

Variance-based sensitivity analysis for functional inputs

Methodology

We observe an application h from n fields $(\mathbf{X}_1, \dots, \mathbf{X}_n)$
of the associated input process \mathbf{X} and n vectors $(\mathbf{Y}_1, \dots, \mathbf{Y}_n)$

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

We propose the following steps to lead to sensitivity analysis.

- ▶ 1. Identify blocks of independent inputs
- ▶ 2. Dimension reduction via Karhunen-Loeve for each input block
- ▶ 3. Approximate of the link between KL coefficients and vectorial outputs by chaos
- ▶ 4. Post-process functional chaos coefficients to derive Sobol' indices

Variance-based sensitivity analysis for functional inputs

Methodology step 1/4: Identify blocks

Identify groups of independent components in \mathbf{X}
by construction, if the underlying process is known

Variance-based sensitivity analysis for functional inputs

Methodology step 2/4: Dimension reduction by Karhunen-Loeve

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

$$\int_{\mathcal{D}} \mathbf{C}(\mathbf{s}, \mathbf{t}) \varphi_k(\mathbf{t}) d\mathbf{t} = \lambda_k \varphi_k(\mathbf{s}) \quad \forall \mathbf{s} \in \mathcal{D}$$

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

Variance-based sensitivity analysis for functional inputs

Methodology step 2/4: Dimension reduction by Karhunen-Loeve

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

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

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 \mathbf{X}_i^j by multiplication with the projection matrix $\mathbf{M}_j^i \in \mathbb{R}^{K_j} \times \mathbb{R}^{Nd}$:

$$\mathbf{M}_j^i \mathbf{X}_i^j = \begin{pmatrix} \xi_1^j \\ \dots \\ \xi_{K_j}^j \end{pmatrix} \in \mathbb{R}^{K_j}, \forall i \in [1, n], \forall j \in [1, d]$$

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

Variance-based sensitivity analysis for functional inputs

Methodology step 2/4: Dimension reduction by Karhunen-Loeve

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

$$\mathbf{Q} = \mathbf{MX} = \begin{pmatrix} \mathbf{M}_1 \mathbf{X}^1 \\ \dots \\ \mathbf{M}_d \mathbf{X}^d \end{pmatrix} \in \mathbb{R}^{K_T} \times \mathbb{R}^n$$

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 step 3/4: Link KL coefficients to outputs

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

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

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

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

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

Variance-based sensitivity analysis for functional inputs

Methodology step 3/4: Link KL coefficients to outputs

From the chaos decomposition:

$$\tilde{g}(x) = \sum_{k=1}^{K_c} \beta_{\alpha_k} \psi_{\alpha_k}(x)$$

Lets expand the multi indices notation:

$$\psi_{\alpha}(x) = \prod_{j=1}^{K_T} P_{\alpha_j}^j(x_j)$$

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

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

Variance-based sensitivity analysis for functional inputs

Methodology step 4/4: Derive Sobol' indices from chaos coefficients

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 α_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):

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

For the total order Sobol indices we sum over the multi-indices α_k that are non-zero on the K_j indices corresponding to the KL decomposition of the j -th input (noted GT_j):

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

This generalizes to higher order indices.

Variance-based sensitivity analysis for functional inputs

API

```
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

API

```
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

API

KL validation - marginal #2

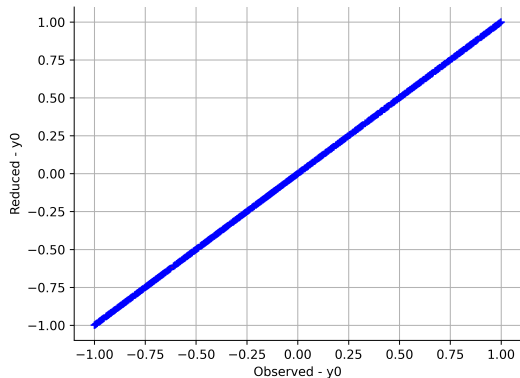


Figure: KL validation

Chaos validation - $q_2=[0.999988]$

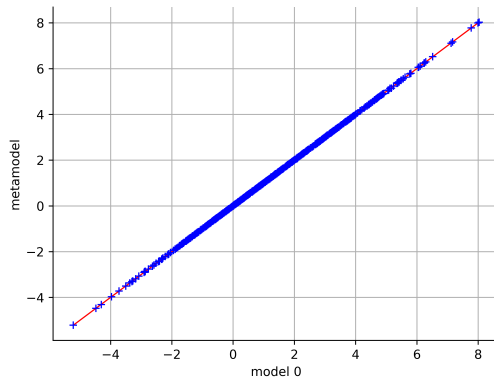


Figure: chaos validation

Variance-based sensitivity analysis for functional inputs

API

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

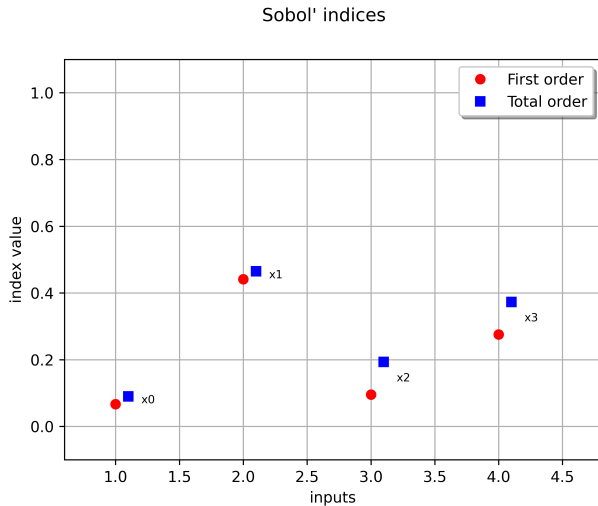
# retrieve Sobol indices
sobol = ot.FieldFunctionalChaosSobolIndices(result)
sobol_1 = sobol.getFirstOrderIndices()
sobol_t = sobol.getTotalOrderIndices()

# plot indices
View(sobol.draw())

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

Variance-based sensitivity analysis for functional inputs

API



Variance-based sensitivity analysis for functional inputs

Outlook

- ▶ Development is settling down
- ▶ Expected to land in OT 1.20 (fall 2022)
- ▶ Extension to Vector \mapsto Field, Field \mapsto Field ?