OpenTURNS release highlights

Julien Schueller

UserDay #14, 21 June 2021, online event













OpenTURNS et al.

Overview

New features since last year in releases:

• v1.16: fall 2020

• v1.17: spring 2021

OpenTURNS

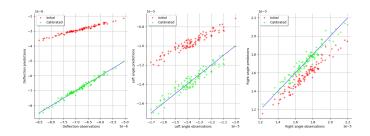
Contents

- Grid graphs
- Probabilistic modelling capabilities
- 3 Karhunen-Loeve
- 4 Covariance models
- Miscellaneous

OpenTURN'

Grid graphs

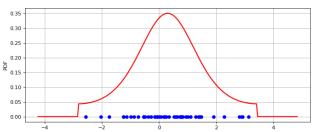
```
graph = ot.GridLayout(d, d)
for i in range(d):
    for j in range(d):
        pdf = dist.getMarginal([i, j]).drawPDF()
        graph.setGraph(i, j, pdf)
```



Von Mises distribution estimation

$$\begin{split} \hat{\mu}_n &= \operatorname{Arg}(z_n) \\ \hat{\kappa}_n & \text{s.t.} \quad \frac{I_1(\hat{\kappa}_n)}{I_0(\hat{\kappa}_n)} = \sqrt{\frac{n}{n-1} \left(|z_n|^2 - \frac{1}{n}\right)} \end{split}$$

VonMises(mu = 0.29867, kappa=1.04484)

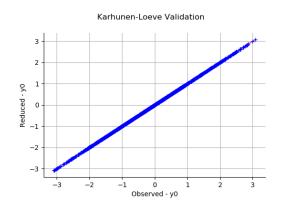


Block independent distribution

```
c1 = ot.NormalCopula(R)
atom1 = ot.ComposedDistribution([ot.Exponential(2.0), ot.WeibullMax(2.0, 2.0)], c1)
c2 = ot.ClaytonCopula(3.0)
atom2 = ot.ComposedDistribution([ot.Normal(2.0, 1.0), ot.Triangular(2.0, 3.0, 4.0)], c2)
distribution = ot.BlockIndependentDistribution([atom1. atom2])
                                                                                         0.000561429
                                                                                         0.00443782
                                                                                                                       0.0494347
         1.50
                                                           0.011961
                                                                                         0.00974982
                                                                                                                       0.0765105
                                                                                         0.0317253
                                                           0.0980555
                                                                                         0.0608184
                                                                                                                     0.302191
                                                           0.196753
                                                                                         0.128429
                                                                                                                       0.510918
                                                           0.438944
                                                                                         0.333562
                                                                                                                       0.980188
         0.75
                                                                                         2.57982e-05
                             0.000100883
                                                                                         0.00018595
                             0.00381176
                                                                                         0.0013511
                                                                                                                       0.0201524
                             0.011961
                                                                                                                       0.037938
                                                                                         0.00679951
                                                                                                                       0.0638487
                             0.196753
                                                                                         0.0632949
                                                                                         0.123802
```

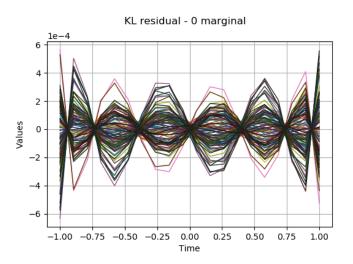
Karhunen-Loeve validation: qqplot

```
algo = ot.KarhunenLoeveSVDAlgorithm(processSample, threshold)
algo.run()
klresult = algo.getResult()
validation = ot.KarhunenLoeveValidation(processSample, klresult)
validation.drawValidation()
```



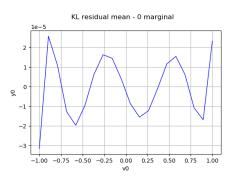
Karhunen-Loeve validation: residual

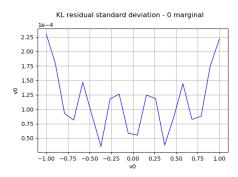
validation.computeResidual().drawMarginal(0)



Karhunen-Loeve validation: residual mean/stddev

validation.computeResidualMean().drawMarginal(0)
validation.computeResidualStandardDeviation().drawMarginal(0)

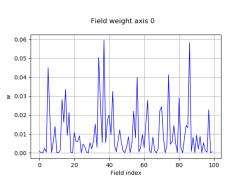




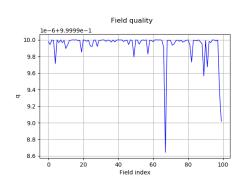
OpenTURNS et al.

Karhunen-Loeve validation: weight/quality of obs

validation.drawObservationWeight(0)
validation.drawObservationQuality()



$$v_k^i = \frac{(\xi_k^{(i)})^2}{\sum_{i=1}^N (\xi_k^{(i)})^2}$$

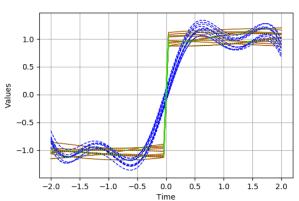


$$q^i = \frac{\left\| \overset{\sim}{\boldsymbol{X}}^i(t) \right\|^2}{\left\| \boldsymbol{X}^i(t) \right\|^2}$$

Karhunen-Loeve reduction

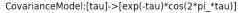
```
reduced1 = ot.KarhunenLoeveReduction(result)(sample)
reduced2 = ot.KarhunenLoeveReduction(result, trend)(sample)
```

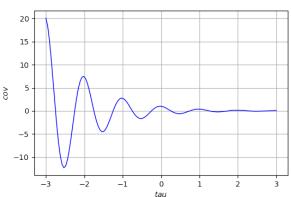
reduction of sign(x) w/o trend - 0 marginal



Covariance models

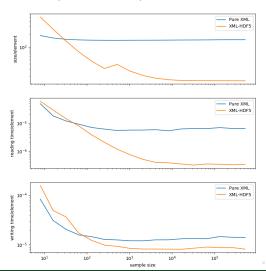
- IsotropicCovarianceModel
- KroneckerCovarianceModel
- StationaryFunctionalCovarianceModel





XML/H5 storage backend

```
study = ot.Study()
study.setStorageManager(ot.XMLH5StorageManager('study.xml'))
study.save() # study.xml + study.h5
```



Various classes

- FejerAlgorithm: intégrate cyclic functions
- MinimumVolumeClassifier: classify according to a density
- KFoldSplitter/LeaveOneOutSplitter: split sample in training/validation parts
- VertexValuePointToFieldFunction: define a filed function from a vectorial one

OpenTURNS

Other improvements

- New discourse forum (rip users@openturns.org)
- New Python wheels for some compiled modules (otmorris, ...)
- Experimental weekly Python binaries of the development version
- More and more bugfixes

OpenTURNS

END

Thank you for your attention! Any questions?

