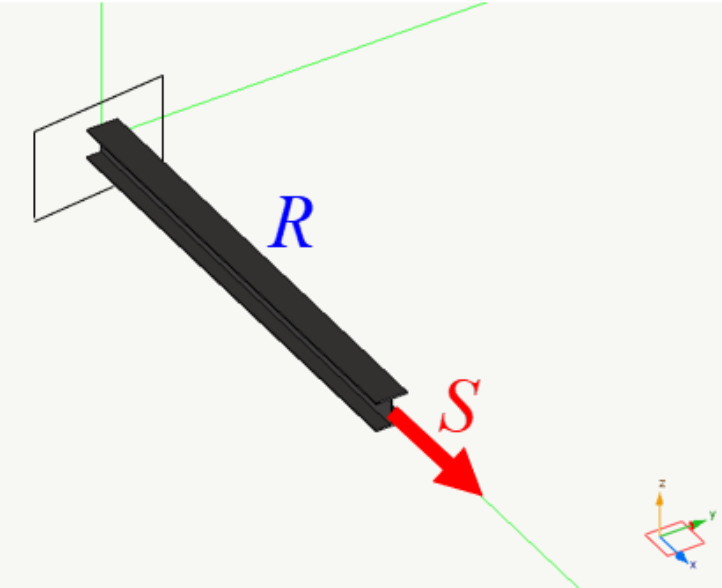
RISK AND RELIABILITY IN ENGINEERING

PART B: STRUCTURAL RELIABILITY

Example B3.2: Bar, Nonlinear Limit State

Umberto Alibrandi
Department of Engineering
Aarhus University

EXAMPLE B3.2: BAR, NONLINEAR LS



Variable		Distribution	μ	ν
x_1	A	Normal	2,534	0.05
x_2	F_y	Normal	329 MPa	0.10
	S		650 kN	

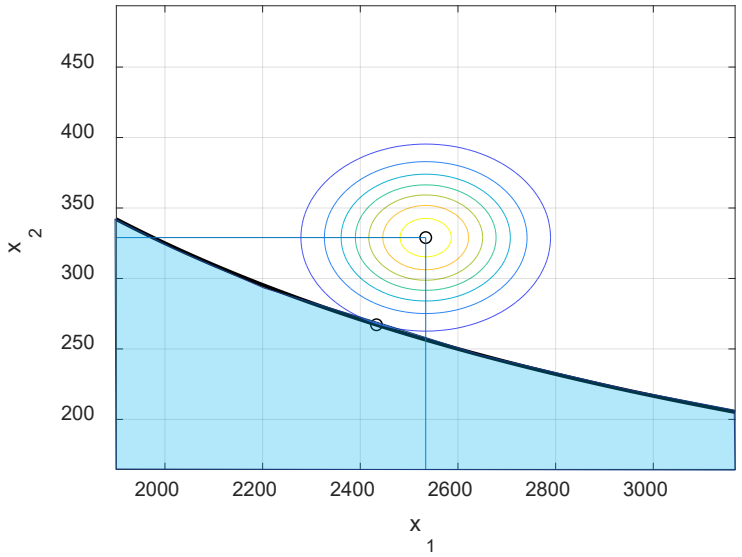
$$P_f = Prob[AF_y \leq S]$$

Correlation		
	X_1	X_2
X_1	1	0
X_2	0	1

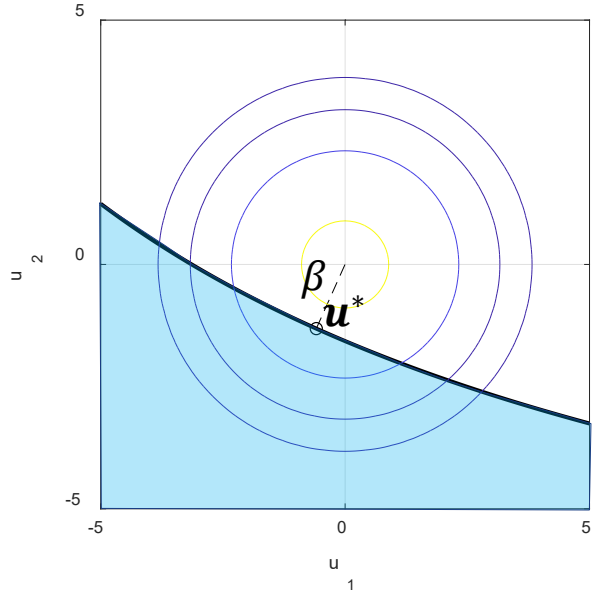
$$G(A, F_y) = AF_y - S$$

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

EXAMPLE B3.2: BAR, NONLINEAR LS



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



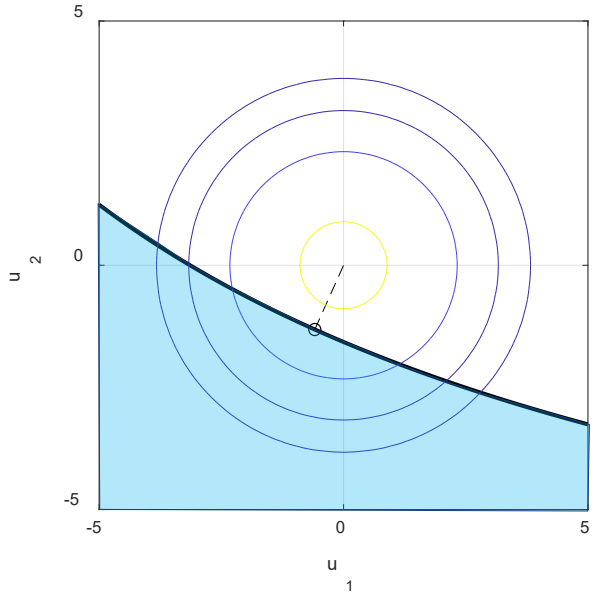
$$G(x_1, x_2) = x_1 x_2 - S$$

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

$$= \sigma_1\sigma_2 u_1 u_2 + \mu_2\sigma_1 u_1 + \mu_1\sigma_2 u_2 + \mu_1\mu_2 - S$$

EXAMPLE B3.2: BAR, NONLINEAR LSF

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



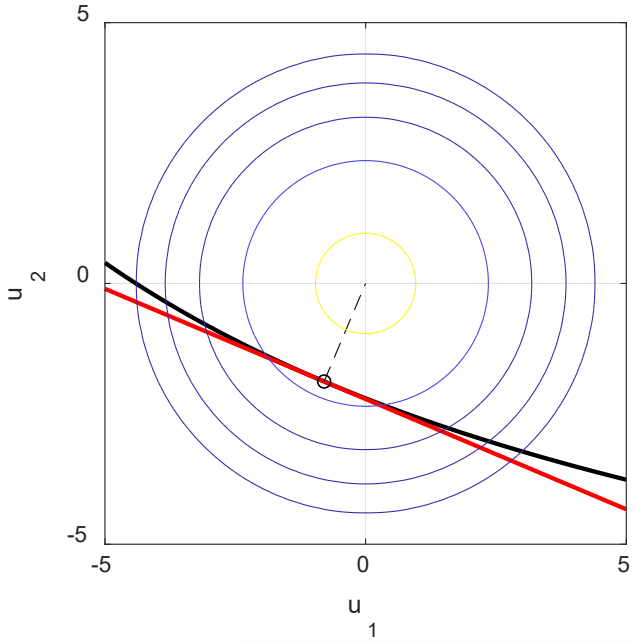
$$\begin{cases} \dfrac{\partial g}{\partial u_1} = \sigma_1(\mu_2 + \sigma_2 u_2) = \sigma_1 x_2 \\ \dfrac{\partial g}{\partial u_2} = \sigma_2(\mu_1 + \sigma_1 u_1) = \sigma_2 x_1 \end{cases}$$

$$\boldsymbol{u}^* = \begin{Bmatrix} -0.795 \\ -1.880 \end{Bmatrix}$$

$$\beta = \|\boldsymbol{u}^*\| = 2.042$$

$$P_f = \Phi(-\beta) = 2.06 \times 10^{-2}$$

EXAMPLE B3.2: BAR, NONLINEAR LSF FORM

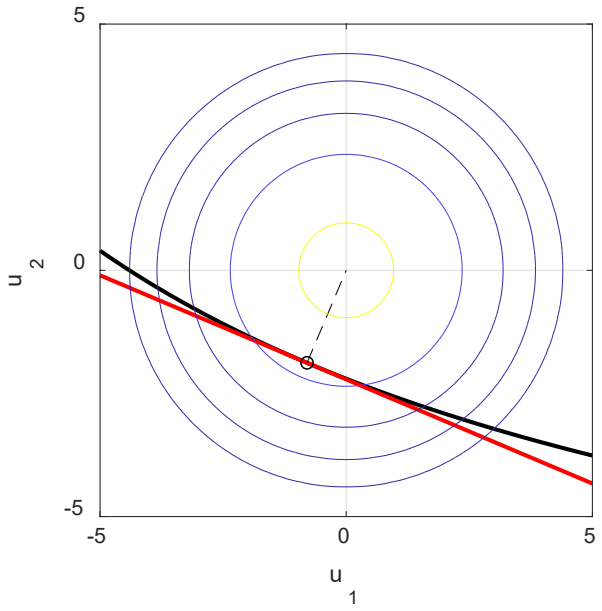


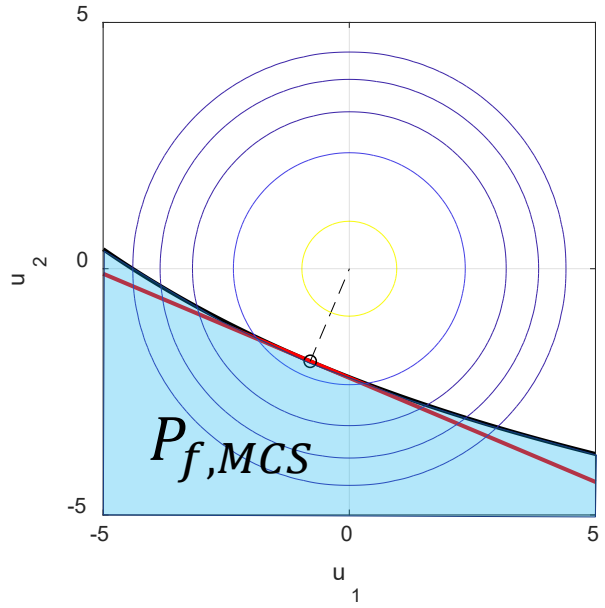
Profile	Distribution		β	u^*	x^*	x_m	δ
HE120A	x1	Normal	2.04	-0.80	2433.20 mm^2	2,534 mm^2	15.16%
	x2	Normal		-1.88	267.14 MPa	329 MPa	84.84%

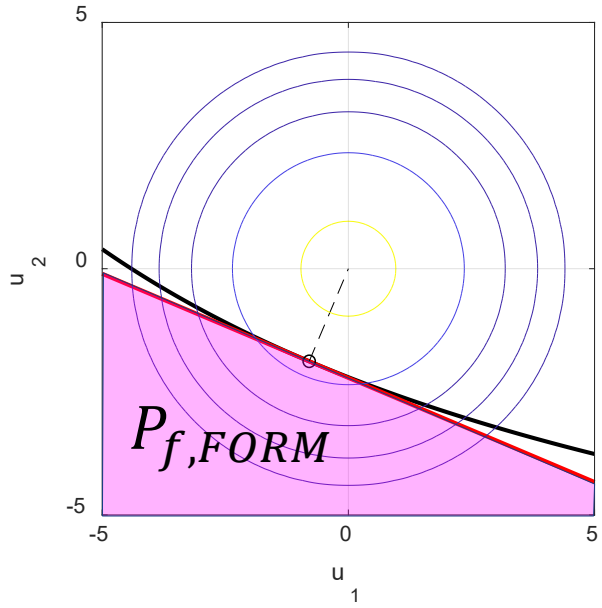
Profile	β	$P_{f,FORM}$	β_{MCS}	$P_{f,MCS}$ ($\nu = 5\%$) (target)	N samples MCS	error β	error P_f
HE120A	2.042	2.06×10^{-2}	2.021	2.16×10^{-2}	544,409	1.02%	-4.84%

EXAMPLE B3.2: BAR, NONLINEAR LS SUMMARY

$$P_{f,FORM} < P_{f,MCS}$$







Profile	β	$P_{f,FORM}$	β_{MCS}	$P_{f,MCS}$ ($\nu = 5\%$) (target)	N samples MCS	error β	error P_f
HE120A	2.042	2.06×10^{-2}	2.021	2.16×10^{-2}	544,409	1.02%	-4.84%

EXAMPLE B3.2

OpenAIUQ

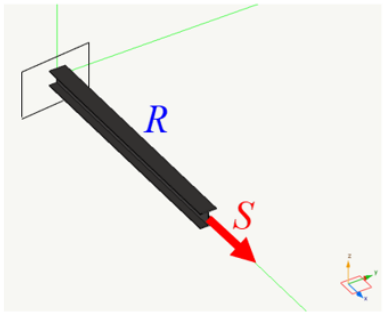
Example B3.2: Bar Example, nonlinear LS

In [1]:
```
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

EXAMPLE B3.2: BAR, NONLINEAR LS



Variable		Distribution	μ	ν
x_1	A	Normal	2,534	0.05
x_2	F_y	Normal	329 MPa	0.10
	S		650 kN	

$$P_f = Prob[AF_y \leq S]$$

Correlation		
	X_1	X_2
X_1	1	0
X_2	0	1

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

Define Marginal distributions

In [2]:
```
#=======================
#variable x1

#define mean
m1=2534
#define standard deviation
s1=0.05*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=329
#define standard deviation
s2=0.10*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 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
```

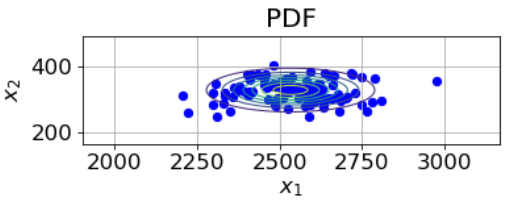
In [5]:
```
#samples in x-space
X.samples
```

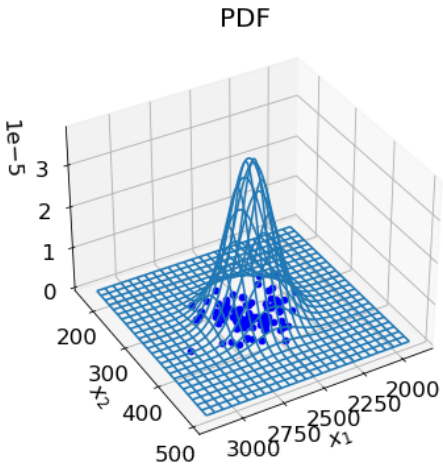
Out[5]:
```
array([[2494.31787272,  256.24861878],
       [2819.24213247,  321.90661808],
       [2527.18079001,  352.18343981],
       [2575.92458799,  400.38732923],
       [2501.79955935,  326.14599457],
       [2387.73798168,  349.07827642],
       [2454.78394646,  309.01062592],
       [2773.68725965,  384.44514984],
       [2612.75587418,  313.74326551],
       [2356.37219481,  318.00343671]])
```

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

Out[6]:
```
array([[-0.31319753, -2.21128818],
       [ 2.25131912, -0.21560431],
       [-0.0538217 ,  0.70466382],
       [ 0.33089651,  2.16982764],
       [-0.25414712, -0.08674789],
       [-1.15439636,  0.61028196],
       [-0.62522536, -0.60757976],
       [ 1.89177001,  1.68526291],
       [ 0.62159332, -0.46373053],
       [-1.40195584, -0.33424205]])
```

In [8]:
```
#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

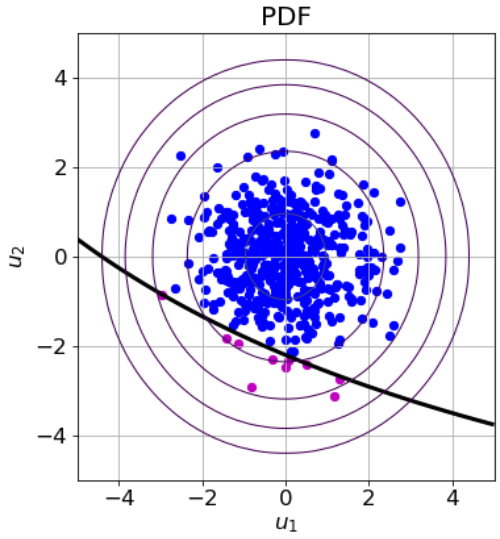
In [9]:
```
G=auq.Gfun(X,Gfun_id=1)
```

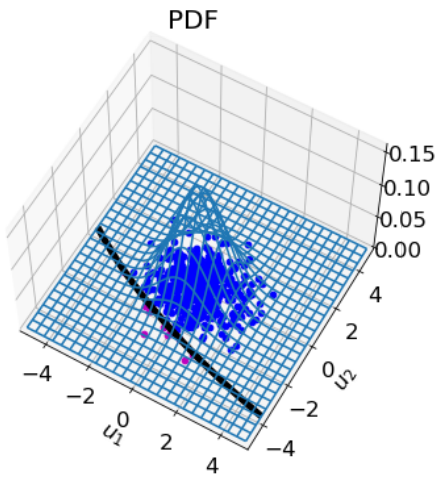
Simulate samples

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

Out[10]:
```
array([103441.25425706, 254709.06132029,  77164.50844401, 249589.5392381 ,
       206597.5745135 , 264625.56139374,  38556.55594012, 145204.41146364,
       268971.78133528, 136763.90468971])
```

In [12]:
```
#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 [13]:
```
%%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, 0.017, 0.24046523534965444]
[2000, 0.018, 0.16515985522449994]
[3000, 0.02, 0.12780193008453875]
[4000, 0.021, 0.1079572225687652]
[5000, 0.0216, 0.09518014109707582]
[6000, 0.021, 0.0881467031385031]
[7000, 0.020428571428571428, 0.08276563205914547]
[8000, 0.019875, 0.07851310830747073]
[9000, 0.02011111111111111, 0.07357819491766517]
[10000, 0.0213, 0.06778521727302475]
[11000, 0.021272727272727273, 0.06467298649819091]
[12000, 0.021666666666666667, 0.06134183330175674]
[13000, 0.02176923076923077, 0.05879324644608905]
[14000, 0.021785714285714287, 0.056632675666628995]
[15000, 0.021933333333333332, 0.054523820584629644]
[16000, 0.02175, 0.05301946075160332]
[17000, 0.02164705882352941, 0.05156130113695795]
Wall time: 1.8 s
```

In [14]:
```
print('Pf=',MCS.Pf)
print('beta=',MCS.beta)
print('cov=',MCS.cov)
print('samples=',MCS.num)
```

```
Pf= 0.021825065419520073
beta= 2.017435067255699
cov= 0.04995342051219887
samples= 17961
```

FORM

In [15]:
```
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 183686
e1 = 1.0 , e2 = 0.0

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

Now carrying out iteration number 2
e1 = 0.03525279301803106 , e2 = 0.11193216013808752

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

Now carrying out iteration number 3
```

```
e1 = 0.00018759169492193963 , e2 = 0.01299365058180723

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

Now carrying out iteration number 4
e1 = 1.292768590892795e-06 , e2 = 0.0008865137237080691
```

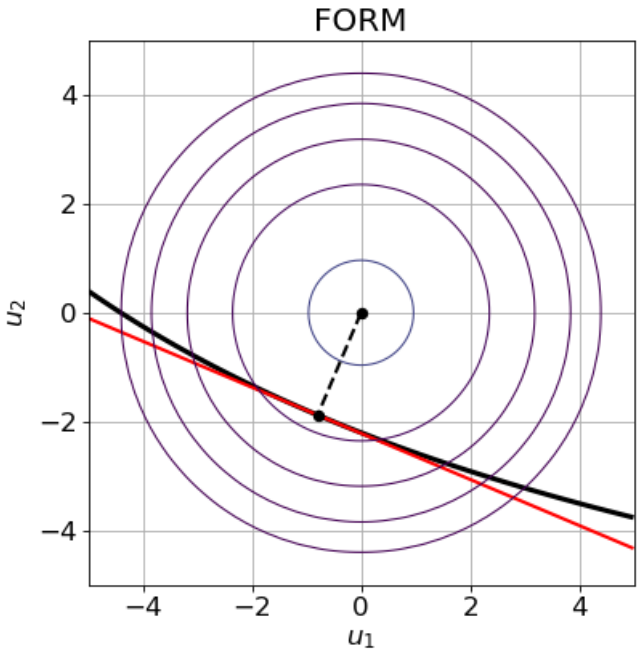
In [16]:
```
print('Pf=',FORM.Pf)
print('beta=',FORM.beta)
print('uMPP=',FORM.uMPP)
print('xMPP=',FORM.xMPP)
print('alpha=',FORM.alphaMPP)
```

```
Pf= 0.020591150876981327
beta= 2.0416898950788203
uMPP= [-0.79583506 -1.88019808]
xMPP= [2433.16769829  267.14148329]
alpha= [-0.3894382 -0.9210526]
```

In [17]:
```
print('alpha sens=',FORM.alphasens)
```

```
alpha sens= [15.16621091 84.83378909]
```

In [18]:
```
#%%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 [19]:
```
err=100*(FORM.Pf-MCS.Pf)/MCS.Pf
print(err)
```

```
-5.6536579333006145
```

Thank you - Grazie

Umberto Alibrandi

Email: ua@cae.au.dk

Tel: 2296 7726

AARHUS
UNIVERSITY