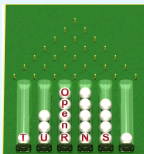


# OpenTURNS Developer training: first steps

Trainer : Régis LEBRUN  
Airbus  
regis.lebrun@airbus.com

Developers training



# OpenTURNS: first steps

- 1 Navigation in the source code
- 2 Library development
- 3 Module development

# Navigation in the source code

## The Uniform distribution

- Locate the class within the library source code;
- Follow its inheritance graph in order to explore the Bridge pattern;
- Locate the associated regression test;
- Execute the test;
- Locate its SWIG interface file and its associated Python module;
- Execute the associated python test.

# Library development 1/5

## Projects

- (\*) **InverseDistanceWeightingInterpolation** as a specialization of **EvaluationImplementation** (see `lib/src/Base/Func`). Given a set of data  $(x_i, y_i)_{i=1, \dots, N}$  in  $\mathbb{R}^n \times \mathbb{R}^p$ , the IDW interpolation is defined by:

$$\forall x \in \mathbb{R}^n, u(x) = \begin{cases} \frac{\sum_{i=1}^N w_i(x) y_i}{\sum_{i=1}^N w_i(x)} & \text{if } \forall i, d(x, x_i) > 0 \\ y_i & \text{if } \exists i, d(x, x_i) = 0 \end{cases} \quad (1)$$

where  $w_i(x) = \frac{1}{d(x, x_i)^p}$  and  $p > 0$  a given smoothness parameter.

The distance  $d$  can be the Euclidean distance, the 1-norm or the sup norm.

- (\*) **PiecewiseLinearDistribution** as a specialization of **DistributionsImplementation** (see `lib/src/Uncertainty/Distribution`). Given a set of data  $(x_i, y_i)_{i=1, \dots, N}$  in  $\mathbb{R}^n \times \mathbb{R}^p$  the PDF is linear over the data (with due renormalization).

# Library development 2/5

## Projects

- ⑧ (\*) **TawnCopula** as a specialization of `ExtremeValueCopula` (see `lib/src/Uncertainty/Distribution`). This copula is defined by its Pickand function:

$$\forall t \in [0, 1], A(t) = (1 - \psi_1)(1 - t) + (1 - \psi_2)t + \left[ \{\psi_1 t\}^{1/\theta} + \{\psi_2(1 - t)\}^{1/\theta} \right]^\theta \quad (2)$$

where  $0 < \theta \leq 1$  and  $0 \leq \psi_1, \psi_2 \leq 1$ .

# Library development 3/5

## Projects

- (\*\*) **ArchiMaxCopula** as a specialization of `CopulaImplementation` (see `lib/src/Uncertainty/Distribution`). Given an Archimedean copula with generator  $\psi$  and an extreme value copula with Pickand function  $A$ , an archimax copula  $C$  is defined by:

$$\forall (u, v) \in [0, 1]^2, C(u, v) = \psi^{-1} \left( \min \left( \psi(0), [\psi(u) + \psi(v)] A \left( \frac{\psi(u)}{\psi(u) + \psi(v)} \right) \right) \right) \quad (3)$$

It becomes (\*\*\*) if one wants to implement an efficient sampling algorithm.

# Library development 4/5

## Projects

- 13 (\*\*\*) Extend archimedean copulas from 2-d to  $n$ -d. Given a 2-d Archimedean copula with generator  $\psi$ , implement its  $n$ -d counterpart using:

$$\forall (u_1, \dots, u_n) \in [0, 1]^n, C(u_1, \dots, u_n) = \psi^{-1}(\psi(u_1) + \dots + \psi(u_n)) \quad (4)$$

The main difficulties are the architecture of this extension and the implementation of an efficient sampling algorithm.

# Library development 5/5

## Projects

- 16 (\*) Extend **SolverImplementation** and **Solver** to the resolution of systems of nonlinear equations and provide a generic implementation using the **LeastSquaresProblem** class. The solutions  $x^*$  of a nonlinear system of equations  $f_1(x) = 0, \dots, f_n(x) = 0$  where  $x = (x_1, \dots, x_n)$ , if they exist, have to be found in the set of solutions of the following least-squares problem:

$$x^* = \arg \min \sum_{j=1}^n f_j^2(x) \quad (5)$$

for which many solvers are available in OpenTURNS.



## Module development 1/2

### Projects

- 18 (\*) or (\*\*) **CloudMesher**: mesh generation over a cloud of points using kernel mixture, pca, rotation, then levelset mesher on an interval

## Module development 2/2

### Projects

- 22 (\*\*) **CubaIntegration** as a specialization of `IntegrationAlgorithmImplementation` (see `lib/src/Base/algo`). This algorithm is obtained by interfacing the `cuba` C library. A possible name for the module is **OTCuba**.
- 23 (\*\*) **HIntLibIntegration** as a specialization of `IntegrationAlgorithmImplementation` (see `lib/src/Base/algo`). This algorithm is obtained by interfacing the `HIntLib` C++ library, see <https://github.com/JohannesBuchner/HIntLib>. A possible name for the module is **OTHIntLib**.