RISK AND RELIABILITY IN ENGINEERING

PART B: STRUCTURAL RELIABILITY

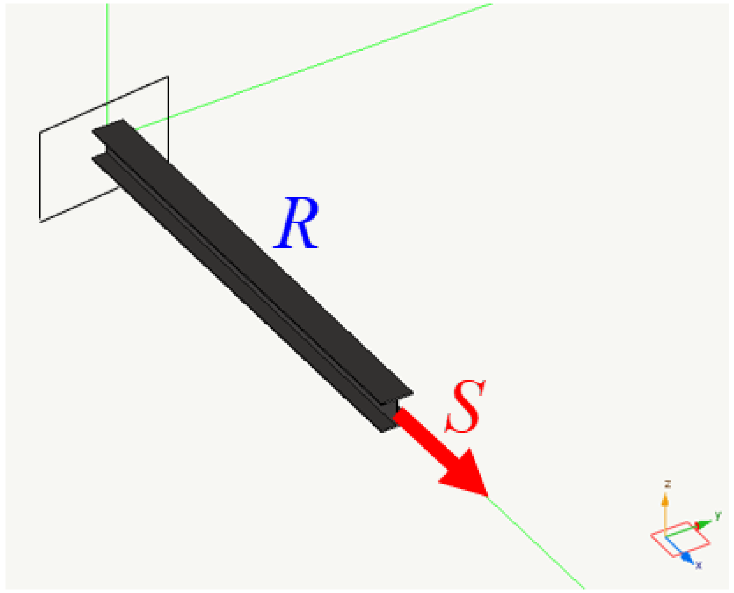
Example B3.1: Bar, Linear Limit State

Umberto Alibrandi

Department of Engineering

Aarhus University

EXAMPLE B3.1: BAR, LINEAR LS

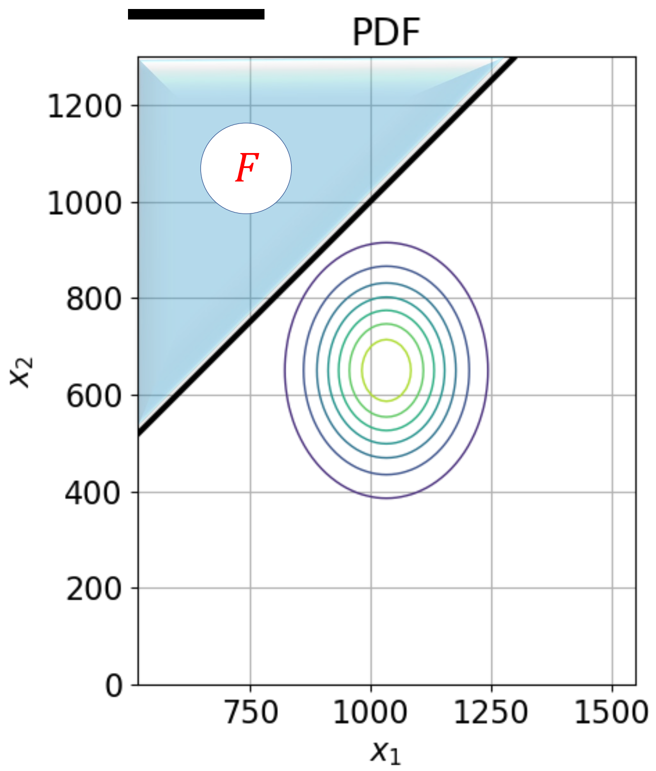


Variable		Distribution	μ	σ	ν
	A		3,142		
x_1	F_y	Normal	$329\ MPa$	$32.90\ MPa$	0.10
x_2	S	Normal	$650\ kN$	$130\ kN$	0.20

$$G(F_y, S) = AF_y - S \quad \Rightarrow \quad G(x_1, x_2) = Ax_1 - x_2$$

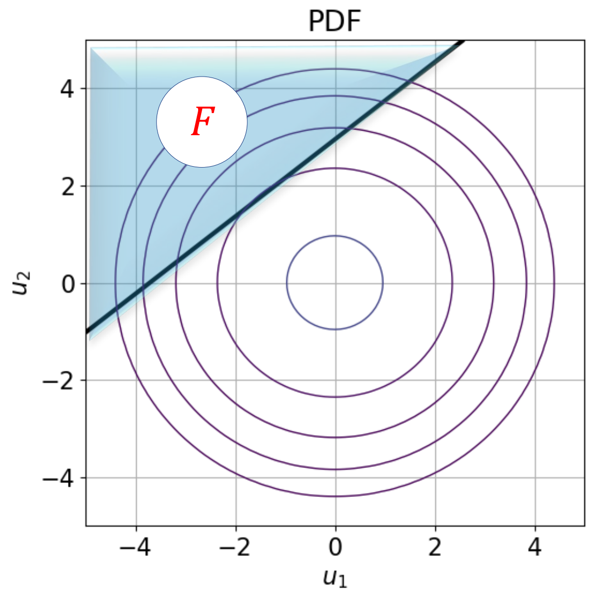
Correlation		
	X_1	X_2
X_1	1	0
X_2	0	1

EXAMPLE B3.1: BAR, LINEAR LS



$$\begin{cases} u_1 = \dfrac{x_1-\mu_1}{\sigma_1} \\ u_2 = \dfrac{x_2-\mu_2}{\sigma_2} \end{cases}$$

$$\begin{cases} x_1 = \mu_1 + u_1\sigma_1 \\ x_2 = \mu_2 + u_2\sigma_2 \end{cases}$$



$$G(x_1, x_2) = Ax_1 - x_2$$

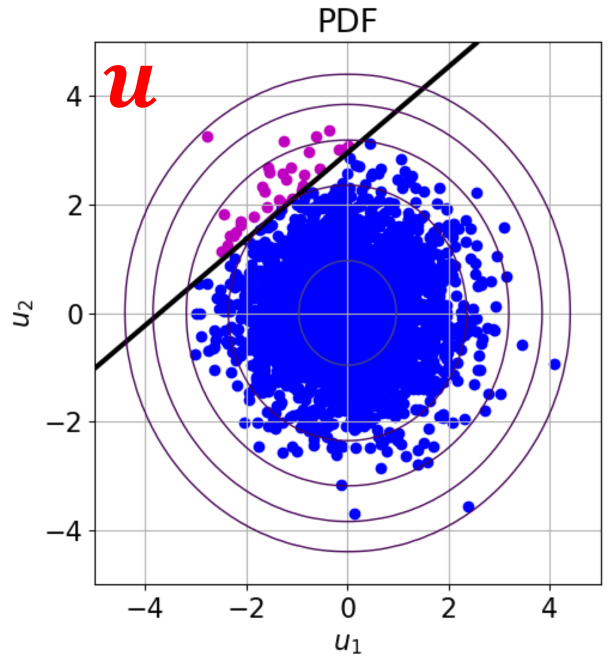
$$g(u_1, u_2) = A(\mu_1 + u_1\sigma_1) - (\mu_2 + u_2\sigma_2)$$

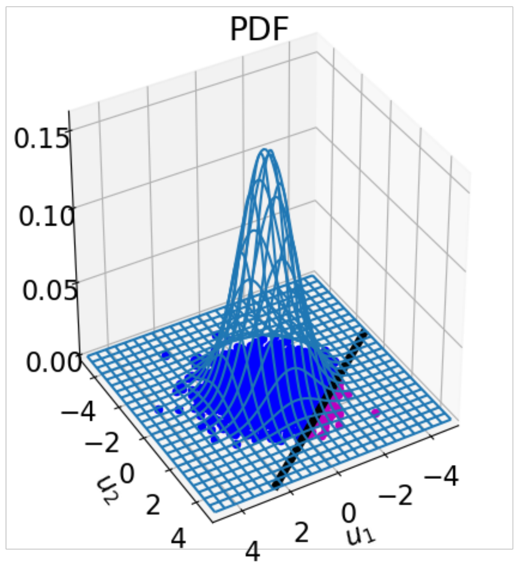
EXAMPLE B3.1: BAR, LINEAR LSMCS

Failure Probability $P_f = \int_{g(\boldsymbol{u})\leq 0} \varphi(\boldsymbol{u})d\boldsymbol{u} = 1.00 \times 10^{-2}$

$$g(u_1, u_2) = A(u_1\sigma_1 + \mu_1) - (u_2\sigma_2 + \mu_2)$$

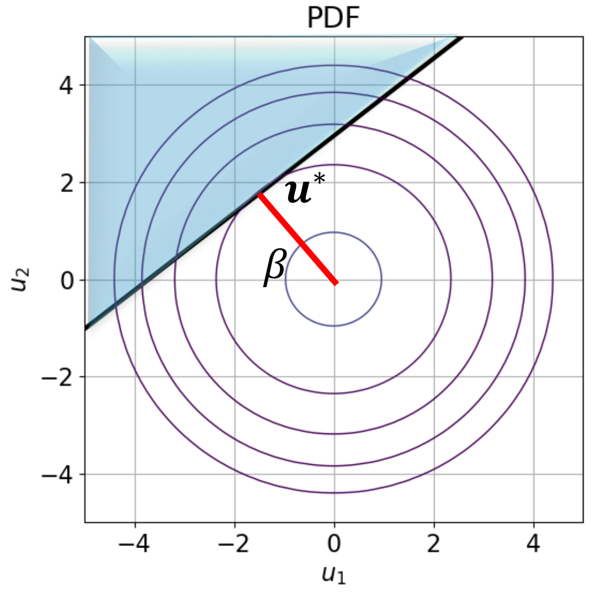
1) Generate $u_1^{(k)}, u_2^{(k)}$
2) Evaluate $g^{(k)} = g\left[u_1^{(k)}, u_2^{(k)}\right]$
3) If $\begin{cases} g^{(k)} \leq 0 & N_f = N_f + 1, N = N + 1 \\ g^{(k)} > 0 & N = N + 1 \end{cases}$,
4) Go to step 1 until needed
5) $P_f = \frac{N_f}{N}$





EXAMPLE B3.1: BAR, LINEAR LSFORM

$$g(u_1, u_2) = A(\mu_1 + u_1\sigma_1) - (\mu_2 + u_2\sigma_2)$$



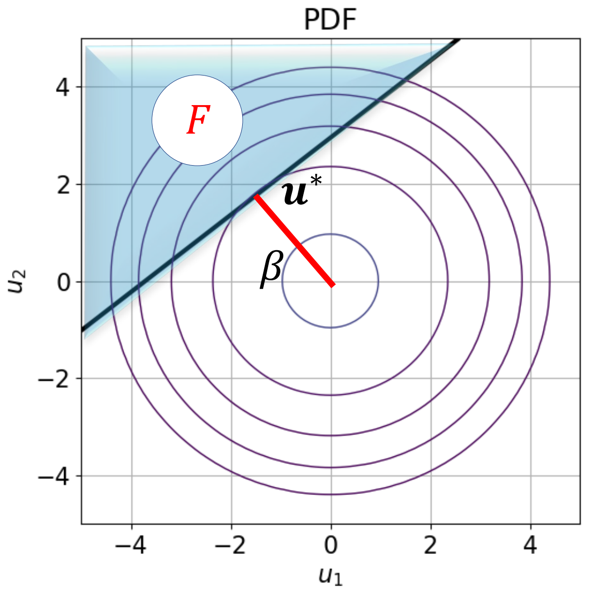
$$\begin{cases} \dfrac{\partial g}{\partial u_1} = A\sigma_1 \\ \dfrac{\partial g}{\partial u_2} = -\sigma_2 \end{cases}$$

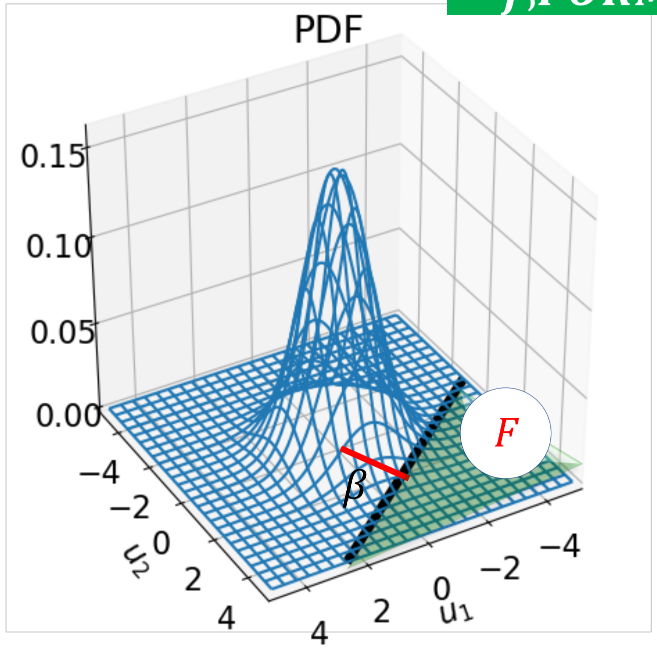
$$\boldsymbol{u}^* = \begin{Bmatrix} -1.43 \\ 1.80 \end{Bmatrix}$$

$$\beta = \|\boldsymbol{u}^*\| = 2.310$$

$$P_f = \Phi(-\beta) = 1.04 \times 10^{-2}$$

EXAMPLE B3.1: BAR, LINEAR LS, SUMMARY





$$P_{f,FORM} \equiv P_{f,MCS}$$

$$\delta = \begin{Bmatrix} 38.73 \\ 61.27 \end{Bmatrix}$$

β	$P_{f,FORM}$	β_{MCS}	$P_{f,MCS}$ $(\nu = 5\%)$ (target)	*N samples* MCS	*error* P_f
2.31	1.04×10^{-3}	2.323	1.00×10^{-2}	39,310	3.59%

EXAMPLE B3.1

OpenAIUQ

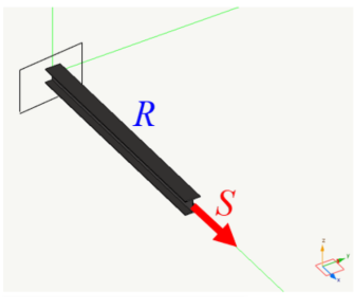
Example B3.1: Bar Example

In [1]:

```python
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



Variable		Distribution	μ	σ	ν
	A		3,142		
x_1	F_y	Normal	$329\ MPa$	$32.90\ MPa$	0.10
x_2	S	Normal	$650\ kN$	$130\ kN$	0.20

$$G(F_y, S) = AF_y - S \Rightarrow G(x_1, x_2) = Ax_1 - x_2$$

Correlation		
	X_1	X_2
X_1	1	0
X_2	0	1

Define Marginal distributions

In [2]:

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

#define mean
m1=329
#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=650000
#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]:

```python
#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 to check the joint PDF

In [4]:
```
#to check the joint distribution, generate samples (num=number of samples)
X.gen_samples(num=10) #num=number of samples
```

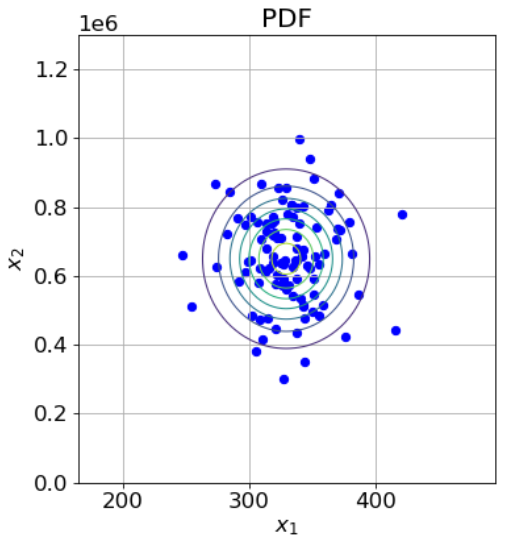
In [5]:
```
#samples in x-space
X.samples
```

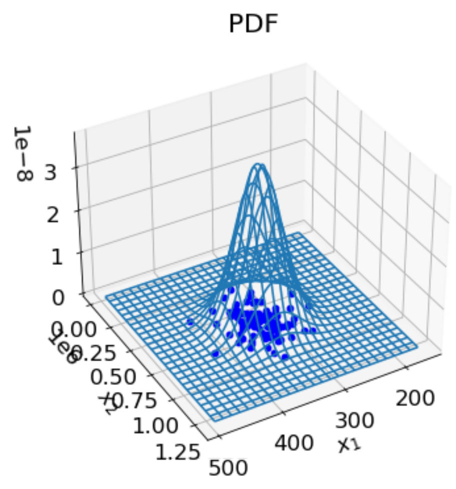
Out[5]:
```
array([[3.63461247e+02, 5.47109818e+05],
       [3.29276180e+02, 6.67750542e+05],
       [3.19703761e+02, 7.02095294e+05],
       [3.17183773e+02, 5.93319788e+05],
       [3.20622808e+02, 6.79366035e+05],
       [3.55341166e+02, 8.04139526e+05],
       [3.55244196e+02, 6.79772456e+05],
       [4.12425935e+02, 8.46374159e+05],
       [3.51397841e+02, 5.42017387e+05],
       [3.46883938e+02, 5.46189055e+05]])
```

In [6]:
```
#samples in u-space
X.samples_u
```

Out[6]:
```
array([[ 1.0474543 , -0.79146294],
       [ 0.00839454,  0.13654263],
       [-0.28256044,  0.40073303],
       [-0.35915584, -0.43600163],
       [-0.2546259 ,  0.22589258],
       [ 0.80064335,  1.18568866],
       [ 0.79769594,  0.22901889],
       [ 2.53574269,  1.51057046],
       [ 0.68078543, -0.83063548],
       [ 0.54358475, -0.79854573]])
```

In [7]:
```
#plot joint distribution
X.plot_w2(numfig=1,figsize=(12,6),dpi=60, space='x',samples=100,sec=[1,2],view=[35,6
#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

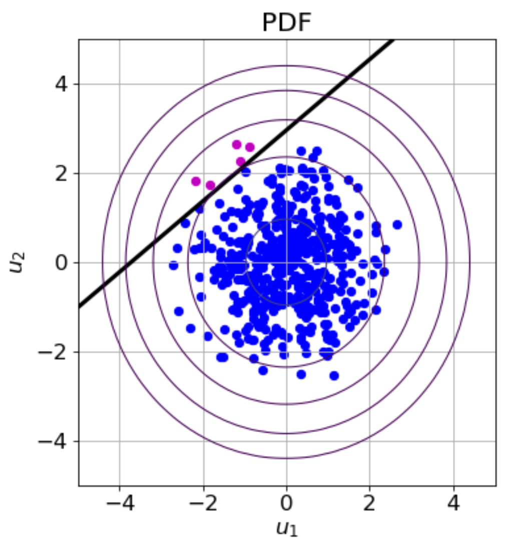
```
In [8]:  G=auq.Gfun(X,Gfun_id=1)
```

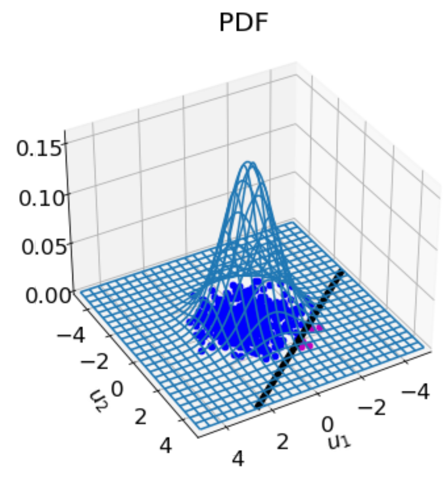
Simulate samples

```
In [9]:  G.gen_samples(X,num=10)
         G.samples
```

```
Out[9]: array([190461.74177507, 427231.25074835, 353416.91319398, 209404.65406644,
               -36651.25984947, 475518.76574369, 279595.2218905 , 680967.36480251,
               382678.1208171 , 390149.14417891])
```

```
In [10]: #plot joint distribution
         G.plot_w2(numfig=2,figsize=(12,6),dpi=60,space='u',samples=500,sec=[1,2],view=[35,60
         #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 [11]:
```
%%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, 0.007, 0.3766392741830608]
[2000, 0.0085, 0.24150264887111428]
[3000, 0.009, 0.1915821069507894]
[4000, 0.00975, 0.15934561694952779]
[5000, 0.011, 0.13409630189463906]
[6000, 0.011166666666666667, 0.12148541668719622]
[7000, 0.011571428571428571, 0.11046638343626777]
[8000, 0.010875, 0.10662669869016492]
[9000, 0.010666666666666666, 0.1015162822189404]
[10000, 0.0104, 0.09754683293364586]
[11000, 0.010272727272727272, 0.09358765106153057]
[12000, 0.01, 0.09082951062292476]
[13000, 0.010153846153846154, 0.08659581109288426]
[14000, 0.010571428571428572, 0.08176385622833714]
[15000, 0.010866666666666667, 0.0778993110270348]
[16000, 0.011125, 0.07453507582400308]
[17000, 0.011117647058823529, 0.07233382169907467]
[18000, 0.011111111111111112, 0.07031674369909663]
[19000, 0.010894736842105263, 0.06912514951483842]
[20000, 0.0109, 0.0673584141982481]
[21000, 0.010761904761904762, 0.06616010666136278]
[22000, 0.01040909090909091, 0.06573703432327693]
[23000, 0.010347826086956521, 0.06448412526660607]
[24000, 0.01025, 0.06343007160440414]
[25000, 0.01008, 0.0626757845443834]
[26000, 0.010269230769230770, 0.06088396187501587]
[27000, 0.010259259259259259, 0.05977517265731702]
[28000, 0.010214285714285714, 0.058828472789876254]
[29000, 0.010448275862068965, 0.05714759202610912]
[30000, 0.0105, 0.05604703240377533]
[31000, 0.010677419354838710, 0.0546707414595563]
[32000, 0.01065625, 0.05386372916581337]
[33000, 0.010606060606060607, 0.05316803388164572]
[34000, 0.010911764705882353, 0.0516333809182735]
[35000, 0.010771428571428570, 0.05122449087426615]
[36000, 0.01075, 0.0505588919022915]
[37000, 0.010648648648648649, 0.050110318687500086]
Wall time: 3.58 s
```

In [12]:
```
print('Pf=',MCS.Pf)
print('beta=',MCS.beta)
print('cov=',MCS.cov)
print('samples=',MCS.num)
```

```
Pf= 0.010680763836444061
beta= 2.3015329050620967
cov= 0.0499828060377058
samples= 37076
```

FORM

In [13]:
```
FORM=auq.FORM(G)
```

```
print('FORM')
print('-------------------')
FORM.FORM()
```

```
FORM
-------------------
..................................

Now carrying out iteration number 1
Value of the limit state function in the first step 383718
e1 = 1.0 , e2 = 0.0

..................................

Now carrying out iteration number 2
e1 = 1.941681285976636e-14 , e2 = 6.332165561495272e-14

The step size has been reduced by a factor of 1/1048576 before continuing
```

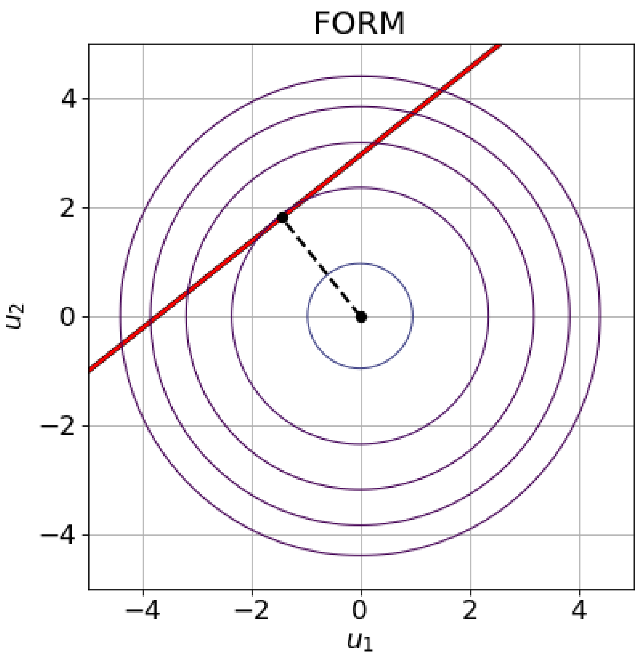
In [14]:
```
print('Pf=',FORM.Pf)
print('beta=',FORM.beta)
print('uMPP=',FORM.uMPP)
print('xMPP=',FORM.xMPP)
print('alpha=',FORM.alphaMPP)
```

```
Pf= 0.01043548480180217
beta= 2.3103105077869106
uMPP= [-1.43790502  1.80830412]
xMPP= [2.81692925e+02 8.85079536e+05]
alpha= [-0.62238604  0.78271043]
```

In [15]:
```
print('alpha sens=',FORM.alphasens)
```

```
alpha sens= [38.73643876 61.26356124]
```

In [16]:
```
#%%time
FORM.plot(numfig=3,figsize=(6,6),dpi=60, space='u',plot_type='contour',sec=[1,2],lim
#space= 'x','u'
#plot_type='contour','3d'
#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)
```



```
In [17]: err=100*(FORM.Pf-MCS.Pf)/MCS.Pf
         print(err)
```

```
-2.296455931409786
```

Thank you - Grazie

Umberto Alibrandi

Email: ua@cae.au.dk

Tel: 2296 7726

AARHUS
UNIVERSITY