions

```
%%% mu and sigma of the lognormal random variable
mu       = 1;     %% mean (or PAR(1)=-0.8047);
sigma    = 0.4;   %% standard deviation (or PAR(2)=1.2686);
dist_obj = ERADist('lognormal','MOM',[mu,sigma]);
%% plotting the distribution
xx = 0:0.01:3;
figure;
subplot(1,2,1); plot(xx,dist_obj.pdf(xx)); grid minor;
subplot(1,2,2); plot(xx,dist_obj.cdf(xx)); grid minor;
```

3. Difference between `gumbel` and `gumbelMax`.

```
%%% fit a Gumbel distribution to minimum values
xMinima   = min(randn(1000,500), [], 2); N1 = length(xMinima);
[fx1,nx1] = hist(xMinima,ceil(sqrt(N1)));
dist_obj1 = ERADist('gumbel','DATA',xMinima);

%% fit a Gumbel distribution to maximum values
xMaxima   = max(randn(1000,500), [], 2); N2 = length(xMaxima);
[fx2,nx2] = hist(xMaxima,ceil(sqrt(N2)));
dist_obj2 = ERADist('gumbelMax','DATA',xMaxima);

%% plot
xx = -5:0.01:5;
figure; hold on;
bar(nx1,fx1/(N1*(nx1(2)-nx1(1))),'FaceColor',[0 .7 .7],'EdgeColor',[0 1 1]);
plot(xx,dist_obj1.pdf(xx),'b','LineWidth',2);
bar(nx2,fx2/(N2*(nx2(2)-nx2(1))),'FaceColor',[0 .3 .3],'EdgeColor',[0 1 1]);
plot(xx,dist_obj2.pdf(xx),'r','LineWidth',2);
legend('hist mindata','gumbel','hist maxdata','gumbelmax');
```

4. Difference between `GEVMin` and `GEV`.

```
%%% fit a GEV distribution to minimum values
xMinima   = min(randn(1000,500), [], 2); N1 = length(xMinima);
[fx1,nx1] = hist(xMinima,ceil(sqrt(N1)));
```

```
dist_obj1 = ERADist('GEVMin','DATA',xMinima);

%% fit a GEV distribution to maximum values
xMaxima   = max(randn(1000,500), [], 2); N2 = length(xMaxima);
[fx2,nx2] = hist(xMaxima,ceil(sqrt(N2)));
dist_obj2 = ERADist('GEV','DATA',xMaxima);

%% plot
xx = -5:0.01:5;
figure; hold on;
bar(nx1,fx1/(N1*(nx1(2)-nx1(1))),'FaceColor',[0 .7 .7],'EdgeColor',[0 1 1]);
plot(xx,dist_obj1.pdf(xx),'b','LineWidth',2);
bar(nx2,fx2/(N2*(nx2(2)-nx2(1))),'FaceColor',[0 .3 .3],'EdgeColor',[0 1 1]);
plot(xx,dist_obj2.pdf(xx),'r','LineWidth',2);
legend('hist mindata','GEVMin','hist maxdata','GEV');
```

5. Generating random numbers:

```
%%% create distribution object
mu       = 48.16;          %% mean (or PAR(1)=58.853)
sigma    = 23.76;          %% standard deviation (or PAR(2)=18.526)
dist_obj = ERADist('gumbel','MOM',[mu,sigma]);
%% generate an array of m x n random numbers
m     = 1e4;
n     = 3;
theta = dist_obj.random(m,n);
[n,x] = hist(theta(:,1),ceil(sqrt(m)));
figure; bar(x,n/(m*(x(2)-x(1))));
```

6. Applying Nataf transformation

```
%%% marginal distribution 1
alpha = 2;    %% shape parameter
beta  = 1;    %% rate parameter
M(1)  = ERADist('gamma','PAR',[alpha,beta]);
%% marginal distribution 2
nu    = 5;     %% degrees of freedom
M(2) = ERADist('chisquare','PAR',nu);
%% correlation matrix
Rho = [ 1.0 0.5;
        0.5 1.0  ];

%% applying Nataf transformation
T_Nataf = ERANataf(M,Rho);
%% generation of random samples from the join distribution
N = 1e3;   %% number of samples
X = T_Nataf.random(N);
%% samples in standard space
U = T_Nataf.X2U(X);

%% plot the samples in physical and standard spaces
figure;
subplot(121); plot(X(1,:),X(2,:),'bx','LineWidth',2);
title('Physical space','Interpreter','Latex','FontSize', 20);
xlabel('$X_1$','Interpreter','Latex','FontSize', 20);
ylabel('$X_2$','Interpreter','Latex','FontSize', 20);
set(gca,'FontSize',17);
subplot(122); plot(U(1,:),U(2,:),'ro','LineWidth',2); axis equal;
title('Standard space','Interpreter','Latex','FontSize', 20);
xlabel('$U_1$','Interpreter','Latex','FontSize', 20);
ylabel('$U_2$','Interpreter','Latex','FontSize', 20);
set(gca,'FontSize',17);
```

7. Nataf transformation: Joint PDF and CDF

```
%%% marginal distribution 1
mu    = 1;
sigma = 0.4;
M(1)  = ERADist('lognormal','MOM',[mu,sigma]);
%% marginal distribution 2
M(2) = ERADist('normal','MOM',[mu,sigma]);
%% correlation matrix
Rho = [ 1.0 0.6
        0.6 1.0 ];
%% applying Nataf transformation
T_Nataf = ERANataf(M,Rho);

%% joint PDF and CDF
jointf_X = @(X) T_Nataf.pdf(X);
jointF_X = @(X) T_Nataf.cdf(X);

%% plot joint PDF
n       = 50;
x1      = linspace(0,3,n);
x2      = linspace(-0.5,2.5,n);
[X1,X2] = meshgrid(x1,x2);
ff_x    = jointf_X([X1(:) X2(:)]');  ff_x = reshape(ff_x,n,n);
FF_x    = jointF_X([X1(:) X2(:)]');  FF_x = reshape(FF_x,n,n);
figure;
subplot(121); pcolor(X1,X2,ff_x); colorbar; shading interp; axis equal tight;
xlabel('$X_1$','Interpreter','Latex','FontSize', 20);
ylabel('$X_2$','Interpreter','Latex','FontSize', 20);
title('PDF','Interpreter','Latex','FontSize', 20);
set(gca,'FontSize',17);
subplot(122); pcolor(X1,X2,FF_x); colorbar; shading interp; axis equal tight;
xlabel('$X_1$','Interpreter','Latex','FontSize', 20);
ylabel('$X_2$','Interpreter','Latex','FontSize', 20);
title('CDF','Interpreter','Latex','FontSize', 20);
set(gca,'FontSize',17);
```

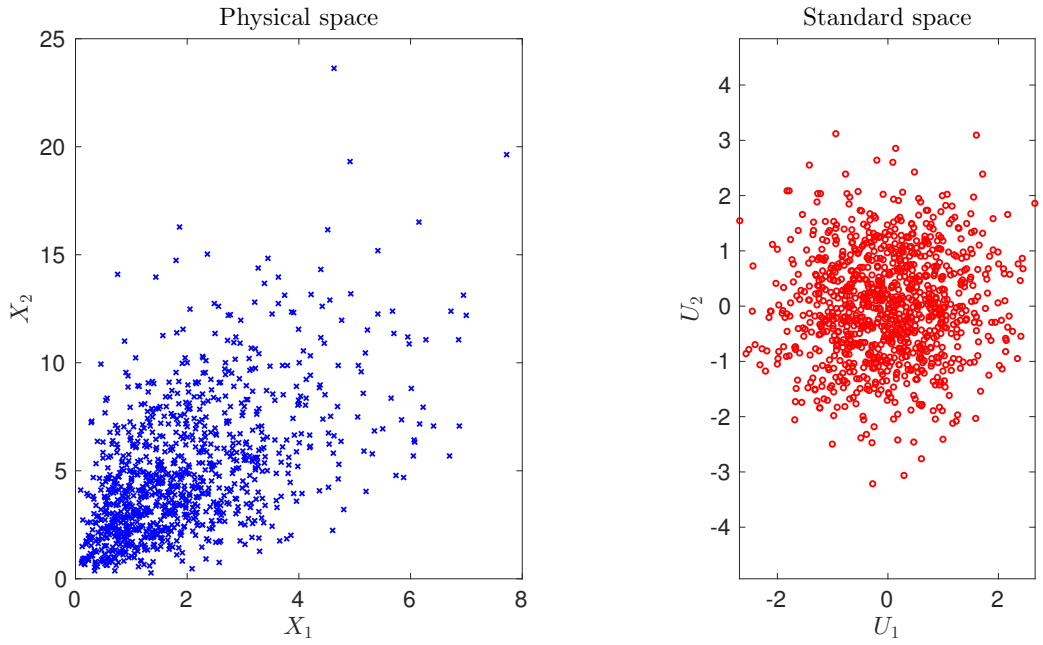

Figure 1: Samples in physical and standard spaces from the Nataf transformation (Example 6).

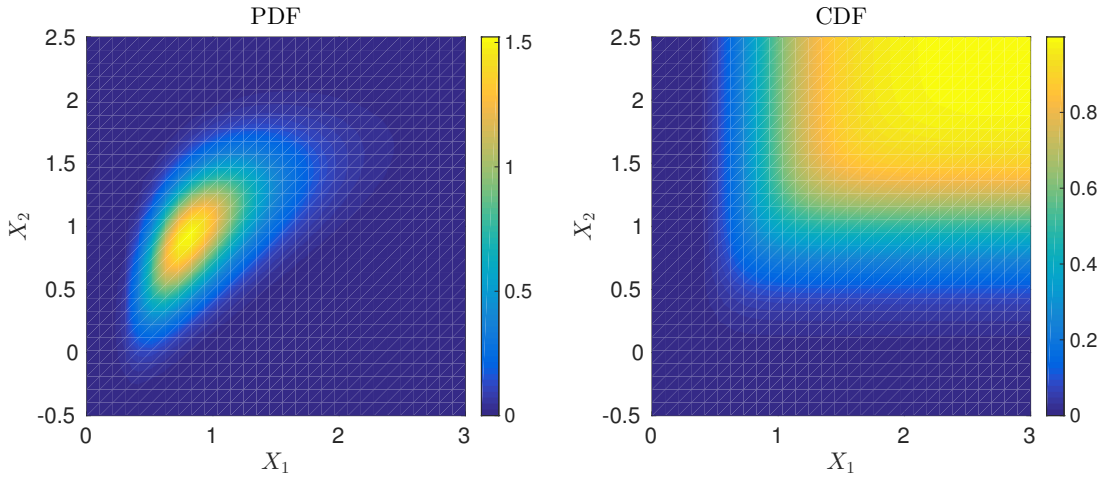

Figure 2: Joint PDF and CDF from the Nataf transformation (Example 7).

5 Supported Probability Distributions

Name	PMF/CDF	Support	Parameters	Mean	Standard deviation
Binomial	$p_X(x) = \binom{n}{x} p^x (1-p)^{n-x}$ $F_X(x) = \sum_{i=0}^{\lfloor x \rfloor} \binom{n}{i} p^i (1-p)^{n-i}$	$x \in \{0, \ldots, n\}$	$n \in \mathbb{N}_0$ $p \in [0,1]$	np	$\sqrt{np(1-p)}$
Geometric	$p_X(x) = (1-p)^{x-1} p$ $F_X(x) = 1 - (1-p)^x$	$x \in \{1, 2, 3, \ldots\}$	$p \in (0,1]$	$\frac{1}{p}$	$\sqrt{\frac{1-p}{p^2}}$
Negative binomial	$p_X(x) = \binom{x+k-1}{x} (1-p)^k p^x$ $F_X(x) = 1 - I_p(k+1, r)$	$x \in \{0, 1, 2, \ldots\}$	$k \in \mathbb{N}_0$ $p \in (0,1)$	$\frac{pk}{1-p}$	$\sqrt{\frac{pk}{(1-p)^2}}$
Poisson	$p_X(x) = \frac{\lambda^x \exp(-\lambda)}{x!} = \frac{(vt)^x \exp(-vt)}{x!}$ $F_X(x) = \exp(-\lambda) \sum_{i=0}^{\lfloor x \rfloor} \frac{\lambda^i}{i!}$	$x \in \{0, 1, 2, \ldots\}$	$\lambda > 0$ $v > 0, \quad t > 0$	$\lambda = vt$	$\sqrt{\lambda} = \sqrt{vt}$
Uniform	$f_X(x) = \begin{cases} \frac{1}{b-a} & x \in [a,b] \\ 0 & \text{otherwise} \end{cases}$ $F_X(x) = \begin{cases} 0 & x < a \\ \frac{x-a}{b-a} & x \in [a,b) \\ 1 & x \geq b \end{cases}$	$x \in [a,b]$	$-\infty < a < \infty$ $-\infty < b < \infty$	$\frac{1}{2}(a+b)$	$\sqrt{\frac{1}{12}(b-a)^2}$
Standard normal	$\phi(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}u^2\right)$ $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{u} \exp\left(-t^2/2\right) \mathrm{d}t$	$u \in \mathbb{R}$	–	0	1

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
Log-normal	$f_X(x) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left(-\frac{(\ln(x)-\mu)^2}{2\sigma^2}\right)$ $F_X(x) = \frac{1}{2} + \frac{1}{2}\operatorname{erf}\left[\frac{\ln x - \mu}{\sqrt{2}\sigma}\right]$	$x \in (0, \infty)$	$\mu \in \mathbb{R}$ $\sigma > 0$	$e^{\mu+(\sigma^2/2)}$	$\sqrt{(e^{\sigma^2}-1)e^{2\mu+\sigma^2}}$
Exponential	$f_X(x) = \lambda \exp(\lambda x)$ $F_X(x) = 1 - \lambda \exp(\lambda x)$	$x \in [0, \infty)$	$\lambda > 0$	λ^{-1}	$\sqrt{\lambda^{-2}}$
Gamma	$f_X(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$ $F_X(x) = \frac{1}{\Gamma(\alpha)}\gamma(\alpha, \beta x)$	$x \in (0, \infty)$	$\alpha > 0$ $\beta > 0$	$\frac{\alpha}{\beta}$	$\sqrt{\frac{\alpha}{\beta^2}}$
Beta	$f_X(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}$ $F_X(x) = I_x(\alpha, \beta)$	$x \in (0, 1)$	$\alpha > 0$ $\beta > 0$	$\frac{\alpha}{\alpha+\beta}$	$\sqrt{\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}}$
Gumbel	$f_X(x) = \frac{1}{a_n}\exp(-z - \exp(-z))$ $F_X(x) = \exp(-\exp(-z))$ with $z = \frac{x-b_n}{a_n}$	$x \in (-\infty, \infty)$	$b_n \in \mathbb{R}$ $a_n > 0$	$b_n + a_n\gamma$ $\gamma \approx 0.577216$	$\sqrt{\frac{\pi^2}{6}a_n^2}$
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 $k > 1$	$a_n\left[\Gamma\left(1-\frac{2}{k}\right) - \Gamma^2\left(1-\frac{1}{k}\right)\right]^{1/2}$ for $k > 2$
Weibull	$f_X(x) = \frac{k}{a_n}\left(\frac{x}{a_n}\right)^{k-1} \exp\left(-\left(\frac{x}{a_n}\right)^k\right)$ $F_X(x) = 1 - \exp\left(-\left(\frac{x}{a_n}\right)^k\right)$	$x \in [0, \infty)$	$a_n \in (0, \infty)$ $k \in (0, \infty)$	$a_n\Gamma(1+1/k)$	$a_n\left[\Gamma\left(1+\frac{2}{k}\right) - \Gamma^2\left(1+\frac{1}{k}\right)\right]^{1/2}$
GEV	$f_X(x) = \frac{1}{\sigma}(t(x))^{\xi+1}\exp(-t(x))$ $F_X(x) = \exp(-t(x))$ with $t(x) = \left(1 + (\frac{x-\mu}{\sigma})\xi\right)^{-1/\xi}$	$x \in [\mu - \sigma/\xi, \infty)$ $x \in (-\infty, \mu - \sigma/\xi]$	$\mu \in \mathbb{R}$ $\sigma > 0$ $\xi \in \mathbb{R}$	$\mu + \sigma\frac{\Gamma(1-\xi)-1}{\xi}$ for $\xi \neq 0, \xi < 1$	$\sqrt{\sigma^2(g_2 - g_1^2)/\xi^2}$ for $\xi \neq 0, \xi < 1/2$ where $g_k = \Gamma(1-k\xi)$

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 $\alpha > 2$
Rayleigh	$f_X(x) = \frac{x}{\sigma^2}\exp(-x^2/2\sigma^2)$ $F_X(x) = 1 - \exp(-x^2/2\sigma^2)$	$x \in [0, \infty)$	$\sigma > 0$	$\sigma\sqrt{\frac{\pi}{2}}$	$\sqrt{\frac{4-\pi}{2}\sigma^2}$
Chi-squared	$f_X(x) = \frac{1}{2^{\frac{k}{2}}\Gamma\left(\frac{k}{2}\right)} x^{\left(\frac{k}{2}-1\right)} \exp\left(-\frac{x}{2}\right)$ $F_X(x) = \frac{1}{\Gamma\left(\frac{k}{2}\right)}\gamma\left(\frac{k}{2}, \frac{x}{2}\right)$	$x \in [0, \infty)$	$k \in \mathbb{N}_{>0}$	k	$\sqrt{2k}$

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

- the error function,
$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2)\mathrm{d}t$$
- 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,
$$\gamma(s, x) = \int_0^x t^{s-1} \exp(-t)\ \mathrm{d}t$$

References

[1] Govindjee, S. (2013) - "Object Oriented Programming and Classes in MATLAB". Lecture notes. http://www.ce.berkeley.edu/~sanjay/e7/oop.pdf.

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

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

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

[5] MathWorks (2016) - "Statistics and Machine Learning Toolbox". www.mathworks.com/help/stats/.

[6] MathWorks (2016) - "Object-Oriented Programming". www.mathworks.com/help/matlab/object-oriented-programming.html.

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