Variance-based sensitivity analysis for functional inputs

Methodology

First build a metamodel of the application h is known from n fields $(\boldsymbol{X_1}, \ldots, \boldsymbol{X_n})$ of the associated input process $\boldsymbol{X}$ and n vectors $(\boldsymbol{Y_1}, \ldots, \boldsymbol{Y_n})$

$$h : \left| \begin{array}{ccc} \mathcal{M}_N \times (\mathbb{R}^d)^N & \rightarrow & \mathbb{R}^p \\ \boldsymbol{X} & \mapsto & \boldsymbol{Y} \end{array} \right.$$

We use the Karhunen-Loeve decomposition to find the $(\lambda_k, \boldsymbol{\varphi}_k)_{k \geq 1}$ solutions of the Fredhlom equation:

$$\int_{\mathcal{D}} \boldsymbol{C}(\boldsymbol{s}, \boldsymbol{t}) \boldsymbol{\varphi}_k(\boldsymbol{t}) \, d\boldsymbol{t} = \lambda_k \boldsymbol{\varphi}_k(\boldsymbol{s}) \quad \forall \boldsymbol{s} \in \mathcal{D}$$

The SVD decomposition helps to approach the covariance function $\boldsymbol{C}$ by its empirical estimator.

Variance-based sensitivity analysis for functional inputs

Methodology

The linear projection function $\pi_{\boldsymbol{\lambda},\varphi}$ of the Karhunen-Loeve decomposition writes:

$$\pi_{\boldsymbol{\lambda},\varphi} : \left| \begin{array}{ccc} L^2(\mathcal{D},\mathbb{R}^d) & \rightarrow & \mathcal{S}^{\mathbb{N}} \\ f & \mapsto & \left(\frac{1}{\sqrt{\lambda_k}} \int_{\mathcal{D}} f(\boldsymbol{t})\varphi_k(\boldsymbol{t})\, d\boldsymbol{t} \right)_{k\geq 1} \end{array} \right.$$

This integral is replaced by a specific weighted and finite sum and to write the projections of the j-th marginal of i-th input field $\boldsymbol{X_i^j}$ by multiplication with the projection matrix $\boldsymbol{M^j} \in \mathbb{R}^{K_j} \times \mathbb{R}^{Nd}$:

$$\boldsymbol{M_j X_i^j} = \begin{pmatrix} \xi_1^j \\ \dots \\ \xi_{K_j}^j \end{pmatrix} \in \mathbb{R}^{K_j}, \forall i \in [1,n], \forall j \in [1,d]$$

with K_j the retained number of modes in the decomposition of the j-th input

Variance-based sensitivity analysis for functional inputs

Methodology

The projections of all the d components of n fields are assembled in the Q matrix:

$$\boldsymbol{Q} = \boldsymbol{M}\boldsymbol{X} = \begin{pmatrix} \boldsymbol{M_1}\boldsymbol{X^1} \\ \dots \\ \boldsymbol{M_d}\boldsymbol{X^d} \end{pmatrix} \in \mathbb{R}^{K_T} \times \mathbb{R}^{\boldsymbol{n}}$$

with $K_T = \sum_{j=1}^{d} K_j$ the total number of modes accross input components

Variance-based sensitivity analysis for functional inputs

Methodology

Then a functional chaos decomposition is built between the projected modes sample $\boldsymbol{Q}$ and the output samples $\boldsymbol{Y}$

$$\tilde{g}(x) = \sum_{k=1}^{K_c} \beta_{\boldsymbol{\alpha}_k} \Psi_{\boldsymbol{\alpha}_k}(x)$$

The final metamodel consists in the composition of the Karhunen-Loeve projections and the functional chaos metamodel.

$$\tilde{h} : \left| \begin{array}{ccccc} \mathcal{M}_N \times (\mathbb{R}^d)^N & \to & \mathbb{R}^{K_T} & \to & \mathbb{R}^p \\ \boldsymbol{X} & \mapsto & \boldsymbol{Q} & \mapsto & \boldsymbol{Y} \end{array} \right.$$

A limitation of this approach is that the projected modes sample has a dimension K_T so the dimension of the input fields $\boldsymbol{X_i}$ and the associated number of modes must remain modest.

Variance-based sensitivity analysis for functional inputs

Methodology

From the chaos decomposition:

$$\tilde{g}(x) = \sum_{k=1}^{K_c} \beta_{\boldsymbol{\alpha}_k} \Psi_{\boldsymbol{\alpha}_k}(x)$$

Lets expand the multi indices notation:

$$\Psi_{\boldsymbol{\alpha}}(x) = \prod_{j=1}^{K_T} P^j_{\alpha_j}(x_j)$$

with α that contains the marginal degrees associated to the K_T input components

$$\boldsymbol{\alpha} \in \mathbb{N}^{K_T} = \{\underbrace{\alpha_1, \dots, \alpha_{K_1}}_{K_1}, \dots, \underbrace{\alpha_{K_T - K_d}, \dots, \alpha_{K_T}}_{K_d}\}$$

Variance-based sensitivity analysis for functional inputs

Methodology

Sobol indices of the input field component $j \in [1, d]$ can be computed from the coefficients of the chaos decomposition that involve the matching KL coefficients.

For the first order Sobol indices we sum over the multi-indices $\boldsymbol{\alpha}_k$ that are non-zero on the K_j indices corresponding to the KL decomposition of j-th input and zero on the other $K_T - K_j$ indices (noted G_j):

$$S_j = \frac{\sum_{k=1, \boldsymbol{\alpha}_k \in G_j}^{K_c} \beta_{\boldsymbol{\alpha}_k}^2}{\sum_{k=1}^{K_c} \beta_{\boldsymbol{\alpha}_k}^2}$$

For the total order Sobol indices we sum over the multi-indices $\boldsymbol{\alpha}_k$ that are non-zero on the K_j indices corresponding to the KL decomposition of the j-th input (noted GT_j):

$$S_{T_j} = \frac{\sum_{k=1, \boldsymbol{\alpha}_k \in GT_j}^{K_c} \beta_{\boldsymbol{\alpha}_k}^2}{\sum_{k=1}^{K_c} \beta_{\boldsymbol{\alpha}_k}^2}$$

This generalizes to higher order indices.

Variance-based sensitivity analysis for functional inputs

Implementation

```
algo = ot.FieldToPointFunctionalChaosAlgorithm(x, y)  # x~ProcessSample, y~Sample

# 1. KL parameters
algo.setCenteredSample(False)  # our input sample is not centered (default)
algo.setThreshold(4e-2)  # we expect to explain 96% of variance
algo.setRecompress(False)  # whether to re-truncate modes
algo.setNbModes(10) # max KL modes (default=unlimited)

# 2. chaos parameters:
ot.ResourceMap.SetAsUnsignedInteger('FunctionalChaosAlgorithm-BasisSize', N) # chaos ba
algo.setSparse(True)

algo.setBlockIndices([[0], [1], [2, 3]])  # possibility to group inputs
algo.run()
```

Variance-based sensitivity analysis for functional inputs

Implementation

```python
result = algo.getResult()

# inspect eigen values
kl_results = result.getInputKLResultCollection()
n_modes = [len(res.getEigenvalues()) for res in kl_results]

# validate KL decompositions
for i in range(in_dim):
    View(ot.KarhunenLoeveValidation(x.getMarginal(i), kl_results[i]).drawValidation())

# inspect chaos residuals
print(result.getFCEResult().getResiduals())
print(result.getFCEResult().getRelativeErrors())

# validate chaos decomposition
validation = ot.MetaModelValidation(result.getModesSample(), result.getOutputSample(),
View(validation.drawValidation())
```

Variance-based sensitivity analysis for functional inputs

Implementation

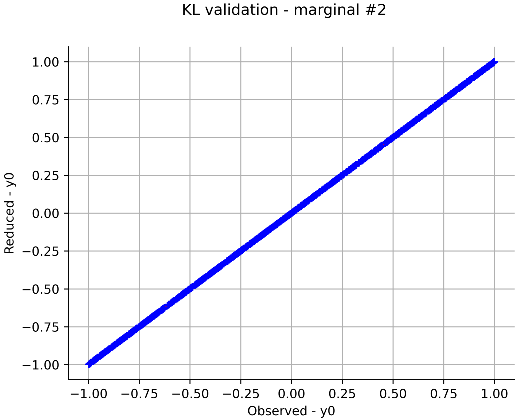


Figure: KL validation

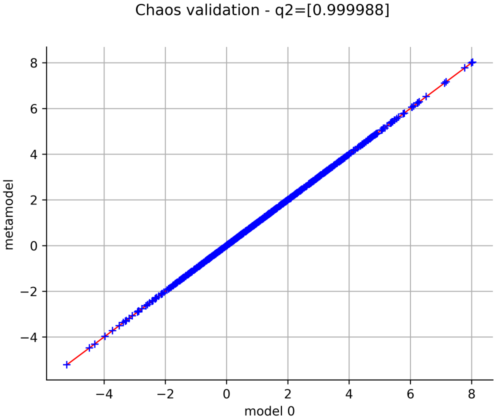


Figure: chaos validation

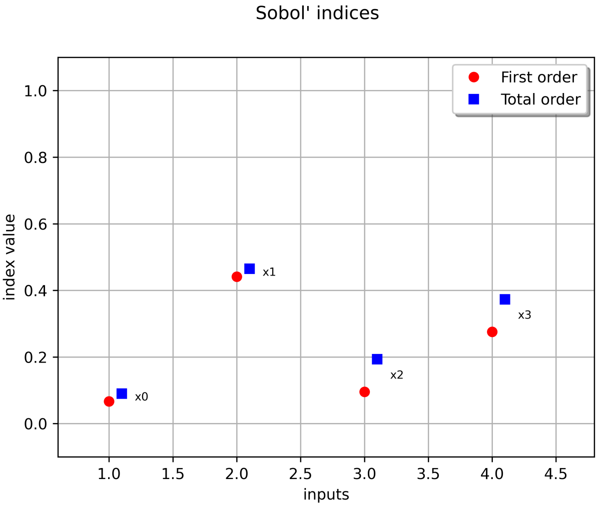
Variance-based sensitivity analysis for functional inputs

Implementation

```python
# evaluate metamodel
metamodel = result.getFieldToPointMetamodel()
y0hat = metamodel(x[0])

# retrieve Sobol indices
sobol = ot.FieldFunctionalChaosSobolIndices(result)
sobol_1 = sobol.getFirstOrderIndices()
sobol_t = sobol.getTotalOrderIndices()

# plot indices
View(sobol.draw())

# higher order indices
sobol12 = sobol.getSobolIndex([0, 1])
```

Variance-based sensitivity analysis for functional inputs

Implementation



Variance-based sensitivity analysis for functional inputs

Outlook

- Development is settling down
- Expected to land in OT 1.20 (fall 2022)