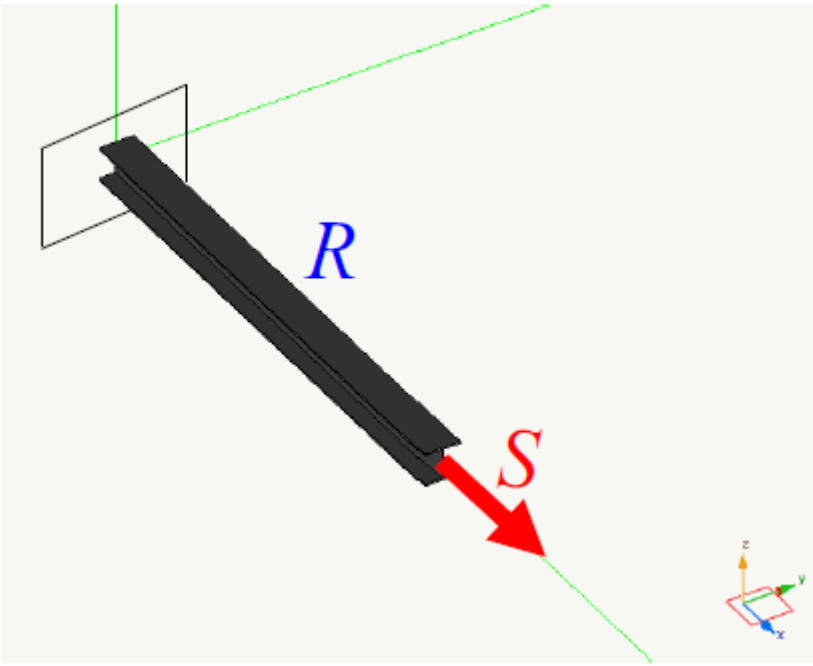
Example B1.1: Bar Example

In [ ]:

```
import OpenAIUQ as auq
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
from scipy import stats
from scipy.stats import pearsonr
from openseespy.postprocessing.Get_Rendering import *

matplotlib.rcParams.update({'font.size':16})
```

Problem definition



We have a steel bar whose yield strength is F_y, whose section has area A, subjected to a nodal load S.

The capacity of the bar is $R = AF_y$, while the demand is S

We assume that the yield strength F_y, and the load S are uncertain.

The basic random variables are $x_1 \equiv R$ and $x_2 \equiv S$

We want to evaluate the failure probability defined as $P_f = Prob[R \leq S]$

We define the Limit State Function (LSF) as:

$G(x_1, x_2) = R - S = x1 - x2$

so that $P_f = Prob(G \leq 0) = Prob(R \leq S)$

We choose a profile HE140A, then the area is $A = 3,140 mm^2$

$x_1 \equiv R$ is modelled through a Gaussian random variable, $\mu = 329 \cdot 3,140 = 1,033 kN$, $\sigma = 103.30 kN$, with coefficient of variation $\nu = 0.10$

$x_2 \equiv S$ is modelled through a Gaussian random variable, $\mu = 650 kN$, $\sigma = 130 kN$, with coefficient of variation $\nu = 0.20$

Define Marginal distributions

In [2]:

```
#=======================
#variable x1

#define mean
m1=1033
#define standard deviation
s1=0.10*m1
#Define distribution: 'Gaussian','Lognormal'
x1=auq.dist('Gaussian')
#Evaluate the parameters through the method of the moments
x1.Momentfit(m1,s1)

#=======================
#variable x2

#define mean
m2=650
#define standard deviation
s2=0.20*m2
#Define distribution: 'Gaussian','Lognormal'
x2=auq.dist('Gaussian')
#Evaluate the parameters through the method of the moments
x2.Momentfit(m2,s2)
```

Define Joint distribution

In [3]:

```
#Include in x all the basic random variables
x=[x1,x2]

#Define the matrix of correlation
corr=np.array([[1,0],
               [0,1]])

#Define the multivariate distribution model: 'Independent','Gaussian','Nataf'
X=auq.dist2(x,corr,'Nataf')
```

Simulate samples

In [4]:

```
#to check the joint distribution, generate samples (num=number of samples)
X.gen_samples(num=10) #num=number of samples
```

In [5]:

```
#samples in x-space
X.samples
```

Out[5]:

```
array([[ 895.91495346,  530.73287056],
       [1101.31467729,  700.59294608],
       [ 949.08630351,  430.02150604],
       [1051.42792789,  410.3105815 ],
       [ 968.00933114,  741.78823631],
       [1055.06171334,  619.02810587],
       [ 969.74606958,  591.55572676],
       [ 943.9487485 ,  654.45270993],
       [ 771.8700044 ,  902.76542481],
       [1017.19313793,  597.77196458]])
```

In [6]:

```
#samples in u-space
X.samples_u
```

Out[6]:

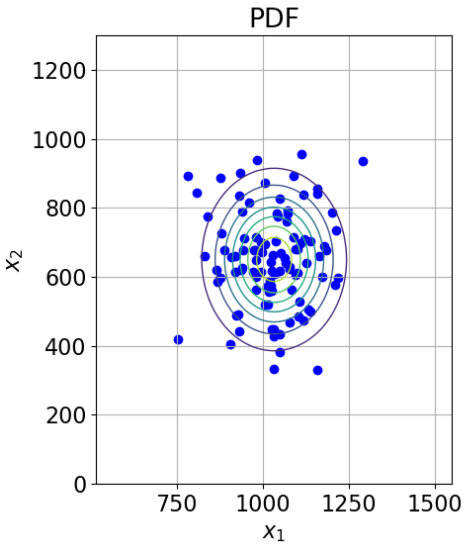
```
array([[-1.32705757, -0.91743946],
       [ 0.66132311,  0.38917651],
       [-0.81233007, -1.69214226],
       [ 0.17839233, -1.84376476],
       [-0.62914491,  0.70606336],
       [ 0.21356935, -0.23824534],
       [-0.61233234, -0.44957133],
       [-0.86206439,  0.03425161],
       [-2.52787992,  1.94434942],
       [-0.15301899, -0.40175412]])
```

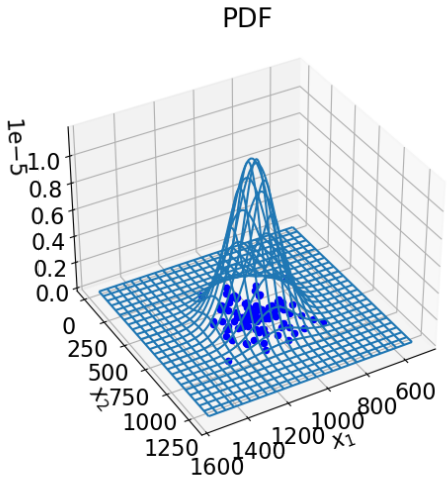
Plot joint distribution

x-space

In [7]:

```
#plot joint distribution
X.plot_w2(numfig=2,figsize=(12,6),dpi=80, space='x',samples=100,sec=[1,2],view=[35,60],rati
#space= 'x','u'
#samples=0 (plot samples)
#sec=[1,2] means plane x1-x2, sec=[1,3] means plane x1-x3, etc
#view=[35,60] parameters of the 3d view, here azimuth=35, elev=60
#ratio='equal', 'no' (equal means the axes have the same ratio)
```

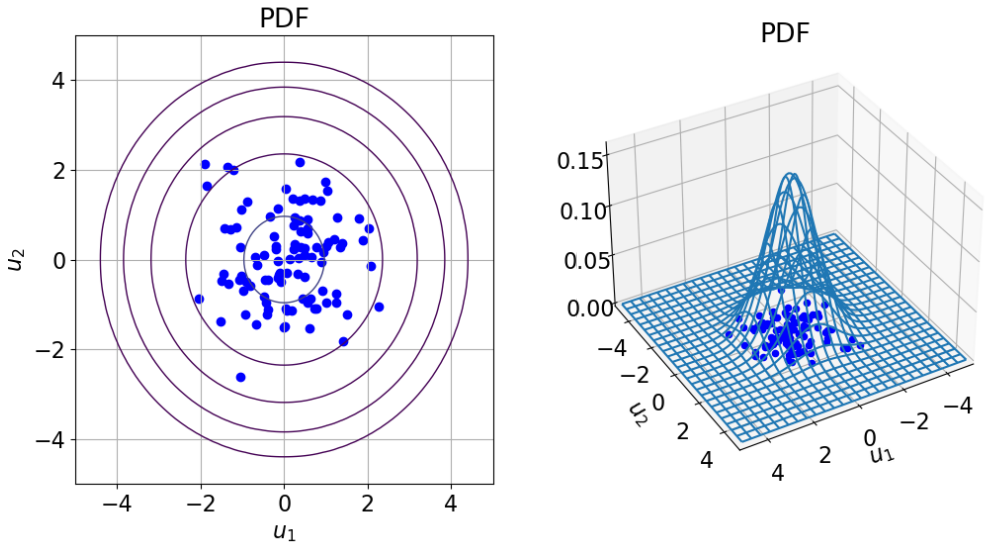




u-space

In [8]:

```
#plot joint distribution
X.plot_w2(numfig=4,figsize=(12,6),dpi=80, space='u',samples=100,sec=[1,2],view=[35,60],rati
#space= 'x','u'
#samples=0 (plot samples)
#sec=[1,2] means plane x1-x2, sec=[1,3] means plane x1-x3, etc
#view=[35,60] parameters of the 3d view, here azimuth=35, elev=60
#ratio='equal', 'no' (equal means the axes have the same ratio)
```



Define Limit State

In [9]:

```
G=auq.Gfun(X,Gfun_id=1)
```

Simulate samples

In [10]:

```
G.gen_samples(X,num=10)
```

In [11]:

```
G.samples
```

Out[11]:

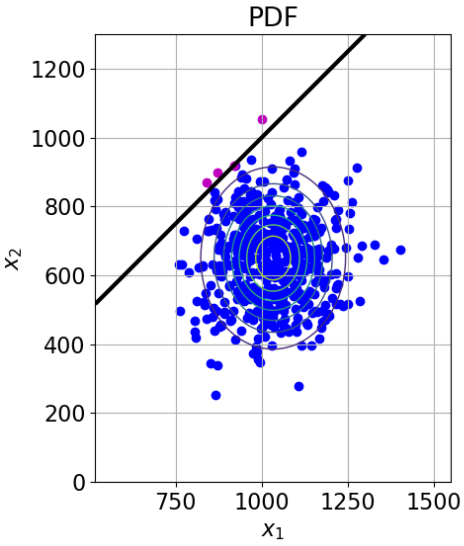
```
array([ 79.12410471, 546.51223186, 328.66228169, 306.08924039,
       246.46829955,  91.28285009, 485.43275795, 498.58575906,
       385.52096729, 618.92424731])
```

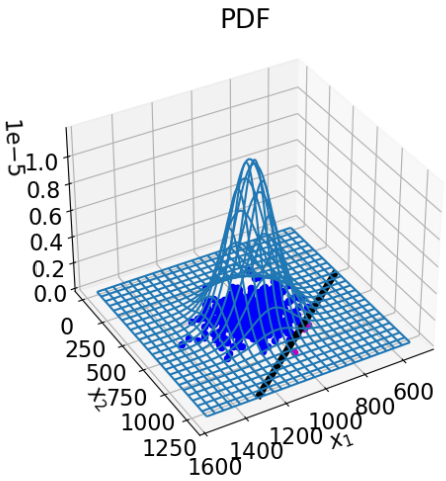
Plot joint distribution with limit state

x-space

In [12]:

```
#plot joint distribution
G.plot_w2(numfig=6,figsize=(12,6),dpi=80,space='x',samples=500,sec=[1,2],view=[35,60],ratio
#space= 'x','u'
#samples=0 (plot samples)
#sec=[1,2] means plane x1-x2, sec=[1,3] means plane x1-x3, etc
#view=[35,60] parameters of the 3d view, here azimuth=35, elev=60
#ratio='equal', 'no' (equal means the axes have the same ratio)
```

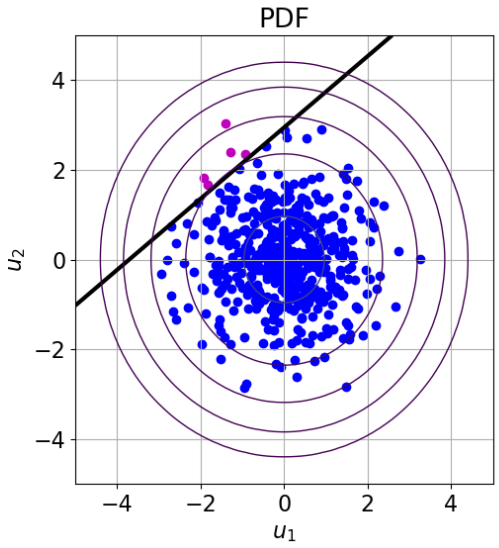


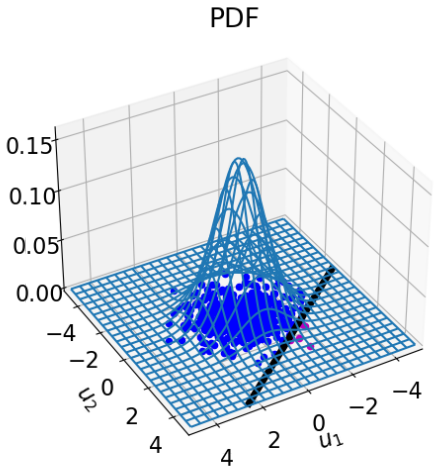


u-space

In [13]:

```
#plot joint distribution
G.plot_w2(numfig=8,figsize=(12,6),dpi=80,space='u',samples=500,sec=[1,2],view=[35,60],ratio
#space= 'x','u'
#samples=0 (plot samples)
#sec=[1,2] means plane x1-x2, sec=[1,3] means plane x1-x3, etc
#view=[35,60] parameters of the 3d view, here azimuth=35, elev=60
#ratio='equal', 'no' (equal means the axes have the same ratio)
```





MCS

In [14]:

```
%%time
print('MCS')
print('--------------------')
print('samples   Pf   cov')
print('--------------------')
MCS=auq.MCS(G)
MCS.crude_MCS(method='cov',cov=0.05,plot='yes')
#cov=0.05, coefficient of variation (here 5%)
#plot='yes','no' ('yes shows the evolution of the MCS')
```

```
MCS
--------------------
samples   Pf   cov
--------------------
[1000, 0.014, 0.2653838190782766]
[2000, 0.0115, 0.20731198920845176]
[3000, 0.009666666666666667, 0.18479563113709244]
[4000, 0.00925, 0.16363687551107492]
[5000, 0.0102, 0.13931203514863638]
[6000, 0.010166666666666666, 0.1273843630760234]
[7000, 0.01042857142857143, 0.11642926176064657]
[8000, 0.010375, 0.10919337329290793]
[9000, 0.010444444444444444, 0.10260207971200835]
[10000, 0.0104, 0.09754683293364586]
[11000, 0.010272727272727272, 0.09358765106153057]
[12000, 0.010333333333333333, 0.089337465800743]
[13000, 0.010153846153846154, 0.08659581109288426]
[14000, 0.009857142857142858, 0.08470506620129419]
[15000, 0.010266666666666667, 0.08016757341244543]
[16000, 0.0103125, 0.07744744063302583]
[17000, 0.010588235294117647, 0.07413994892191247]
[18000, 0.010444444444444444, 0.07255062632820379]
[19000, 0.010473684210526316, 0.0705159134464933]
[20000, 0.01035, 0.06914418178267884]
[21000, 0.01042857142857143, 0.0672204656190587]
[22000, 0.010409090909090911, 0.06573703432327693]
[23000, 0.010391304347826086, 0.06434766629443897]
[24000, 0.010416666666666666, 0.06291528696058958]
[25000, 0.0102, 0.06230223613523819]
[26000, 0.01026923076923077, 0.06088396187501587]
[27000, 0.010148148148148147, 0.0601048916433539]
[28000, 0.01017857142857143, 0.05893265346232321]
[29000, 0.010344827586206896, 0.05743562113107722]
[30000, 0.010533333333333334, 0.05595733757687175]
[31000, 0.010419354838709677, 0.05535085516667302]
[32000, 0.0105, 0.05426730577604232]
[33000, 0.010484848484848484, 0.05347775593741377]
[34000, 0.010470588235294117, 0.05272169382084629]
[35000, 0.010485714285714286, 0.051925178976634706]
[36000, 0.010527777777777778, 0.05109544646700755]
[37000, 0.010621621621621622, 0.05017471708997956]
Wall time: 3.66 s
```

In [15]:

```
print('Pf=',MCS.Pf)
print('beta=',MCS.beta)
print('cov=',MCS.cov)
print('samples=',MCS.num)
```

```
Pf= 0.01068306895435416
beta= 2.301451250045912
cov= 0.04998274780759953
samples= 37068
```