# OTBENCHMARK:

## AN OPEN SOURCE PYTHON PACKAGE FOR BENCHMARKING AND VALIDATING UNCERTAINTY QUANTIFICATION ALGORITHMS

E. Fekhari [1] M. Baudin [1] V. Chabridon [1] Y. Jebroun

[1]EDF R&D - 6 quai Watier, Chatou, France

June 9, 2021

# Introduction

# Industrial context

- In France, the EDF group operates
  - ▷ 56 nuclear power plants
  - ▷ 433 hydraulic power plants
  - ▷ thousands of wind turbines
  - ▷ thousands of photo-voltaic panels
- Industrial risk management through the treatment of uncertainties
- Over the past years, a panel of industrials, public research centers and academics:
  - ▷ jointly proposed a generic methodology for uncertainty management
  - ▷ developed OpenTURNS, a tool for the treatment of uncertainties [1]
- Need for a benchmark platform for uncertainty quantification

# Industrial context

- In France, the EDF group operates
  - 56 nuclear power plants
  - 433 hydraulic power plants
  - thousands of wind turbines
  - thousands of photo-voltaic panels
- Industrial risk management through the treatment of uncertainties
- Over the past years, a panel of industrials, public research centers and academics:
  - jointly proposed a generic methodology for uncertainty management
  - developed OpenTURNS, a tool for the treatment of uncertainties [1]
- Need for a benchmark platform for uncertainty quantification

# Industrial context

- In France, the EDF group operates
  - ▷ 56 nuclear power plants
  - ▷ 433 hydraulic power plants
  - ▷ thousands of wind turbines
  - ▷ thousands of photo-voltaic panels
- Industrial risk management through the treatment of uncertainties
- Over the past years, a panel of industrials, public research centers and academics:
  - ▷ jointly proposed a generic methodology for uncertainty management
  - ▷ developed OpenTURNS, a tool for the treatment of uncertainties [1]
- Need for a benchmark platform for uncertainty quantification

**OpenTURNS**
An Open source initiative for the Treatment of Uncertainties, Risks'N Statistics

# Plan

# Motivations and objectives

# Uncertainty propagation: problem definition

- Uncertain input random vector $\boldsymbol{X} \in \mathbb{R}^d$
- Model $g : \mathbb{R}^d \to \mathbb{R}^p$

$$Y = g(\boldsymbol{X})$$

- Output random variable $Y \in \mathbb{R}^p$

# Uncertainty propagation: problem definition

- Uncertain input random vector $\boldsymbol{X} \in \mathbb{R}^d$
- Model $g : \mathbb{R}^d \to \mathbb{R}^p$

$$\boxed{Y} = \boxed{g}(\boxed{\boldsymbol{X}})$$

- Output random variable $Y \in \mathbb{R}^p$

# Uncertainty propagation: problem definition

- Uncertain input random vector $\boldsymbol{X} \in \mathbb{R}^d$
- Model $g : \mathbb{R}^d \to \mathbb{R}^p$

$$Y = g(\boldsymbol{X})$$

- Output random variable $Y \in \mathbb{R}^p$

# Uncertainty propagation: problem definition

- Uncertain input random vector $\boldsymbol{X} \in \mathbb{R}^d$
- Model $g : \mathbb{R}^d \to \mathbb{R}^p$

$$\boxed{Y} = \boxed{g}(\boxed{\boldsymbol{X}})$$

- Output random variable $Y \in \mathbb{R}^p$

*Reliability analysis formulation:*

- Failure is characterized by a threshold event $E$
- The failure probability, denoted $p_{\mathrm{f}}$, is given by:

$$p_{\mathrm{f}} = \int_{\mathbb{R}^d} \mathbb{1}_E(\mathbf{x}) f_{\boldsymbol{X}}(\mathbf{x}) \mathrm{d}\mathbf{x}$$

- Many standard or advanced algorithms for reliability analysis exist [2]

# Uncertainty propagation: problem definition

- Uncertain input random vector $\boldsymbol{X} \in \mathbb{R}^d$
- Model $g : \mathbb{R}^d \to \mathbb{R}^p$

$$\boxed{Y} = \boxed{g}(\boxed{\boldsymbol{X}})$$

- Output random variable $Y \in \mathbb{R}^p$

*Sensitivity analysis formulation:*

- Compute sensitivity indices reflecting the influence of inputs on the output
- e.g., variance decomposition with the Sobol' indices [3]

$$S_i = \frac{\mathrm{Var}\left[\mathbb{E}[Y|X_i]\right]}{\mathrm{Var}[Y]} \qquad S_{T_i} = \frac{\mathbb{E}\left[\mathrm{Var}[Y|\boldsymbol{X}_{-i}]\right]}{\mathrm{Var}[Y]}$$

  - $S_i$ is the first-order index, $S_{T_i}$ the total-order index of $X_i$
  - $\boldsymbol{X}_{-i}$ stands for $\boldsymbol{X}$ without the $i$-th component

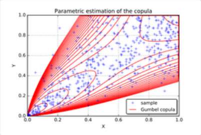
# OpenTURNS: www.openturns.org

- First release: 2007
- Open source, LGPL licensed, C++/Python library
- Project size (2018): 720 classes, more than 6000 services
- Probabilistic programming paradigm for statistical modeling
- Numerical tools dedicated to the treatment of uncertainties
- Generic coupling to any type of physical model

```
user@user-desktop:~$ pip install openturns
```

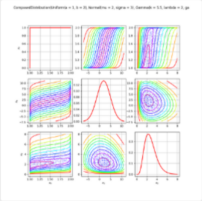
# OpenTURNS: content

## Data analysis



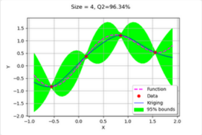

- Manage data and samples
- Sample analysis
- Distribution fitting
- Statistical hypothesis testing
- Estimate dependency and copulas
- Estimate stochastic processes
- Graphics

## Probabilistic modeling



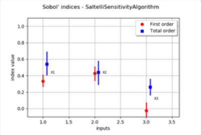
- Distributions
- Copulas
- Stochastic processes

## Meta modeling



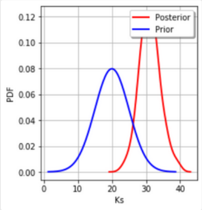

- General purpose metamodels
- Polynomial chaos metamodel
- Kriging metamodel
- Fields metamodels

## Reliability, sensitivity



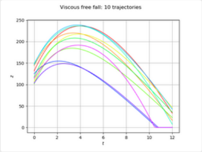

- Central dispersion
- Reliability
- Reliability: processes
- Sensitivity analysis
- Design of experiments

## Calibration




- Least squares and gaussian calibration
- Bayesian calibration

## Functional modeling




- Vectorial functions
- Field functions
- Link to an external code

# otbenchmark: from benchmark repositories to a package

## What is a benchmark?

*"Something that can be measured and used as a standard that other things can be compared with." The Oxford English dictionary*

- Many existing benchmark packages for linear algebra, numerical analysis, optimization
- Some uncertainty quantification repositories of benchmark problems
  - Virtual Library of Simulation Experiments[1]
  - GDR Mascot-Num research group benchmark repository[2]
- "Black-box Reliability Challenge"[3] organized by TNO
  - Online submissions in 2019
  - Almost 30 reliability benchmark problems

[1] http://www.sfu.ca/ ssurjano/index.html
[2] https://www.gdr-mascotnum.fr/benchmarks.html
[3] https://rprepo.readthedocs.io/en/latest/index.html

# `otbenchmark`: from benchmark repositories to a package

## What is a benchmark?

*"Something that can be measured and used as a standard that other things can be compared with." The Oxford English dictionary*

- Many existing benchmark packages for linear algebra, numerical analysis, optimization
- Some uncertainty quantification repositories of benchmark problems
  - Virtual Library of Simulation Experiments[1]
  - GDR Mascot-Num research group benchmark repository[2]
- "Black-box Reliability Challenge"[3] organized by TNO
  - Online submissions in 2019
  - Almost 30 reliability benchmark problems

[1] http://www.sfu.ca/ ssurjano/index.html
[2] https://www.gdr-mascotnum.fr/benchmarks.html
[3] https://rprepo.readthedocs.io/en/latest/index.html

# otbenchmark: from benchmark repositories to a package

## What is a benchmark?

*"Something that can be measured and used as a standard that other things can be compared with." The Oxford English dictionary*

- Many existing benchmark packages for linear algebra, numerical analysis, optimization
- Some uncertainty quantification repositories of benchmark problems
  - ▷ Virtual Library of Simulation Experiments[1]
  - ▷ GDR Mascot-Num research group benchmark repository[2]
- "Black-box Reliability Challenge"[3] organized by TNO
  - ▷ Online submissions in 2019
  - ▷ Almost 30 reliability benchmark problems

[1] http://www.sfu.ca/ ssurjano/index.html

[2] https://www.gdr-mascotnum.fr/benchmarks.html

[3] https://rprepo.readthedocs.io/en/latest/index.html

# otbenchmark: from benchmark repositories to a package

## What is a benchmark?

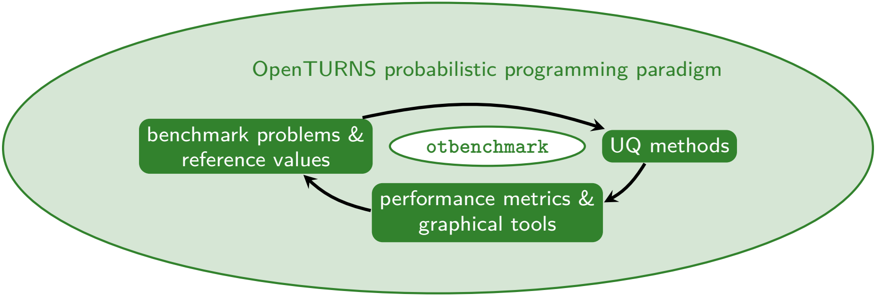
*"Something that can be measured and used as a standard that other things can be compared with." The Oxford English dictionary*

- Many existing benchmark packages for linear algebra, numerical analysis, optimization
- Some uncertainty quantification repositories of benchmark problems
  - Virtual Library of Simulation Experiments[1]
  - GDR Mascot-Num research group benchmark repository[2]
- "Black-box Reliability Challenge"[3] organized by TNO
  - Online submissions in 2019
  - Almost 30 reliability benchmark problems

[1] http://www.sfu.ca/ ssurjano/index.html
[2] https://www.gdr-mascotnum.fr/benchmarks.html
[3] https://rprepo.readthedocs.io/en/latest/index.html

# otbenchmark: benchmark platform for UQ

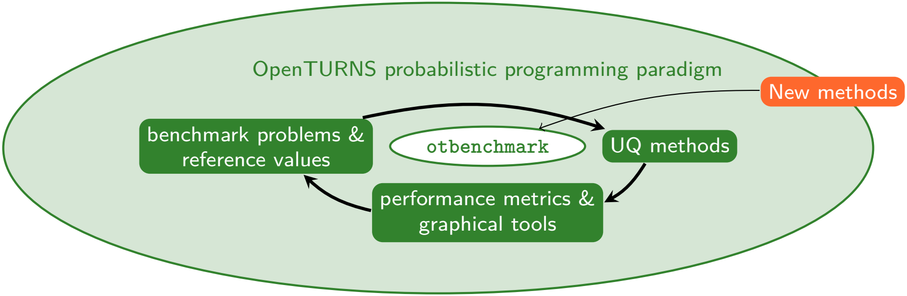
- 26 reliability problems and 4 sensitivity analysis problems so far
- Reference values either computed by exact quadrature methods or large Monte Carlo sampling




- *Scenario #1:* test a new UQ algorithm on a panel of problems and analyze its performances
- *Scenario #2:* apply and compare several UQ algorithms available on a given benchmark problem

# otbenchmark: benchmark platform for UQ

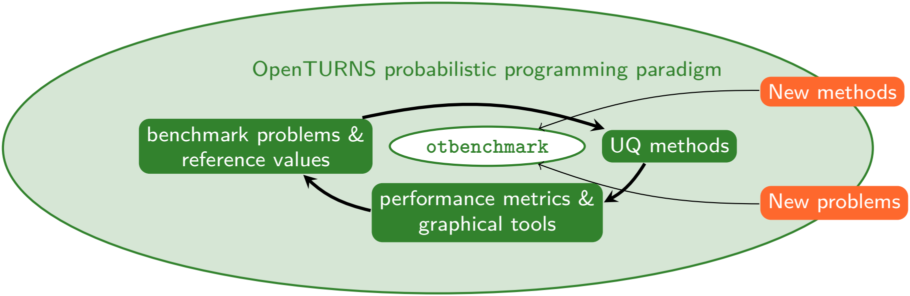
- 26 reliability problems and 4 sensitivity analysis problems so far
- Reference values either computed by exact quadrature methods or large Monte Carlo sampling




- *Scenario #1:* test a new UQ algorithm on a panel of problems and analyze its performances
- *Scenario #2:* apply and compare several UQ algorithms available on a given benchmark problem

# otbenchmark: benchmark platform for UQ

- 26 reliability problems and 4 sensitivity analysis problems so far
- Reference values either computed by exact quadrature methods or large Monte Carlo sampling




- *Scenario #1:* test a new UQ algorithm on a panel of problems and analyze its performances
- *Scenario #2:* apply and compare several UQ algorithms available on a given benchmark problem

# Package architecture

# Package architecture

- Generic classes `ReliabilityBenchmarkProblem` and `SensitivityBenchmarkProblem`
- Problem-specific classes inheriting methods from the generic classes
- `ReliabilityBenchmarkResult` to manage the result of a benchmark
- `ReliabilityBenchmarkProblemList` to perform a benchmark on list of problems
- `DrawEvent` to provide high-dimension graphical analysis tools (e.g., cross-cuts)

# Package architecture

- Generic classes `ReliabilityBenchmarkProblem` and `SensitivityBenchmarkProblem`
- Problem-specific classes inheriting methods from the generic classes
- `ReliabilityBenchmarkResult` to manage the result of a benchmark
- `ReliabilityBenchmarkProblemList` to perform a benchmark on list of problems
- `DrawEvent` to provide high-dimension graphical analysis tools (e.g., cross-cuts)

# Package architecture

- Generic classes `ReliabilityBenchmarkProblem` and `SensitivityBenchmarkProblem`
- Problem-specific classes inheriting methods from the generic classes
- `ReliabilityBenchmarkResult` to manage the result of a benchmark
- `ReliabilityBenchmarkProblemList` to perform a benchmark on list of problems
- `DrawEvent` to provide high-dimension graphical analysis tools (e.g., cross-cuts)

# Package architecture

- Generic classes `ReliabilityBenchmarkProblem` and `SensitivityBenchmarkProblem`
- Problem-specific classes inheriting methods from the generic classes
- `ReliabilityBenchmarkResult` to manage the result of a benchmark
- `ReliabilityBenchmarkProblemList` to perform a benchmark on list of problems
- `DrawEvent` to provide high-dimension graphical analysis tools (e.g., cross-cuts)

# Package architecture

- Generic classes `ReliabilityBenchmarkProblem` and `SensitivityBenchmarkProblem`
- Problem-specific classes inheriting methods from the generic classes
- `ReliabilityBenchmarkResult` to manage the result of a benchmark
- `ReliabilityBenchmarkProblemList` to perform a benchmark on list of problems
- `DrawEvent` to provide high-dimension graphical analysis tools (e.g., cross-cuts)

# Reliability example

# RP 57: problem definition

Considering the threshold event:

$$E = \{g(\boldsymbol{X}) < 0, \quad \boldsymbol{X} \in \mathbb{R}^2\}$$

with the system limit-state function:

$$\boxed{g(\boldsymbol{X}) = \min\left(\max(g_1(\boldsymbol{X}), g_2(\boldsymbol{X})), g_3(\boldsymbol{X})\right)}$$

$$\begin{cases} g_1(\boldsymbol{X}) = -X_1^2 + X_2^3 + 3 \\ g_2(\boldsymbol{X}) = 2 - X_1 - 8X_2 \\ g_3(\boldsymbol{X}) = (X_1 + 3)^2 + (X_2 + 3)^2 - 4 \end{cases}$$

Where $\boldsymbol{X} \in \mathbb{R}^2$ has independent Gaussian marginals:

$$X_1 \sim \mathcal{N}(\mu_1 = 0, \sigma_1 = 1), \quad X_2 \sim \mathcal{N}(\mu_2 = 0, \sigma_2 = 1).$$

For these parameters, the reference probability is:

$$\boxed{p_{\mathrm{f,ref}} = 0.0284}$$

# RP 57: problem definition

Considering the threshold event:

$$E = \{g(\boldsymbol{X}) < 0, \quad \boldsymbol{X} \in \mathbb{R}^2\}$$

with the system limit-state function:

$$\boxed{g(\boldsymbol{X}) = \min\left(\max(g_1(\boldsymbol{X}), g_2(\boldsymbol{X})), g_3(\boldsymbol{X})\right)}$$

$$\begin{cases} g_1(\boldsymbol{X}) = -X_1^2 + X_2^3 + 3 \\ g_2(\boldsymbol{X}) = 2 - X_1 - 8X_2 \\ g_3(\boldsymbol{X}) = (X_1 + 3)^2 + (X_2 + 3)^2 - 4 \end{cases}$$

Where $\boldsymbol{X} \in \mathbb{R}^2$ has independent Gaussian marginals:

$$X_1 \sim \mathcal{N}(\mu_1 = 0, \sigma_1 = 1), \quad X_2 \sim \mathcal{N}(\mu_2 = 0, \sigma_2 = 1).$$

For these parameters, the reference probability is:

$$\boxed{p_{\mathrm{f,ref}} = 0.0284}$$

# RP57: `otbenchmark` demonstration

*Reliability problem definition*

```python
# Import packages
import openturns as ot
import otbenchmark as otb
problem = otb.ReliabilityProblem57()
event = problem.getEvent()
g = event.getFunction()
print("Reference failure probability = %.4f" % (problem.getProbability()))
```

```
Reference failure probability = 0.0284
```

*FORM approximation*

```python
meta_algo = otb.ReliabilityBenchmarkMetaAlgorithm(problem)
bench_results = meta_algo.runFORM(ot.AbdoRackwitz())
pf = bench_results.computedProbability
nb_digits = bench_results.numberOfCorrectDigits
nb_simu = bench_results.numberOfFunctionEvaluations
print("FORM failure probability = %.4f" % (pf))
print("FORM nb. good digits = %.4f" % (nb_digits))
print("FORM nb. function calls = %d" % (nb_simu))
```

```
FORM failure probability = 0.4504
FORM nb. correct digits = -1.1719
FORM nb. function calls = 1004
```

FORM performs poorly on a system limit-state function (as foreseen here)

# RP57: otbenchmark demonstration

## *Reliability problem definition*

```python
# Import packages
import openturns as ot
import otbenchmark as otb
problem = otb.ReliabilityProblem57()
event = problem.getEvent()
g = event.getFunction()
print("Reference failure probability = %.4f" % (problem.getProbability()))
```

```
Reference failure probability = 0.0284
```

## *FORM approximation*

```python
meta_algo = otb.ReliabilityBenchmarkMetaAlgorithm(problem)
bench_results = meta_algo.runFORM(ot.AbdoRackwitz())
pf = bench_results.computedProbability
nb_digits = bench_results.numberOfCorrectDigits
nb_simu = bench_results.numberOfFunctionEvaluations
print("FORM failure probability = %.4f" % (pf))
print("FORM nb. good digits = %.4f" % (nb_digits))
print("FORM nb. function calls = %d" % (nb_simu))
```

```
FORM failure probability = 0.4504
FORM nb. correct digits = -1.1719
FORM nb. function calls = 1004
```

FORM performs poorly on a system limit-state function (as foreseen here)

# RP57: `otbenchmark` demonstration

## *Reliability problem definition*

```python
# Import packages
import openturns as ot
import otbenchmark as otb
problem = otb.ReliabilityProblem57()
event = problem.getEvent()
g = event.getFunction()
print("Reference failure probability = %.4f" % (problem.getProbability()))
```

```
Reference failure probability = 0.0284
```

## *FORM approximation*

```python
meta_algo = otb.ReliabilityBenchmarkMetaAlgorithm(problem)
bench_results = meta_algo.runFORM(ot.AbdoRackwitz())
pf = bench_results.computedProbability
nb_digits = bench_results.numberOfCorrectDigits
nb_simu = bench_results.numberOfFunctionEvaluations
print("FORM failure probability = %.4f" % (pf))
print("FORM nb. good digits = %.4f" % (nb_digits))
print("FORM nb. function calls = %d" % (nb_simu))
```

```
FORM failure probability = 0.4504
FORM nb. correct digits = -1.1719
FORM nb. function calls = 1004
```

FORM performs poorly on a system limit-state function (as foreseen here)

# RP57: `otbenchmark` demonstration

*Monte Carlo simulation*

```
sample_size = 1000
bench_results = meta_algo.runMonteCarlo(sample_size)
pf = bench_results.computedProbability
nb_digits = bench_results.numberOfCorrectDigits
nb_simu = bench_results.numberOfFunctionEvaluations
print("MC failure probability = %.4f" % (pf))
print("MC nb. correct digits = %.4f" % (nb_digits))
print("MC nb. function calls = %d" % (nb_simu))
```

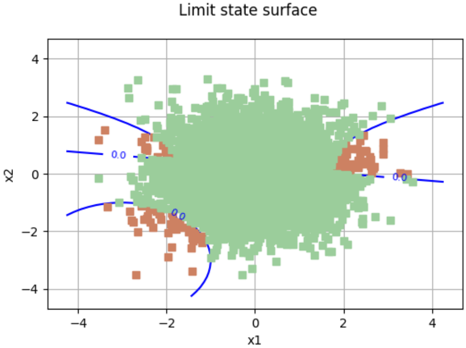
```
MC failure probability = 0.0260
MC nb. good digits = 1.0731
MC nb. function calls = 1000
```

Monte Carlo performs well since $p_{\mathrm{f}}$ is not too rare

# RP57: `otbenchmark` demonstration

## Graphical analysis through cross-cuts

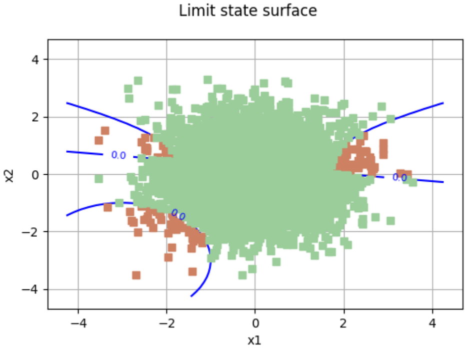
```
sample_size = 2000
drawEvent = otb.DrawEvent(event)
cloud = drawEvent.drawSampleCrossCut(sample_size)
bounds = ot.Interval([-4, -4], [4, 4])
graph = drawEvent.drawLimitStateCrossCut(bounds)
graph.add(cloud)
graph
```



# RP57: otbenchmark demonstration

## Graphical analysis through cross-cuts

```
sample_size = 2000
drawEvent = otb.DrawEvent(event)
cloud = drawEvent.drawSampleCrossCut(sample_size)
bounds = ot.Interval([-4, -4], [4, 4])
graph = drawEvent.drawLimitStateCrossCut(bounds)
graph.add(cloud)
graph
```




otbenchmark can also cross algorithms and problems and compare performances

# Benchmark results

# Benchmark results: crossing problems and algorithms

Table: Estimation of $p_{\mathrm{f}}$ for all the problems using 5 algorithms ($n_{\max} = 10^4$ calls).

| | $p_{\mathrm{f,ref}}$ | Monte Carlo | FORM | SORM | FORM-IS | Subset |
|---|---|---|---|---|---|---|
| RP8 | 7.840e-04 | 9.000e-04 | 6.599e-04 | 7.837e-04 | 7.737e-04 | 8.863e-04 |
| RP14 | 7.520e-03 | 9.000e-04 | 7.003e-04 | 6.988e-04 | 7.598e-04 | 8.720e-04 |
| RP22 | 4.160e-03 | 3.500e-03 | 6.210e-03 | 4.391e-03 | 4.259e-03 | 4.117e-03 |
| RP24 | 2.860e-03 | 3.600e-03 | 6.209e-03 | 6.209e-03 | 2.749e-03 | 2.486e-03 |
| RP25 | 6.140e-06 | 1.000e-04 | 2.105e-03 | 1.064e-05 | 4.644e-05 | 3.415e-05 |
| RP28 | 1.460e-07 | – | 2.850e-08 | 0.000e+00 | 1.332e-07 | 1.756e-07 |
| RP31 | 1.800e-04 | 2.300e-03 | 2.275e-02 | 2.275e-02 | 3.319e-03 | 3.919e-03 |
| RP33 | 2.570e-03 | 1.600e-03 | 1.350e-03 | 1.350e-03 | 2.322e-03 | 2.718e-03 |
| RP35 | 3.540e-03 | 3.000e-03 | 1.350e-03 | 2.134e-03 | 2.377e-03 | 3.430e-03 |
| RP38 | 8.100e-03 | 8.500e-03 | 7.902e-03 | 8.029e-03 | 8.146e-03 | 7.848e-03 |
| RP53 | 3.130e-02 | 3.260e-02 | 1.180e-01 | 2.986e-02 | 3.143e-02 | 2.971e-02 |
| RP55 | 5.600e-01 | 5.660e-01 | 5.000e-01 | 1.093e-05 | 5.645e-01 | 5.655e-01 |
| RP54 | 9.980e-04 | 1.100e-03 | 5.553e-02 | 3.552e-03 | 9.767e-04 | 9.611e-04 |
| RP57 | 2.840e-02 | 2.950e-02 | 4.504e-01 | 0.000e+00 | 2.746e-02 | 2.772e-02 |
| RP75 | 1.070e-02 | 1.030e-02 | 0.000e+00 | 0.000e+00 | 0.000e+00 | 9.409e-03 |
| RP89 | 5.430e-03 | 5.000e-03 | 2.009e-09 | 2.009e-09 | 9.002e-05 | 5.460e-03 |
| RP107 | 2.920e-07 | – | 2.867e-07 | 2.867e-07 | 2.896e-07 | 2.337e-07 |
| RP110 | 3.190e-05 | – | 3.167e-05 | 3.167e-05 | 3.078e-05 | 7.116e-06 |
| RP111 | 7.650e-07 | – | 0.000e+00 | 0.000e+00 | 0.000e+00 | 7.308e-07 |
| RP63 | 3.790e-04 | 1.000e-04 | 1.000e+00 | 0.000e+00 | 0.000e+00 | 4.063e-04 |
| RP91 | 6.970e-04 | 1.000e-03 | 6.984e-04 | 7.001e-04 | 6.964e-04 | 6.838e-04 |
| RP60 | 4.560e-02 | 4.860e-02 | 4.484e-02 | 4.484e-02 | 4.503e-02 | 4.230e-02 |
| RP77 | 2.870e-07 | – | 6.687e-02 | 6.687e-02 | 4.002e-07 | 3.683e-07 |
| Four-branch serial system | 2.186e-03 | 2.900e-03 | 0.000e+00 | 0.000e+00 | 0.000e+00 | 2.428e-03 |
| R-S | 7.865e-02 | 7.870e-02 | 7.865e-02 | 7.865e-02 | 7.792e-02 | 7.633e-02 |
| Axial stressed beam | 2.920e-02 | 2.690e-02 | 2.998e-02 | 2.933e-02 | 2.867e-02 | 2.936e-02 |

# Benchmark results: performance metrics

Figure: Log relative error using 5 algorithms ($n_{\max} = 10^4$ calls).

| | Monte Carlo | FORM | SORM | FORM-IS | Subset |
|---|---|---|---|---|---|
| RP8 | 0.83 | 0.80 | 3.43 | 1.88 | 0.88 |
| RP14 | 0.06 | 0.04 | 0.04 | 0.05 | 0.05 |
| RP22 | 0.80 | 0.31 | 1.26 | 1.62 | 1.99 |
| RP24 | 0.59 | 0.07 | 0.07 | 1.41 | 0.88 |
| RP25 | 1.18 | 2.53 | 0.14 | 0.82 | 0.66 |
| RP28 | 0.00 | 0.09 | 0.00 | 1.06 | 0.69 |
| RP31 | 1.07 | 2.10 | 2.10 | 1.24 | 1.32 |
| RP33 | 0.42 | 0.32 | 0.32 | 1.02 | 1.24 |
| RP35 | 0.82 | 0.21 | 0.40 | 0.48 | 1.51 |
| RP38 | 1.31 | 1.61 | 2.06 | 2.25 | 1.51 |
| RP53 | 1.38 | 0.44 | 1.34 | 2.38 | 1.29 |
| RP55 | 1.97 | 0.97 | 0.00 | 2.10 | 2.01 |
| RP54 | 0.99 | 1.74 | 0.41 | 1.67 | 1.43 |
| RP57 | 1.41 | 1.17 | 0.00 | 1.48 | 1.62 |
| RP75 | 1.43 | 0.00 | 0.00 | 0.00 | 0.92 |
| RP89 | 1.10 | 0.00 | 0.00 | 0.01 | 2.26 |
| RP107 | 0.00 | 1.74 | 1.74 | 2.08 | 0.70 |
| RP110 | 0.00 | 2.14 | 2.14 | 1.45 | 0.11 |
| RP111 | 0.00 | 0.00 | 0.00 | 0.00 | 1.35 |
| RP63 | 0.13 | 3.42 | 0.00 | 0.00 | 1.14 |
| RP91 | 0.36 | 2.70 | 2.35 | 3.04 | 1.72 |
| RP60 | 1.18 | 1.78 | 1.78 | 1.91 | 1.14 |
| RP77 | 0.00 | 5.37 | 5.37 | 0.40 | 0.55 |
| Four-branch serial system | 0.49 | 0.00 | 0.00 | 0.00 | 0.96 |

*Performance metrics*

For a reference value $p_{\mathrm{f,ref}}$ and its approximation $p_{\mathrm{f,apx}}$

- Absolute error: $\epsilon = |p_{\mathrm{f,ref}} - p_{\mathrm{f,apx}}|$
- Relative error: $\eta = \dfrac{|p_{\mathrm{f,ref}} - p_{\mathrm{f,apx}}|}{|p_{\mathrm{f,ref}}|}$
- Log relative error: $N = log_{10}\left(\dfrac{1}{\eta}\right)$

# Conclusion

# Conclusion: `otbenchmark`, a benchmark platform for UQ

- `otbenchmark`'s ambition: gather and standardize UQ problems in a single Python package
- Any user can bring, either his new UQ algorithm or new use-case (with reference values!)
- Wide variety of reliability and sensitivity analysis problems
- Performance metrics and graphical tools for a smart analysis
- Agile development workflow to ensure a robust benchmark
- More problems to come, and more fields of uncertainty quantification to analyze (e.g., central tendency, calibration)

# Welcome to participate!

- GitHub public repository[1]
- Unit-tests and continuous integration using Circle-CI
- `otbenchmark` 0.1.1 package available on PyPI

GitHub + circleci → python Package Index

[1] Official repository: `https://github.com/mbaudin47/otbenchmark`

# Welcome to participate!

- GitHub public repository[1]
- Unit-tests and continuous integration using Circle-CI
- `otbenchmark` 0.1.1 package available on PyPI

GitHub + circleci → python Package Index

```
user@user-desktop:~$ pip install otbenchmark
```

[1] Official repository: `https://github.com/mbaudin47/otbenchmark`

[1] M. Baudin, A. Dutfoy, B. Iooss, and A.-L. Popelin. OpenTURNS: An Industrial Software for Uncertainty Quantification in Simulation. In R. Ghanem, D. Higdon, and H. Owhadi, editors, *Handbook of Uncertainty Quantification*, pages 2001–2038. Springer International Publishing, Cham, 2017.

[2] Vincent Chabridon. *Reliability-oriented sensitivity analysis under probabilistic model uncertainty - Application to aerospace systems*. PhD thesis, 04 2019.

[3] B. Iooss and P. Lemaître. A Review on Global Sensitivity Analysis Methods. In G. Dellino and C. Meloni, editors, *Uncertainty Management in Simulation-Optimization of Complex Systems: Algorithms and Applications*, chapter 5, pages 101–122. Springer US, Boston, MA, 2015.

Thank you