PHIMECA

… solutions for robust engineering

OpenTURNS

A. Dumas, Phimeca Engineering SA

'HPC and Uncertainty Treatment – Examples with Open TURNS and Uranie'

EDF – Phimeca – Airbus Group – IMACS – CEA

PRACE Advanced Training Center – May, 16-18 2018

PRACE

MAISON DE LA SIMULATION

Outline

- Overview of OpenTURNS
  - What is OpenTURNS?
  - Uncertainty methodology
  - OpenTURNS features
  - Uncertainty quantification with OpenTURNS
  - Innovations
- OpenTURNS: Doc and Users
- OpenTURNS in pictures
- OpenTURNS in practice
  - OpenTURNS: Basics and example

Outline

- Overview of OpenTURNS
  - What is OpenTURNS?
  - Uncertainty methodology
  - OpenTURNS features
  - Uncertainty quantification with OpenTURNS
  - Innovations
- OpenTURNS: Doc and Users
- OpenTURNS in pictures
- OpenTURNS in practice
  - OpenTURNS: Basics

What is Open TURNS?

- Partnership since 2005 between:

  EDF - R&D **EADS - Innovation Works** **Phimeca** **IMACS** (since 2014)

- **Open** source initiative to **T**reat Uncertainties, **R**isks'**N** **S**tatistics
  - An *open source* platform dedicated to *uncertainty treatment* in support of probabilistic methods
  - *Uncertainty propagation* through a model up to a variable of interest
  - *Uncertainty quantification* and *Uncertainty ranking*
  - *Meta-model* building
  - working on Unix/Linux platform and Windows (since 2010)

- Open TURNS includes:
  - *C++ scientific library* including the methods for performing uncertainties treatment (statistic, reliability, etc.);
  - A *python module* allowing to define in a simple manner the models in an interpreted language;
  - A complete documentation;
  - A website: www.openturns.org.

Uncertainty methodology (1/2)

Global Methodology of Treatment of Uncertainties

- developed first at EDF R&D in 1990 and then improved by contributions from other companies

Step A : *Study Specification*

Uncertainty sources, model, variable of interest and criteria

Step B : *Uncertainty Quantification*

Joint probability density function of the input uncertain parameters modeling

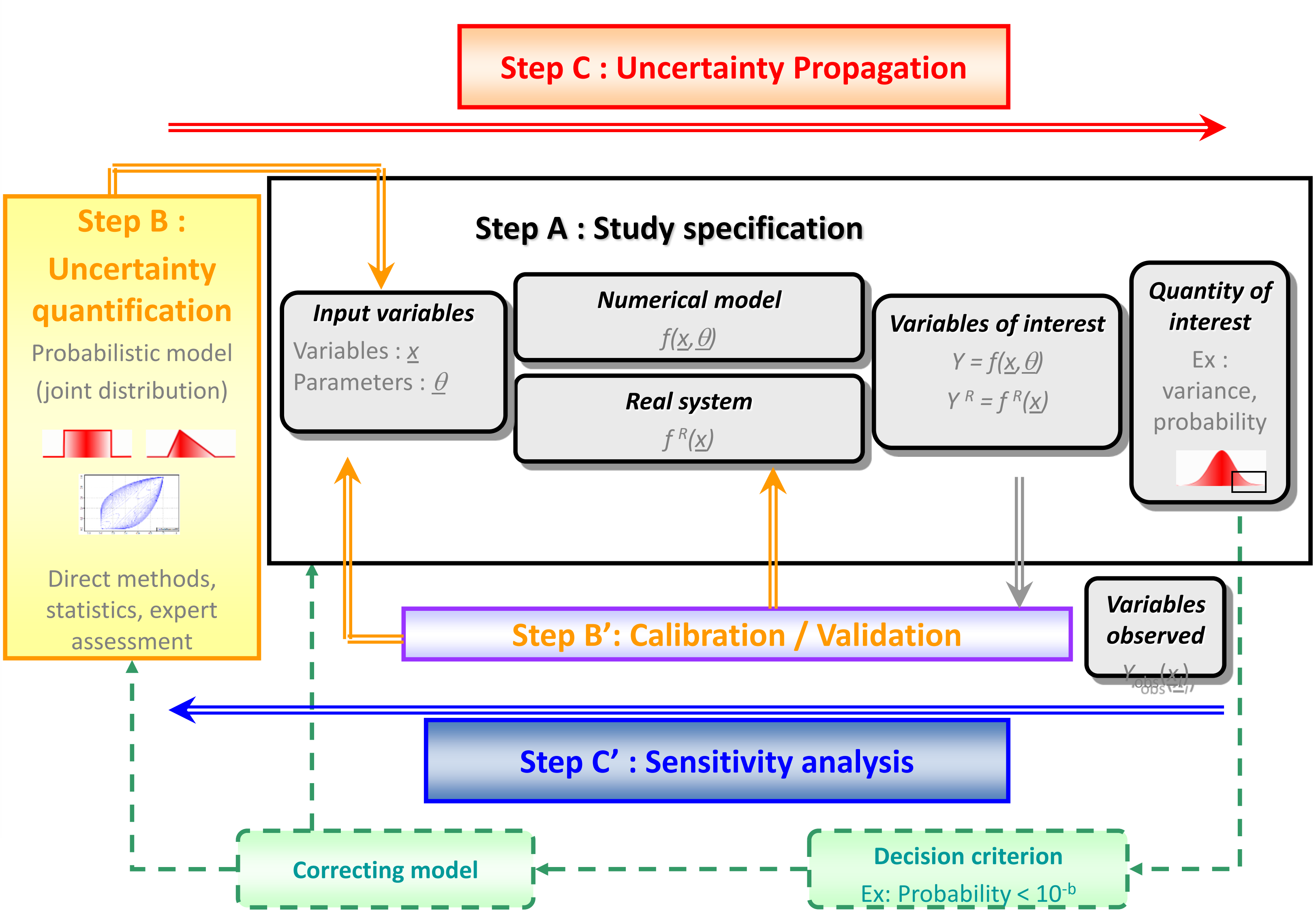
Step C : *Uncertainty Propagation*

Variable of interest uncertainty assessment

Step C' : *Uncertainty Ranking / sensitivity analysis*

Uncertainty sources ranking with respect to their influence on the variable of interest uncertainty

Uncertainty methodology (2/2)

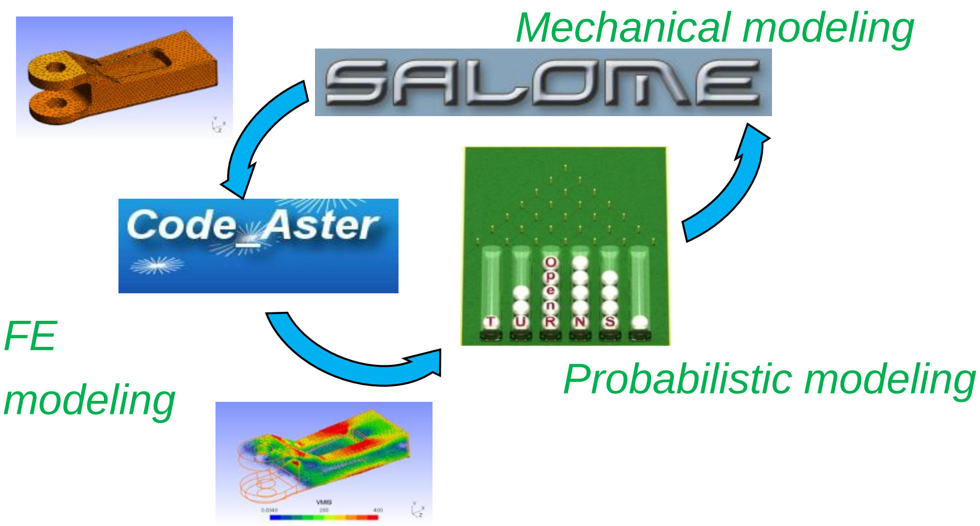


PHIMECA

OpenTURNS features

Code linked to OpenTURNS

- *Interface with python functions* ➔ to perform complex wrappers without compilation + parallelization functionalities
- *Standard Interface* for the *wrappers of any complexity* (distributed wrapper, binary data) development requiring the development of an external wrapper
- *SalomeMeca compatible* ➔ software including the 3 components to perform a mechanical and probabilistic data models coupling (linked to YACS)
- GUI of OpenTURNS within SalomeMeca



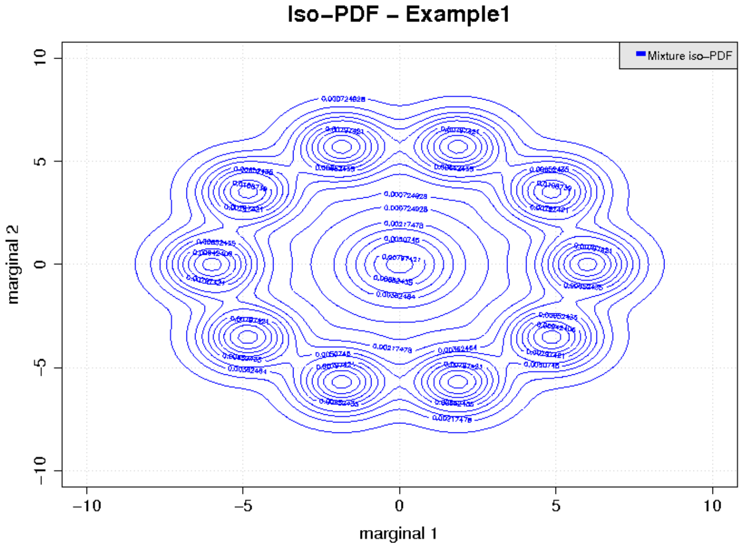
Uncertainty quantification with OpenTURNS

Estimation from data :

- Distribution fittings (parametric or not)
- Validation Tests (quantitative or graphical)
- Estimation of the dependence : copula, correlation coefficient
- Regression

Analytical modeling of joint distributions of dimension n :

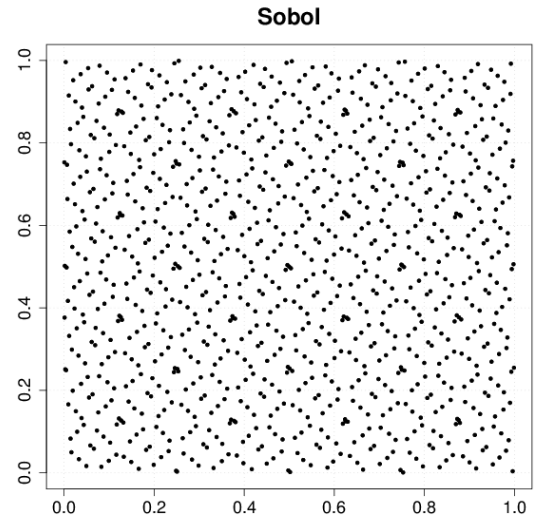
- Combination Marginals + Copula
- Parametric distributions of dimension n (normal, student...)
- Truncated distributions
- Stochastic process
- Non parametric distribution of dimension n: kernel fitting (n), Sklar Copula
- Linear combination of PDF
- Linear combination of random variables
- Random sum of independent discrete variables according to a Poisson process
- Etc.



Uncertainty propagation with OpenTURNS

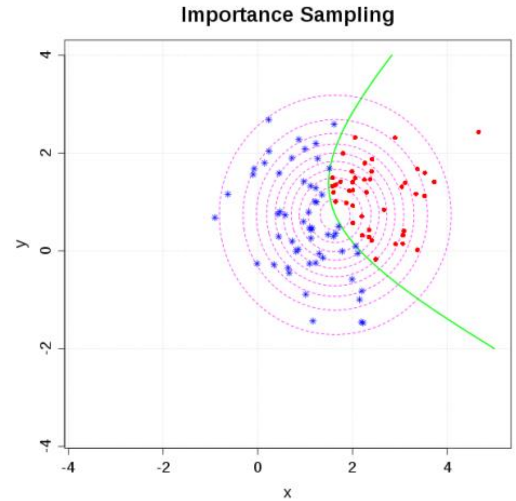
Sampling data:

- Random generator
- Stratified design of experiment
- Latin Hypercube Sampling
- Low Discrepancy Sequence
- Markov chain



Probability estimation:

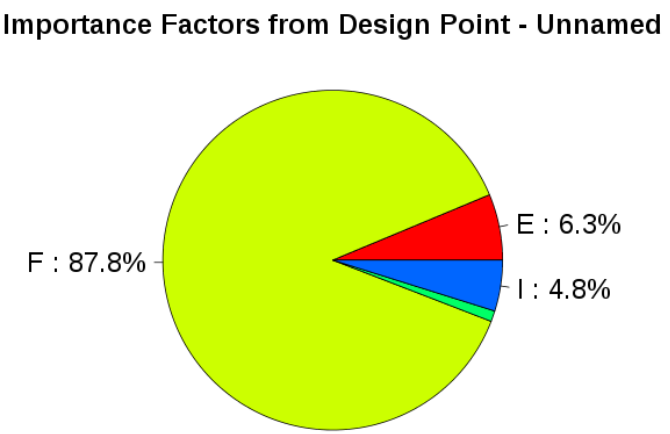
- Isoprobabilistic transformation
- FORM / SORM
- Monte Carlo simulation
- Importance simulation
- Directional simulation
- Latin hypercube simulation
- Simulation algorithms



Uncertainty ranking with OpenTURNS

Ranking and sensitivity analysis:

- Importance factor from Taylor decomposition
- Ranking from correlation
- Sensitivity analysis
- Importance factor from reliability methods



Tools

- Optimization algorithm
- Response surface:
  - Parametric approximation
  - Functional chaos expansion
  - Kriging
- Graph

and Modules…

Innovative and recently implemented algorithms

- the most recent and efficient algorithms of non uniform distribution generation
  - Ziggurat method (2005) for the normal distribution
  - sequential reject algorithm (1993) for the binomial distribution,
  - Tsang & Marsaglia method (2000) for the gamma distribution,
  - Lebrun algorithm (2012) for the MultiNomial distribution,

- the most recent algorithms for evaluating the CDF
  - Marsaglia algorithm for the exact statistics of Kolmogorov (2003),
  - Benton et Krishnamoorthy algorithm for the distributions non centered Student and non centered Chi2 (2003).

- PhD results
  - Sparse chaos expansion polynomials : G. Blatman (EDF/R&D/MMC) (2010)
  - Accelerated simulation algorithm for the evaluation of low probabilities : M. Munoz (EDF/R&D/MRI) : (current dev)
  - Copulas for order statistics distributions: R. Lebrun (EADS) , Richard Fischer (EDF) (2013)

Innovation

- The internal data models is based on the *multidimensional cumulative density function*

- *the sampling approach* : estimation of the output variable statistical characteristics from a large numerical sample

- and *the analytical approach* : exact partial or total solving of some problems using the probabilistic modeling and the implementation in Open TURNS of the function $\mathbb{R}^n \longrightarrow \mathbb{R}^p$ algebra up to the order 2:
  - exact determination of *the distribution of the sum of random variables* thanks to the characteristic functions implemented for each distribution
  - exact representation of the distribution of a random variable such that *its pdf is a linear combination of pdf*
  - exact determination of the distribution of a random variable defined as *the random sum according to a Poisson process of iid discrete random variables* (for ex, cumulated failure times when failures follow a Poisson process)
  - *particular manipulations on distributions* : extraction of marginals, extraction of dependence structure, etc.

Outline

- Overview of OpenTURNS
  - What is OpenTURNS?
  - Uncertainty methodology
  - OpenTURNS features
  - Uncertainty quantification with OpenTURNS
  - Innovations
- OpenTURNS: Doc and Users
- OpenTURNS in pictures
- OpenTURNS in practice
  - Open TURNS: Basics

OpenTURNS: Doc and Users

- *Several guides* intended for users
  - **Installation** : on windows, linux, from anaconda, from sources
  - **API Reference (Sphinx documentation)** : Python docstring of most of the *objects, arguments and methods* in OpenTURNS and available in HTML.
  - **Examples Guide** : application of the whole Global Methodology on *classical mechanical examples*
  - **Reference Guide** : *Theory* of the methods implemented within OpenTURNS
  - **Contribute** : how to contribute to OpenTURNS, core code, modules …

- *… and a sympathetic community :*
  - Openturns.org : official web site
  - a particular page share to communicate about the software
  - the annual Users Day

OpenTURNS: Doc and Users

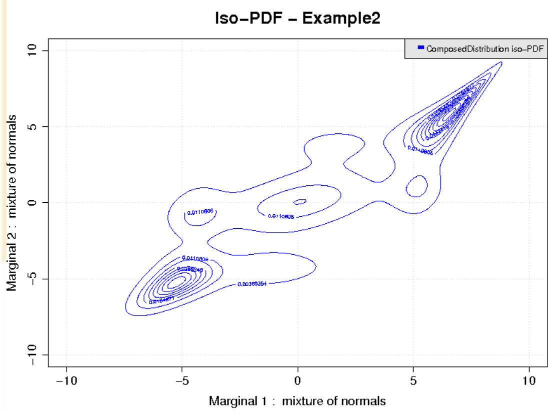
- Mailing list for users: users@openturns.org:
  - Ask any question relative to the installation and use of Open TURNS

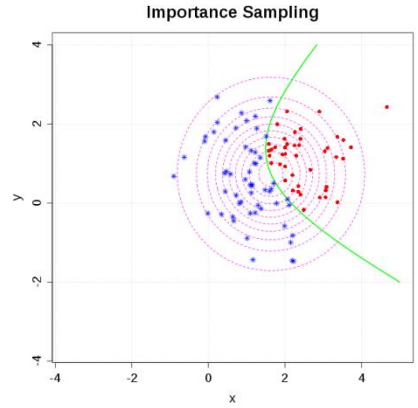
- Bug tracking :
  - http://trac.openturns.org/wiki
  - Communication about the correction of already identified bugs in the Trac and the new ones.

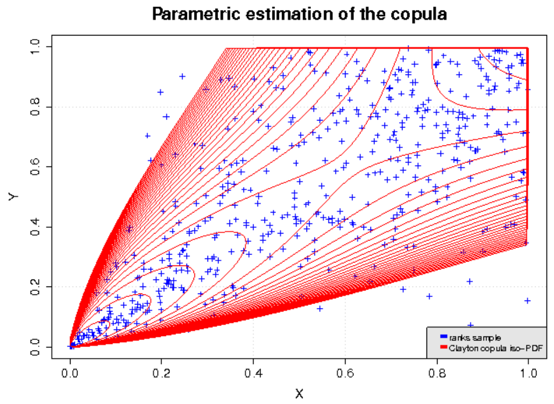
Outline

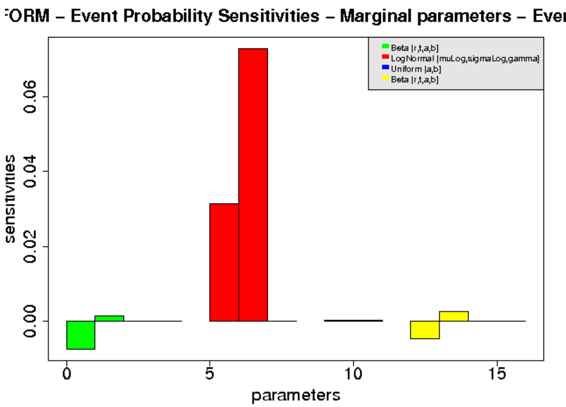
- Overview of OpenTURNS
  - What is OpenTURNS?
  - Uncertainty methodology
  - OpenTURNS features
  - Uncertainty quantification with OpenTURNS
  - Innovations
- OpenTURNS: Doc and Users
- **OpenTURNS in pictures**
- OpenTURNS in practice
  - OpenTURNS: Basics

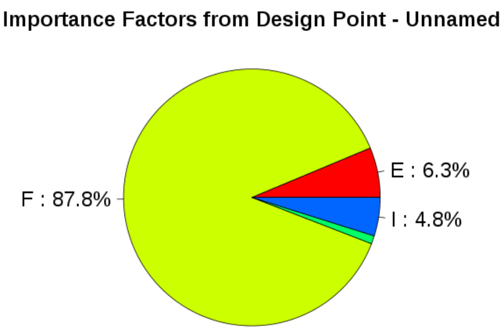
OpenTURNS in pictures

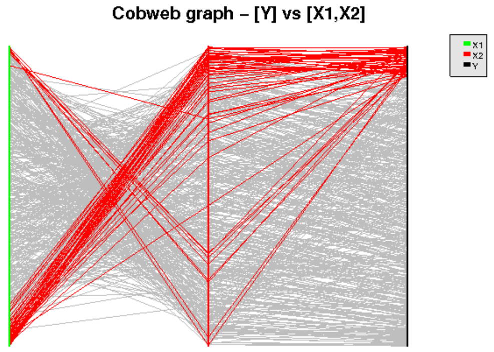






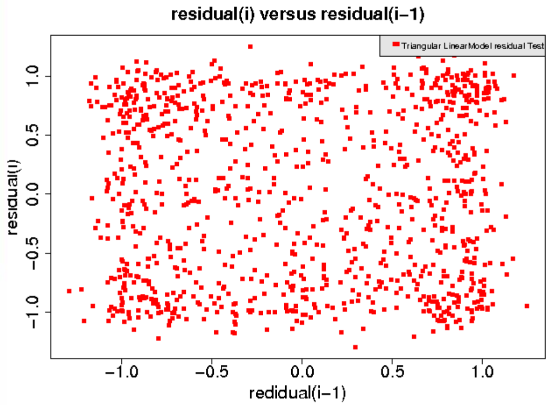


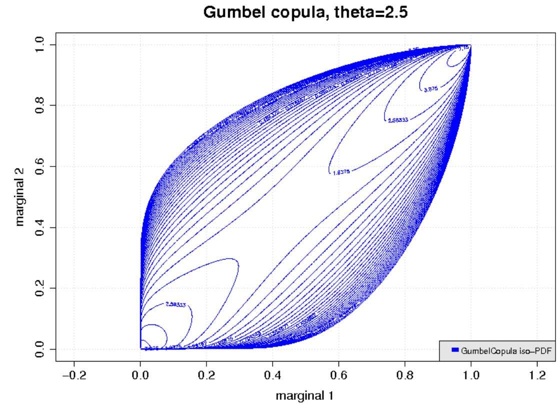


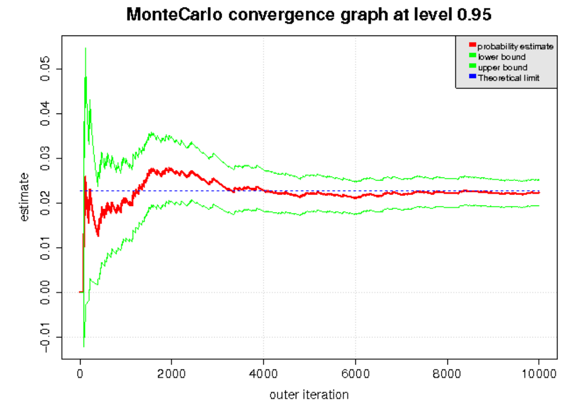


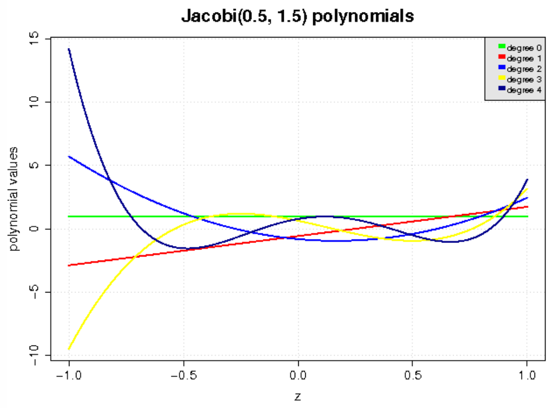
© Phimeca Engineering

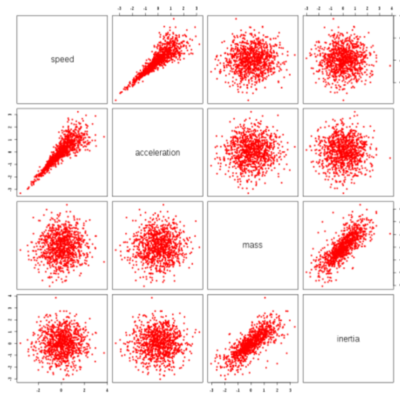
OpenTURNS in pictures

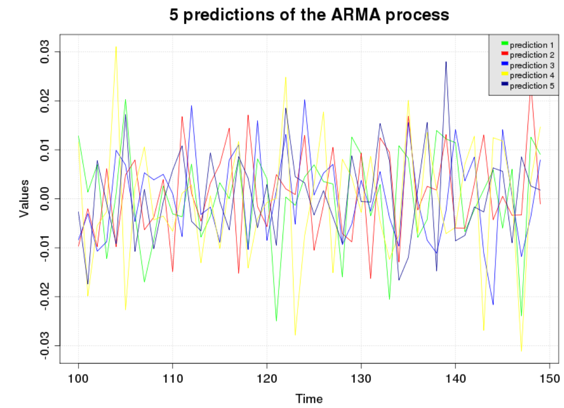






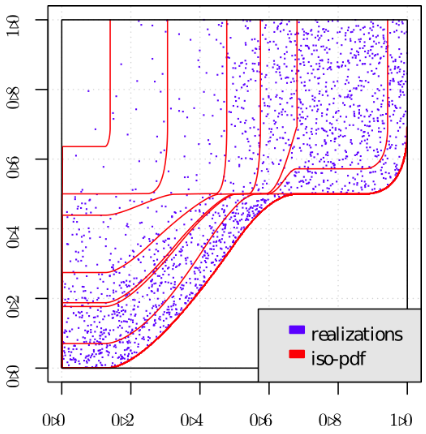


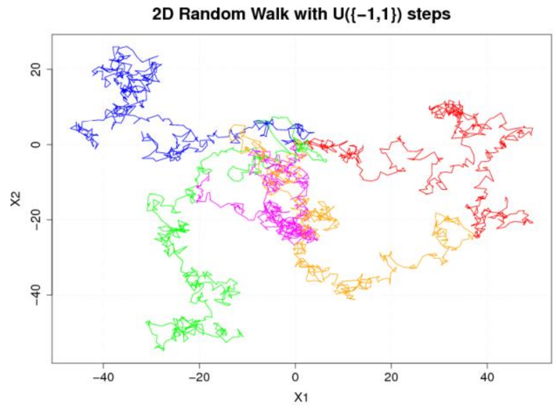


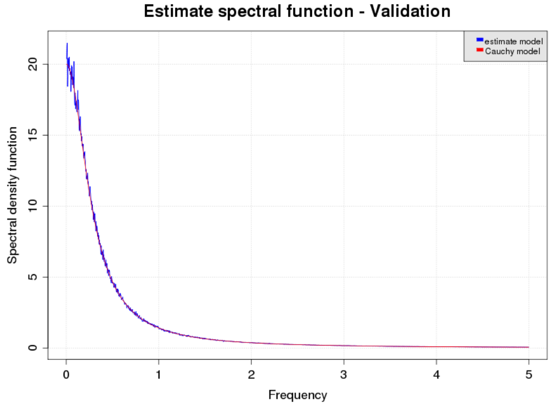


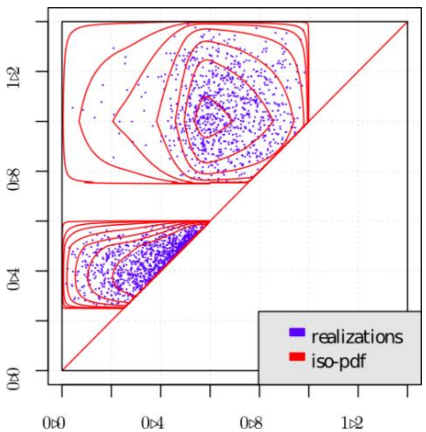
PHIMECA

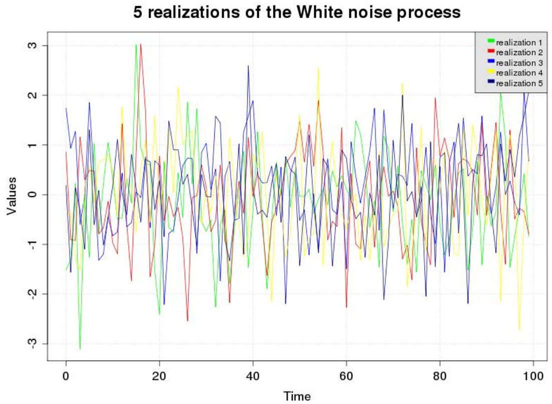
OpenTURNS in pictures

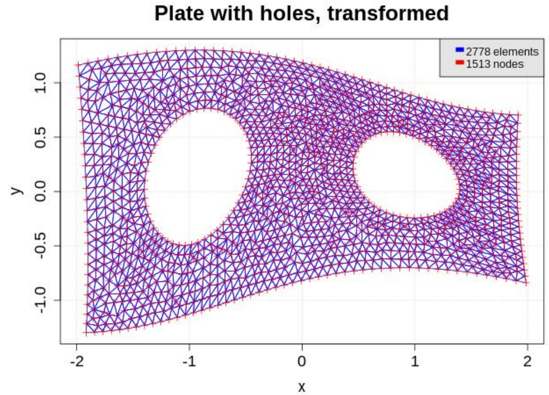












Outline

- Overview of OpenTURNS
  - What is OpenTURNS?
  - Uncertainty methodology
  - OpenTURNS features
  - Uncertainty quantification with OpenTURNS
  - Innovations
- OpenTURNS: Doc and Users
- OpenTURNS in pictures
- OpenTURNS in practice
  - OpenTURNS : Basics

Basic commands in OT (1/3)

Importation of OpenTURNS functionalities

```
>import openturns as ot          # import openturns module with an alias, here ot
```

Mathematical objects

```
>a = ot.Point(3)        # Vector of 3 components of dimension 1
> S = ot.Sample(2, 3)  # Vector of 2 components of dimension 3
>b = ot.Matrix(5 ,7)              # Matrix with 5 rows and 7 columns
>d = ot.Tensor(3, 4, 5)                # Tensor à 3 rows, 4 columns et 5 pages
>d[2, 1, 3] = -2.0                     # assign the value -2 to the 3rd row, 2nd column and
                                       #4th page,  of the tensor d
```

Basic commands in OT (2/3)

Methods

```
>a = ot.Point(3)
>a[0] = 2.
>a[1] = -3.
>a[2] = 5.
>norm_a = a.norm()                                      # Euclidean norm of the vector a


>mat = ot.SquareMatrix(2)                                   # Squared matrix of order 2
>mat[0, 0] = -2.
>mat[0, 1] = 3.
>mat[1, 0] = 0.
>mat[1, 1] = 1.
>det_mat = mat.computeDeterminant()                         # Determinant of the matrix mat


>y = ot.Point(2)
>y[0] = 1.
>y[1] = 5.
>x = mat.solveLinearSystem(y)                           # Solve the system mat*x=y
```

Basic commands in OT (3/3)

Methods

```
mat = ot.SquareMatrix(2)
mat[0, 0] = -2.
mat[0, 1] = 3.
mat[1, 0] = 0.
mat[1, 1] = 1.
det_mat = mat . computeDeterminant ()
```

« () » at the end of the method

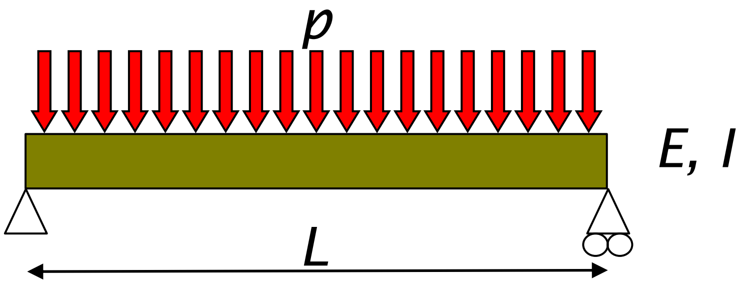
Objet relative to the method

Name of the method:
- begins with a a lower case (*compute*)
- if it is composed of several words the following begin with a capital (*Determinant*)

« . » between the objet and the method

Example of uncertainty propagatio

Bending beam under uniform loading



Maximal displacement

$$v_{\max} = \frac{5}{384}\frac{pL^4}{EI}$$

Probabilistic model

parameter	Symbol	Distribution	Mean	Standard Deviation
Length [mm]	L	Lognormal	5000	50
Young modulus [MPa]	E	Lognormal	30000	4500
Inertia [mm^4]	I	Lognormal	10^9	10^8
Load [N/mm]	p	Lognormal	10	3

Model function

Defining a Function

```
class myfunction(ot.OpenTURNSPythonFunction):

    def __init__(self):
        ot.OpenTURNSPythonFunction.__init__(self, 4, 1)

    def _exec(self, X):
        dep_max = 5.0 / 384.0 * X[3] * X[0] ** 4 / (X[1] * X[2])
        return [dep_max]

depmax = ot.Function(myfunction())

# ou

Depmax = ot.SymbolicFunction(['x1', 'x2', 'x3', 'x4'],
                             ['5. / 384 * x4 * x1 / (x2 * x3)'])
```

Defining the derivatives

Centered Finite difference

```
pas = ot.Point(4)
pas[0] = 5.0
pas[1] = 30.0
pas[2] = 1.0e6
pas[3] = 0.01

myGradient = ot.CenteredFiniteDifferenceGradient(pas,
                                    depmax.getEvaluation ())
myHessian = ot.CenteredFiniteDifferenceHessian(pas,
                                    depmax.getEvaluation())

depmax.setGradient(myGradient)
depmax.setHessian(myHessian)
```

Defining the probabilistic model

1 - Define the marginals

```
moy = ot.Point([5000.0, 300000, 1.0e9, 10.0])

et = ot.Point([50.0, 4500.0, 1.0e8, 3.0])

binf = 0.0

L1 = ot.LogNormalMuSigma(moy[0], et[0], binf).getDistribution()
L1.setName("L")
...
```

Defining the probabilistic model

2 – Define the correlation

The variables are independent

→ Use the independent copula.

```
Copula = ot.IndependentCopula(4)
```

Defining the probabilistic model

3 – Define the composed distribution

- Collection of distributions:

```
Collection = ot.DistributionCollection(4)
Collection[0] = ot.Distribution(L1)
Collection[1] = ot.Distribution(L2)
Collection[2] = ot.Distribution(L3)
Collection[3] = ot.Distribution(L4)
```

- Association of the collection and copula to create the composed distribution.

```
Modelproba = ot.ComposedDistribution(Collection, Copula)
```

Propagation using MC simulations

Defining the DOE

- Sampling of input values (here 1000 values)

```
Input = Modelproba.getSample(1000)
```

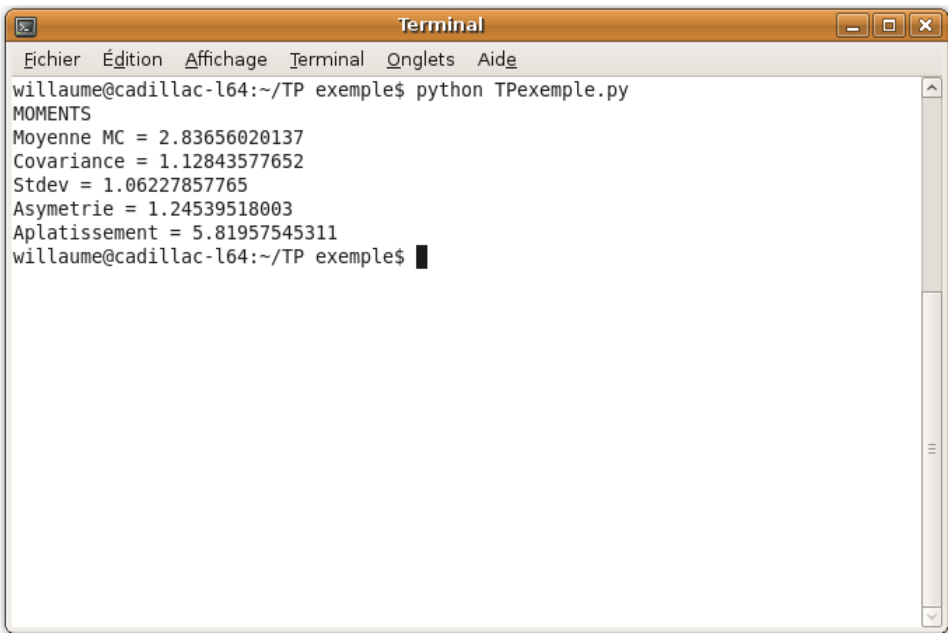
- Evaluation of the output

```
DepMC = depmax(Input)
```

Result of the MC simulation

Evaluation of the 4 first statistical moments

```
Mean = DepMC.computeMean()
Covariance = DepMC.computeCovariance()
stdev= DepMC.computeStandardDeviationPerComponent()
Skewness = DepMC.computeSkewnessPerComponent()
Kurtosis = DepMC.computeKurtosisPerComponent()
```



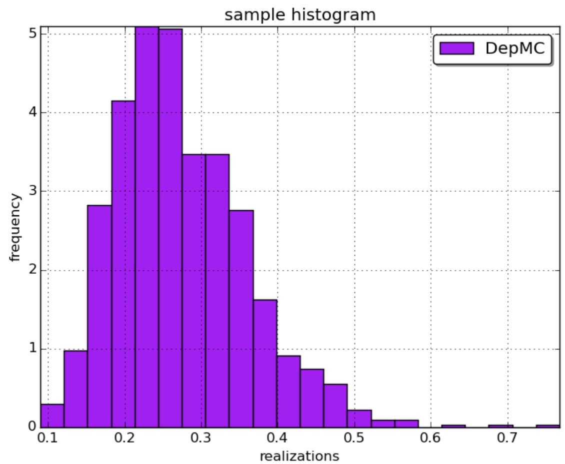
Histogram and empirical CDF

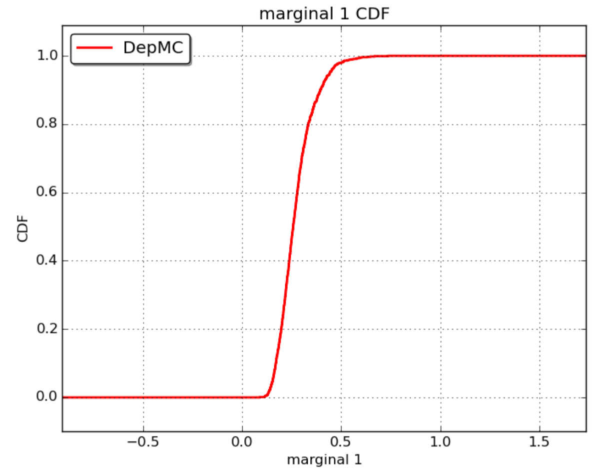
Graphs on the sample DepMC

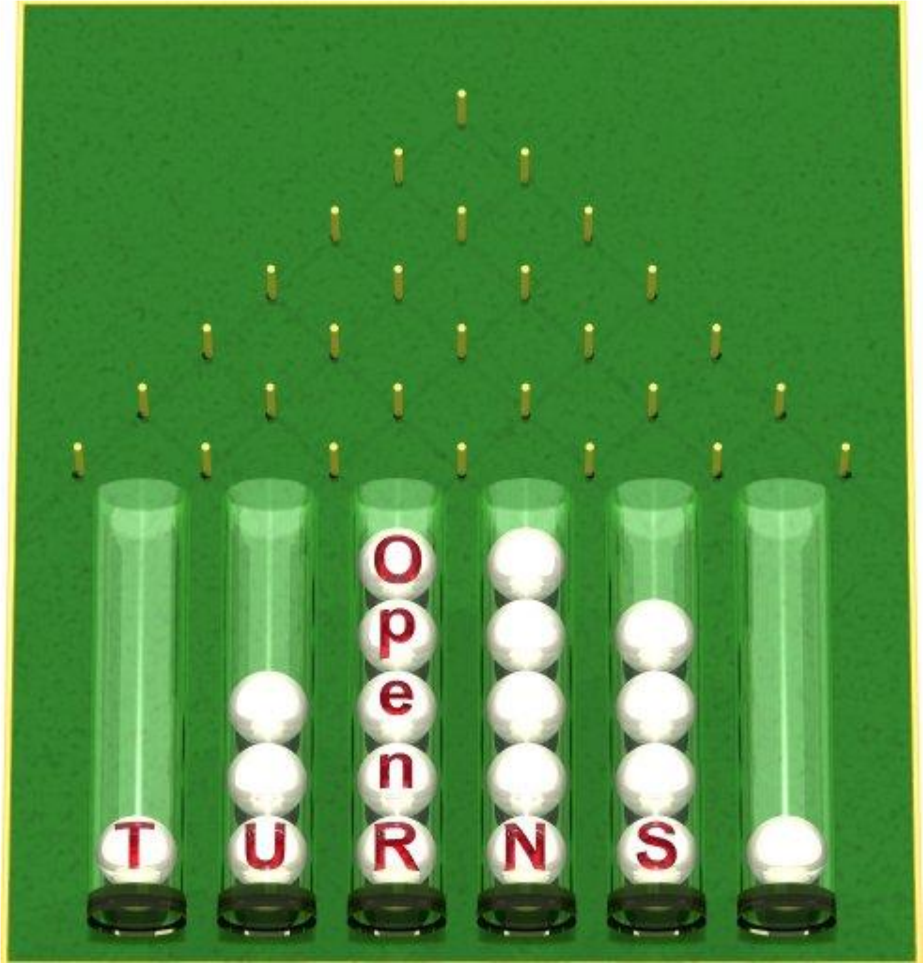
```
from openturns.viewer import View

hist = ot.VisualTest.DrawHistogram(DepMC)
View(hist, bar_kwargs={'label':'DepMC'})

CDF = ot.VisualTest.DrawEmpiricalCDF(DepMC,
                        DepMC.getMin()[0] - 1.0, DepMC.getMax()[0] + 1.0)
View(CDF, step_kwargs={'label':'DepMC'})
```






Open
TURNS