OpenTURNS release highlights

Julien Schueller

UserDay #15, 10 June 2022, EDF Lab Saclay

Overview

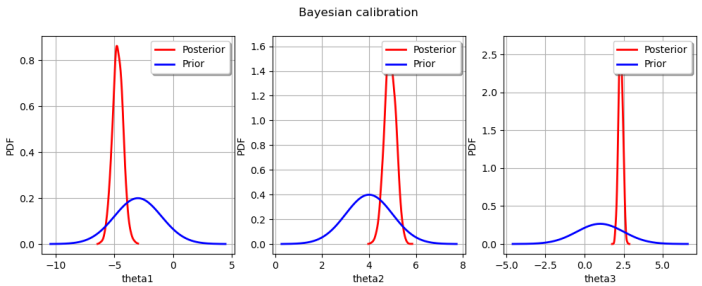
New features since last year in releases:

- v1.18: fall 2021
- v1.19: spring 2022

Contents

Metropolis-Hastings

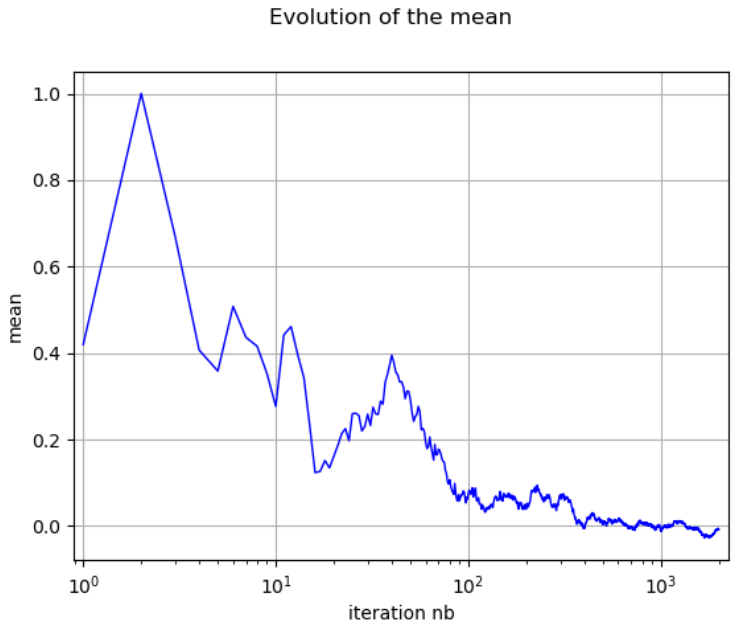
- RandomWalkMH now updates with the instrumental in one go
- new Gibbs class that updates components sequentially
- Separate method to define the likelihood
- Handle improper prior via a log-pdf function
- Direct inconditional sampling according to a random variable




```
from openturns import RandomWalkMetropolisHastings as RWMH
rwmh1 = RWMH(prior, initialState, proposal1, [0])
rwmh2 = RWMH(prior, initialState, proposal2, [1, 2])
mh_coll = [rwmh1, rwmh2]
for mh in mh_coll:
    mh.setLikelihood(conditional, y_obs, linkFunction, x_obs)
sampler = ot.Gibbs(mh_coll)
x = sampler.getSample(1000)
```

Iterative statistics

- Compute statistics without the need to store whole samples
- Moments / threshold exceedance / extrema
- Useful in HPC contexts




```
iter_mom = ot.IterativeMoments(order, dim)
for i in range(size):
    pt = rv.getRealization()
    iter_mom.increment(pt)
variance = iter_mom.getVariance()
skewness = iter_mom.getSkewness()
```

NAIS

- Nonparametric Adaptive Importance Sampling (Morio 2015)
- Adaptive method based on the idea of Importance Sampling

$$\widehat{P}^{\mathrm{IS}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}_{g(\mathbf{x}_i)<T} \frac{h_0(\mathbf{x}_i)}{h(\mathbf{x}_i)}$$

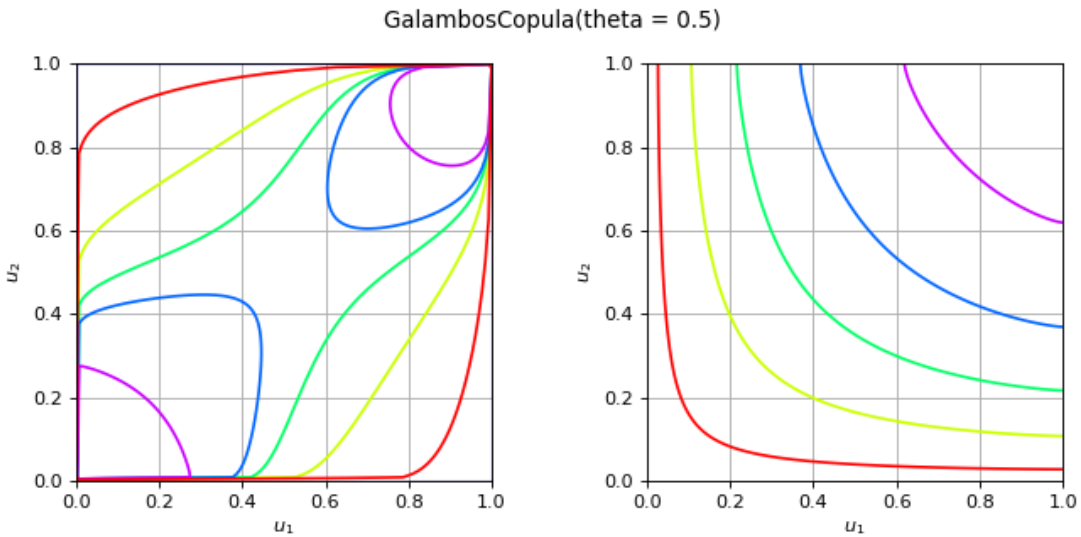
- Updates an importance density from KS (Gauss + Silverman)
- Comparable to subset simulations
- No tuning needed

Galambos copula

- bivariate extreme value copula

$$C(u_1, u_2) = u_1 u_2 \exp\left[(-\log(u_1))^{-\theta} + (-\log(u_2))^{-\theta}\right]^{-1/\theta}$$

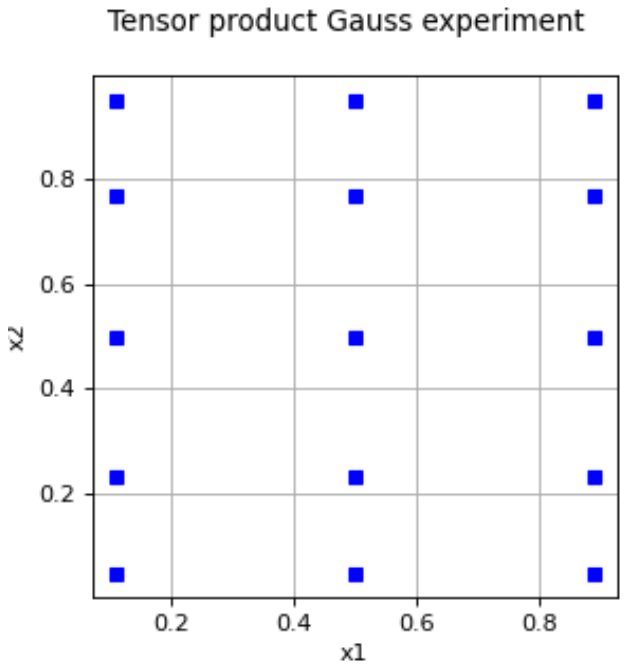
- aims at modelling the dependence of rare events



Tensor product experiment

- Tensorize a set of elementary (d>=1) designs of experiments

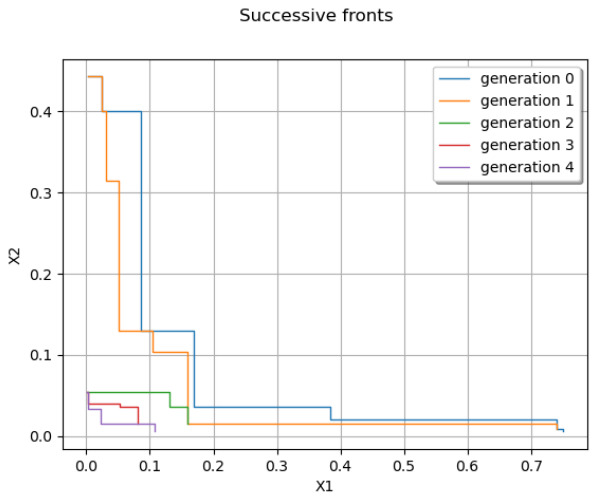
$$\int_{\mathcal{X}} g(\boldsymbol{x}) f(\boldsymbol{x}) d\boldsymbol{x} \approx \sum_{i=1}^{s_t} w_i g\left(\boldsymbol{x}_i\right)$$




```
exp1 = ot.GaussProductExperiment(ot.Uniform(0.0, 1.0), [3])
exp2 = ot.GaussProductExperiment(ot.Uniform(0.0, 1.0), [5])
exp_tens = ot.TensorProductExperiment([exp1, exp2])
nodes, weights = exp_tens.generateWithWeights()
```

Multi-objective optimization

- massively parallel optimization Pagmo library from ESA
- 18 bio-inspired and evolutionary global algorithms
- 4 multi-objective algorithms
- some support batch evaluation / constraints / MINLP




```
pop0 = ot.ComposedDistribution([ot.Uniform(0.0, 1.0)] * 2).getSamp
algo = ot.Pagmo(zdt1, 'nsga2', pop0)
algo.setGenerationNumber(180)
algo.run()
pop1 = algo.getResult().getFinalPoints()
```

Other improvements

- New maintainer in Debian (P. Gruet, EDF)
- Weekly Python wheels for Windows too
- Continued bugfix / documentation / modules effort

END

Thank you for your attention!
Any questions?