OpenTURNS release highlights

J. Schueller (Phimeca)

User Day #18, June 13th 2025, EDF Lab

Overview

New features since last year in releases:

- v1.24: fall 2024
- v1.25: spring 2025

Contents

Warning

Not about

- New conditional modelling features
- New quantile estimation features
- New GP API

Line sampling simulation 1/5

Description

- Contract EDF with E. Ardillon, A. Ajenjo, J. Mure with Phimeca
- Simulation algorithm comparable to directional sampling
- Implements the original algorithm from [a] + adaptive important direction variant from [b]
- The line sampling algorithm estimates the probability of the event $\mathcal{D}_f$:

$$P_f = \mathbb{P}\left(g\left(\boldsymbol{X}\right) \leq 0\right) = \int_{\mathbb{R}^d} 1_{\{g(\boldsymbol{x}) \leq 0\}} \mu_{\boldsymbol{X}}\left(\boldsymbol{x}\right) \mathrm{d}\boldsymbol{x}$$

---

[a] Koutsourelakis, H. Pradlwarter, G. Schueller, *Reliability of structures in high dimensions, part i: algorithms and applications*, Probabilistic Engineering Mechanics 19 (4) (2004) 409–417

[b] Angelis M., Patelli E., Beer M., *Advanced line sampling for efficient robust reliability analysis*, Structural safety, 52 :170-182, 2015.

Algorithm

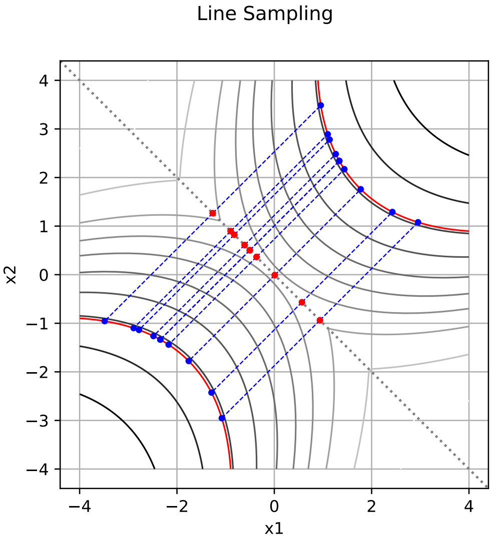
The generic line sampling algorithm follows the steps for $k = 1, \dots n$:

- Draw a sample $\boldsymbol{z_k} \sim Z$ and project it on the hyperplane normal to $\boldsymbol{\alpha}$ to obtain $P_{\boldsymbol{\alpha}}^{\perp}(\boldsymbol{z_k})$.
- Find the roots of $r \mapsto g \circ T^{-1}(r\boldsymbol{\alpha} + P_{\boldsymbol{\alpha}}^{\perp}(\boldsymbol{z_k}))$.
- Use the roots to compute $p_{\boldsymbol{z_k}} = \mathbb{P}\left(R \in I_{P_{\boldsymbol{\alpha}}^{\perp}(\boldsymbol{z_k})}\right)$.

The global probability P_f is computed from all the $p_{\boldsymbol{z_k}}$ probabilities:

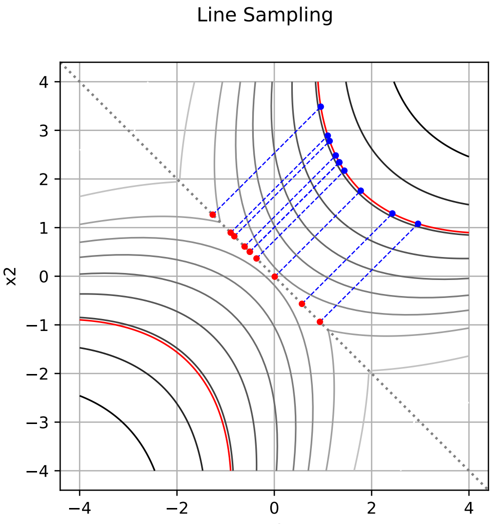
$$\widehat{P}_{f,LS} = \frac{1}{n}\sum_{i=1}^{n} p_{\boldsymbol{z_k}}$$

Line sampling simulation 3/5



Line sampling simulation 4/5

Only search in the direction of α:



Line sampling simulation 5/5

```python
# 1. preliminary FORM
optim = ot.Cobyla()
optim.setStartingPoint(X.getMean())
algo = ot.FORM(optim, event)
algo.run()
result_form = algo.getResult()
alpha = result_form.getStandardSpaceDesignPoint()  # from u*

# 2. LS
alpha = [0.0, 0.0, 0.0, 0.0, 1.0]
rootStrategy = ot.SafeAndSlow(ot.Brent(1e-3, 1e-3, 1e-3, 5), 8, 0.01)
algo = otexp.LineSampling(event, alpha, rootStrategy)
algo.setMaximumOuterSampling(2000)
algo.setMaximumCoefficientOfVariation(5e-2)
algo.setSearchOppositeDirection(False)  # search 1 direction instead of both
algo.setAdaptiveImportantDirection(True)  # adaptive alpha
algo.run()
result = algo.getResult()
pf = result.getProbabilityEstimate()
```

Description

Contract Airbus with Phimeca, based on Crombecq's LOLA-Voronoi design[a].

- Exploration criterion based on Voronoi cell volume:

$$v(\boldsymbol{p}_i) = \int_{\boldsymbol{x} \in \mathcal{V}_i} \mu_{\boldsymbol{X}}(\boldsymbol{x}) d\boldsymbol{x} = \mathbb{E}\left[1_{\mathcal{V}_i}(\boldsymbol{X})\right]$$

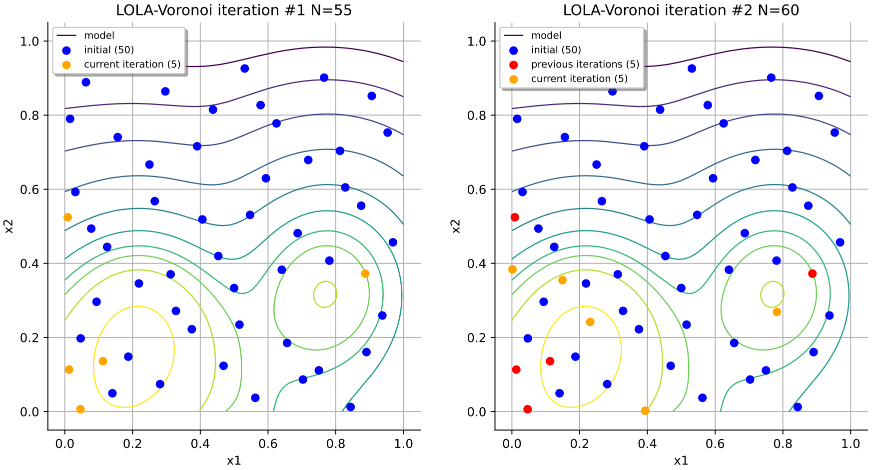
- Exploitation criterion based on a measure of the nonlinearity of the model:

$$e_k(\boldsymbol{p}_r) = \sum_{i=1}^{m} \left|g_k(\boldsymbol{p}_{ri}) - (g_k(\boldsymbol{p}_r) + \boldsymbol{J}_i(\boldsymbol{p}_{ri} - \boldsymbol{p}_r))\right|$$

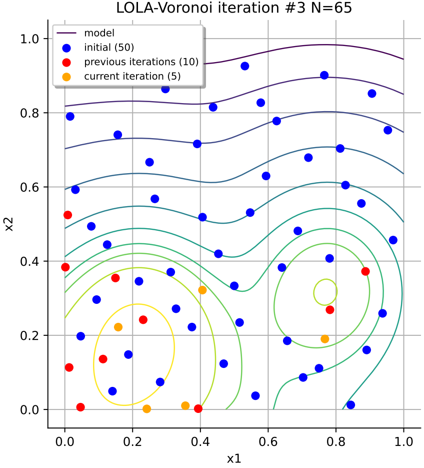
- Combined into hybrid score:

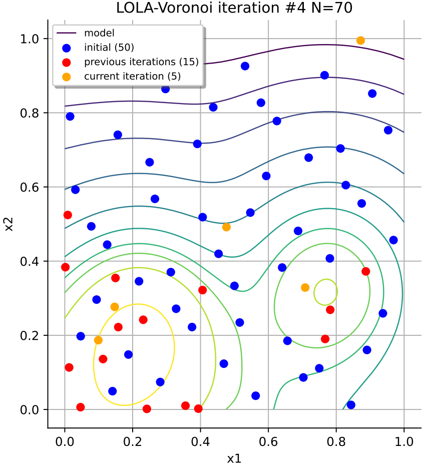
$$h(\boldsymbol{p}_i) = \lambda v(\boldsymbol{p}_i) + (1 - \lambda) \frac{e(\boldsymbol{p}_i)}{\sum_{j=1}^{n} e(\boldsymbol{p}_j)} \forall i \in [1, n]$$

LOLA-Voronoi sequential design 2/7

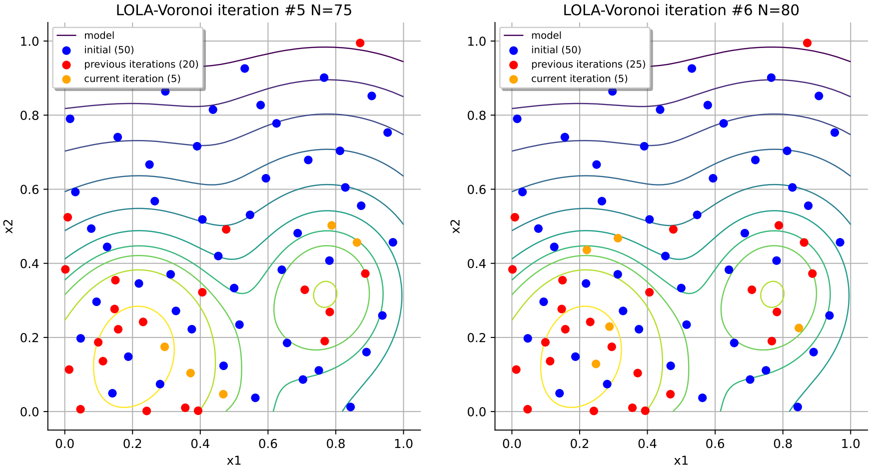


LOLA-Voronoi sequential design 3/7

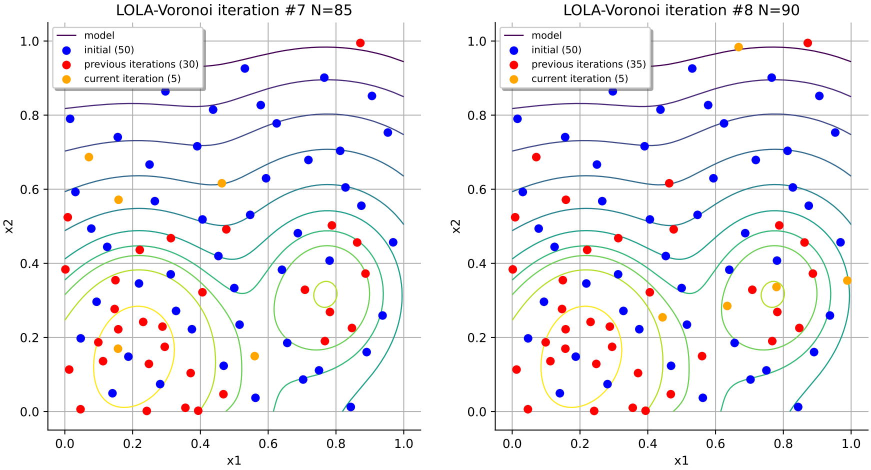




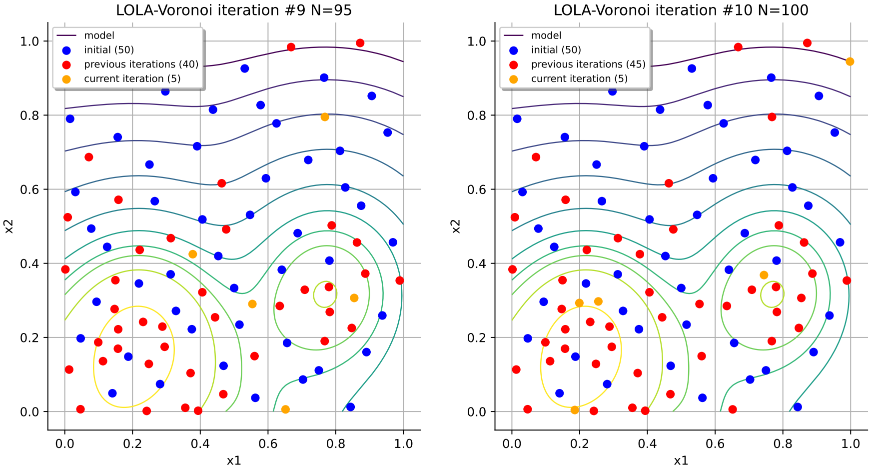
LOLA-Voronoi sequential design 4/7



LOLA-Voronoi sequential design 5/7



LOLA-Voronoi sequential design 6/7



LOLA-Voronoi sequential design 7/7

```python
# 1. initial design
distribution = ot.JointDistribution([ot.Uniform(-1.0, 1.0)] * 2)
N = 50
design = ot.LowDiscrepancyExperiment(ot.HaltonSequence(), distribution, N)
x0 = design.generate()
y0 = f1(x0)

# 2. sequential experiment
algo = otexp.LOLAVoronoi(x0, y0, distribution)
for i in range(10):
    x = algo.generate(5)       # generate 5 new input samples
    y = f(x)                   # evaluate output samples
    algo.update(x, y)          # update state with new x/y pairs

# 3. learn metamodel on all samples
x_final = algo.getInputSample()
y_final = algo.getOutputSample()
algo_mm = ot.FunctionalChaosAlgorithm(x_final, y_final, distribution)
algo_mm.run()
metamodel = algo_mm.getResult().getMetaModel()
```

LeastSquaresEquationsSolver

Least squares solver for the resolution of a system of non-linear equations by numerical optimization.

```
import openturns as ot
import openturns.experimental as otexp

inputs = ['x', 'y']
formulas = ['y*x-sin(2*x)', '1 + cos(y) + x']
analytical = ot.SymbolicFunction(inputs, formulas)

algo = otexp.LeastSquaresEquationsSolver()
algo.setResidualError(1e-8)
starting_point = [2.0, 1.0]
solution = algo.solve(analytical, starting_point)
```

RatioOfUniforms 1/3

Description

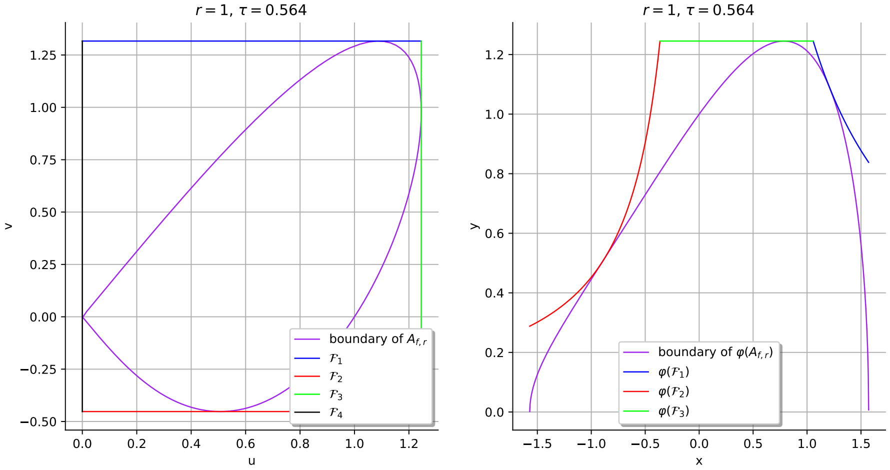
Allows to generate samples from a PDF $p(x) = cf(x) \forall \boldsymbol{x} \in \mathbb{R}^d$ by rejection sampling.

$$A_{f,r} = \left\{ (u, v_1, \dots, v_d) \in \mathbb{R}^{d+1} \,|\, 0 \leq u \leq f\left(\frac{v_1}{u^r}, \dots, \frac{v_d}{u^r}\right)^{\frac{1}{1+rd}} \right\}$$

Let $(U, V_1, \dots, V_d)$ be a random variable uniformly distributed on $A_{f,r}$.
Then $\left(\frac{V_1}{U^r}, \dots, \frac{V_d}{U^r}\right)$ is a random vector distributed according to p.

- Used internally to sample our conditional (continous) distributions in moderate dimension.
- Exposed as API for custom distributions

$$\log f(x) = \log(\cos(x)) + x, x \in]-\pi/2, \pi/2[$$



RatioOfUniforms 3/3

```
import openturns as ot
import openturns.experimental as otexp
from math import pi

f = ot.SymbolicFunction('x', '(1.5+sin(x))*exp(x)')
log_UnscaledPDF = ot.ComposedFunction(ot.SymbolicFunction('x', 'log(x)'), f)
range_PDF = ot.Interval(0.0, 2.0 * pi)
ratioOfU = otexp.RatioOfUniforms(log_UnscaledPDF, range_PDF, False)
collMultiStart = ratioOfU.initialize()
x = ratioOfU.getRealization()
sample = ratioOfU.getSample(10)
ratioOfU = otexp.RatioOfUniforms(ot.Student(8.5, 3))
```

Documentation improvements

- Improved functions API with graphs
- Improved documentation of Directional sampling classes
- Consistency of examples
- SVG graphics

Other improvements

- Smolyak nested tensorization
- Advanced validation of Distribution classes
- Allow MultiStart optimization in FORM algorithms

Packaging 1/2

Python channels

- Pip / Conda
- Versions: 3.9+
- OS: Windows, Linux, MacOS
- Architectures: x86_64, arm64 (Linux+MacOS)

Packaging 2/2

Supported Linux distributions

- Ubuntu 22/24/25
- Debian 11/12
- Fedora 41/42
- OpenSUSE 15.6
- Mageia 9
- ArchLinux

... and FreeBSD

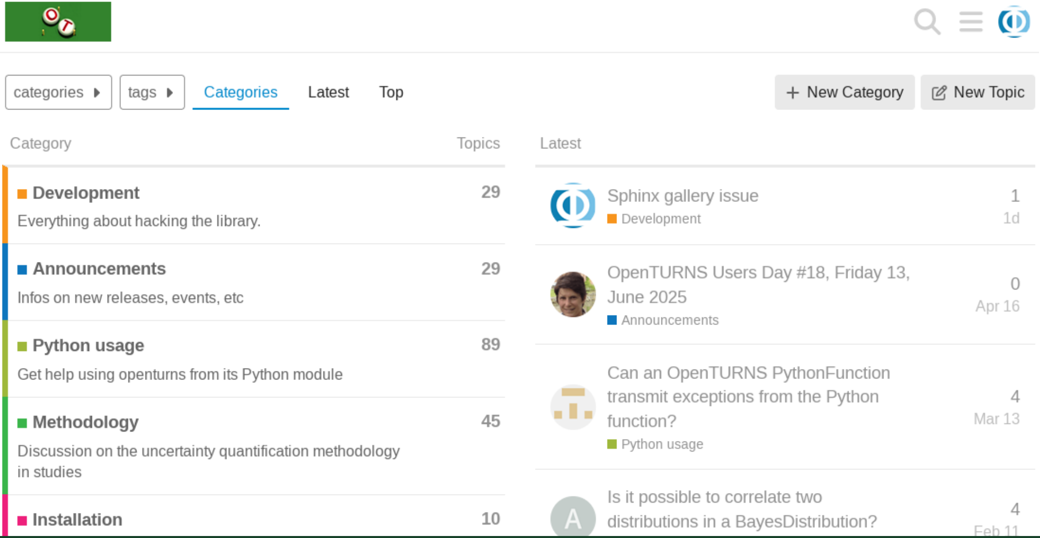
Outlook

2025 work

- Work on functional chaos expansion API
- Dimension reduction models
- Interfacing with SMT2 surrogate model toolbox
- Classification with random forests
- Calibration with bounds, fonctional models, ABC bayesian method
- LMG sensitivity indices (dependent variables for linear models)
- VonMises-Fisher distribution in high dimension, tensorized covariance models
- Performance of conditioned GP on a large grid

Community

Join our forum: https://openturns.discourse.group



END

Thank you for your attention!
Any questions?