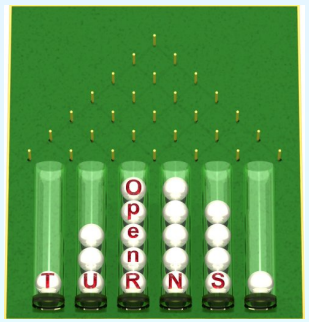
OpenTURNS Developer training: first steps

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

Developers training



OpenTURNS: first steps

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

1. (*) `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:

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

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

Library development 3/5

Projects

16. (**) `AffineTransformDistribution`, the affine transformation of an 1-d distribution.

Library development 4/5

Projects

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

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

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

Library development 5/5

Projects

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

$$\forall (u_1, \ldots, u_n) \in [0,1]^n, C(u_1, \ldots, u_n) = \psi^{-1}\left(\psi(u_1) + \cdots + \psi(u_n)\right) \quad (3)$$

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

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. (**) `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.