Variance-based sensitivity analysis for functional inputs

Application to pilot's actions

Catalina OBANDO, Regis LEBRUN, Sofiane HADDAD (Airbus)
Julien Schueller (Phimeca)
10 of June 2022

Content

1. Variance-based sensitivity analysis for functional inputs
2. Use case : Pilot actions
3. OT implementation
4. Preliminary results

Variance-based sensitivity analysis for functional inputs

Methodology

We observe an application h 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 propose the following steps to lead to sensitivity analysis.

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

AIRBUS

Variance-based sensitivity analysis for functional inputs

Methodology

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

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

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

Variance-based sensitivity analysis for functional inputs

Methodology: Step 1/3

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

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

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 $\boldsymbol{M^j} \in \mathbb{R}^{K_j} \times \mathbb{R}^{Nd}$:

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

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

Variance-based sensitivity analysis for functional inputs

Methodology: 1/3

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

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

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

AIRBUS

Variance-based sensitivity analysis for functional inputs

Methodology: Step 2/3

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

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

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

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

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: Step 2/3

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

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

Variance-based sensitivity analysis for functional inputs

Methodology: Step 3/3

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.

[Airbus Amber]

Use Case : Human monitoring in the cockpit

Context: Pilot manual actions during cruise flight phase and autopilot deactivated

Objective: build a system that flags inconsistent pilot actions on the stick given the context of the flight

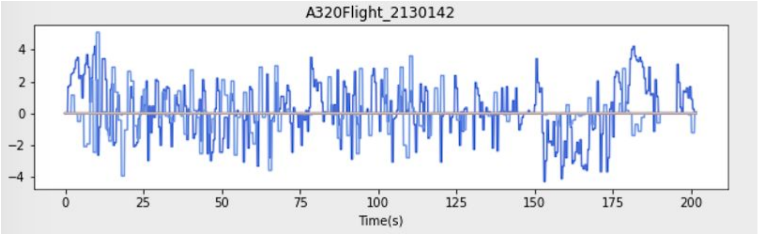
Current solution is based on four criteria define through expert knowledge capturing:

- Piloting activity
- Commands deemed excessive and persistent in pitch
- Commands deemed excessive and persistent in roll
- "Non-normality" compared to reference flights

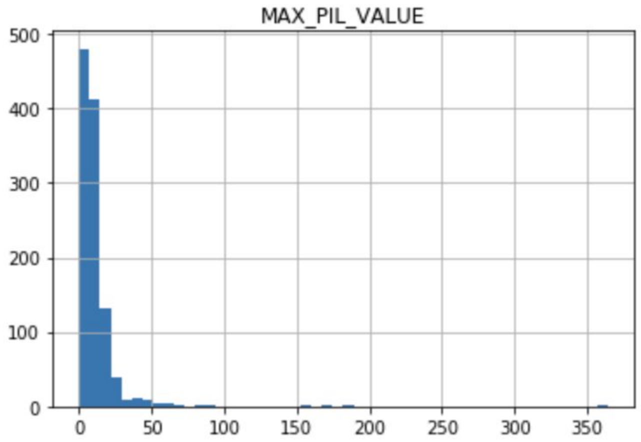
Then each component is multiply by a gain and the and the P value is taken as the maximum at each time point

Drawbacks:

- The ordering is not always right
- Timewise value

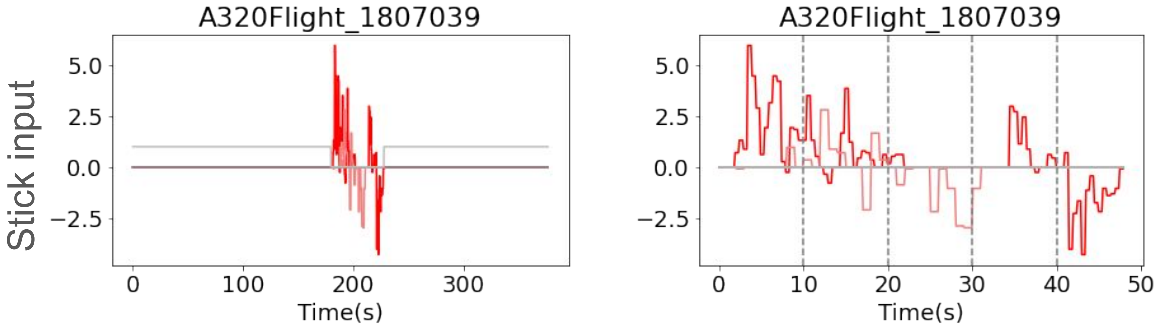


GENERIC NAME	NAME	Description
ALT	ALT	Altitude (1013.25mb)
AP1E	AP1E	AP1 engaged
AP2E	AP2E	AP2 engaged
CAS	CAS	Computed Airspeed
LATG	LATG	Lateral Load Factor
LONG	LONG	Longitudinal Load Factor
MACH	MACH/CMACH/MNADC	Mach number
PTCH	PTCH	Pitch
ROLL	ROLL	Roll
SAT	SAT, SAT2, SAT3, SAT4, CSAT	Static Air Temperature
SFLP	SFLP, FLAPPOS FLAP LEVER P(	Slat Flap Lever Position
STKPC	STKPC	Stick Pitch Capt
STKPF	STKPF	Stick Pitch F/O
STKRC	STKRC	Stick Roll Capt
STKRF	STKRF	Stick Roll F/O
TLA1	TLA1	SELECTED THROTTLE LEVER ANGLE



Proposed solution

- Take into account the inter dependencies of variables
- Exploits the temporal nature of data
- Output a temporal index of (ab)normality



Preprocessing

- Select variables of interest (expert knowledge), for each flight segment time window where AP disconnection occurs
- Rescale to [-1,1]
- Fix window of 6 seconds, sliding window with 50% overlap

Multivariate field $\boldsymbol{x} = (x_1, x_2, ..., x_n)$

$$x_i \in \{0, 1, ..., N\} \times \mathbb{R}$$

n

parameters = 9

N

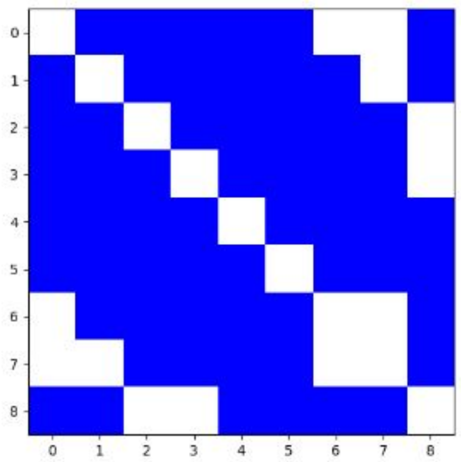
time steps = 48

AIRBUS

Properties

- Stationary processes

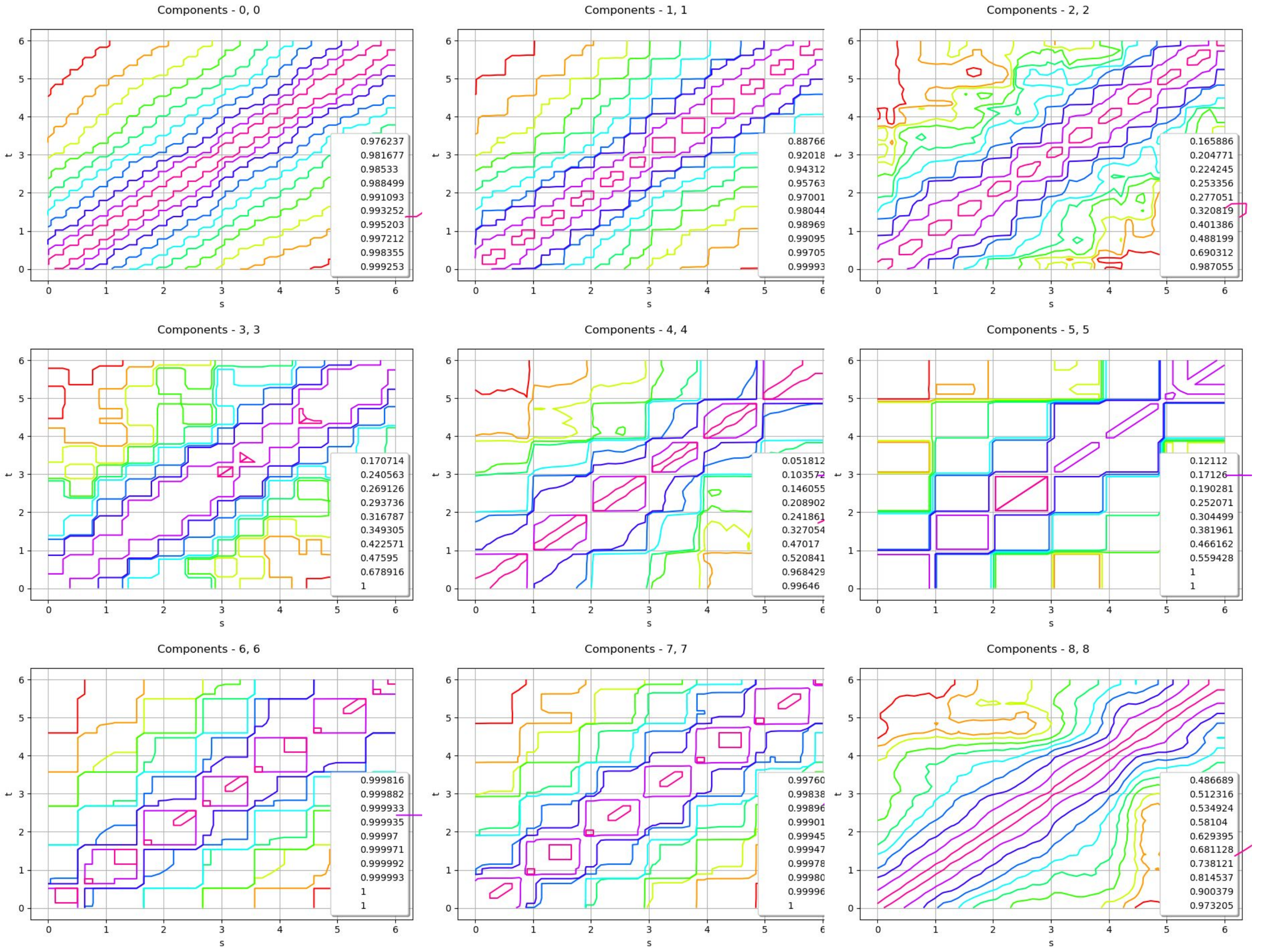
- Bloc structure of independence with threshold τ



Marginals

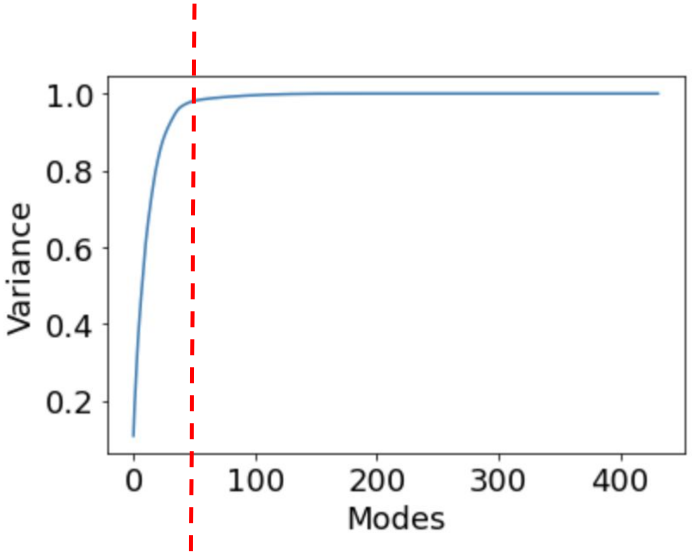
Blocs = [0, 1, 6, 7], [2, 3, 8], [4], [5]

Process sample correlation



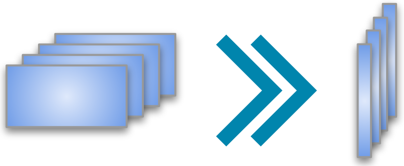
AIRBUS

KL validation

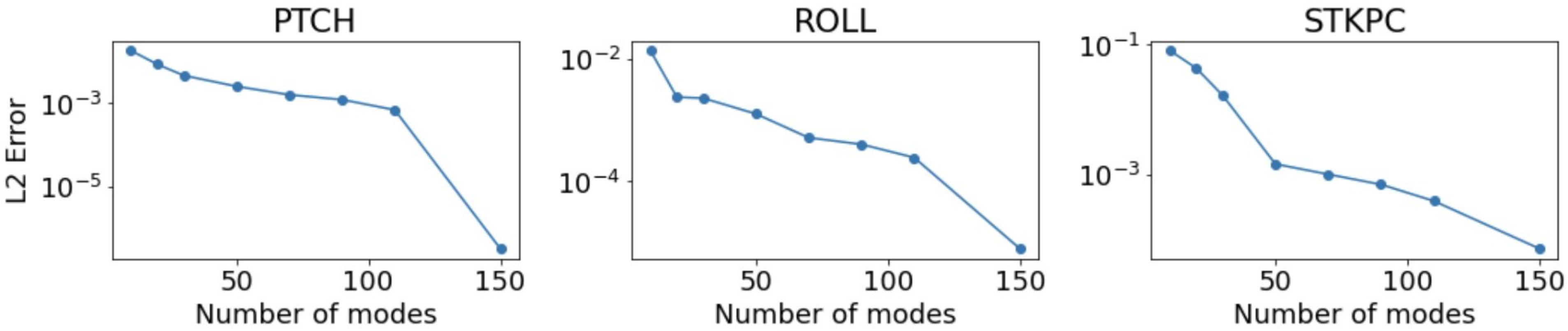


Dimensionality reduction

$$Y(x_1, x_2, ..., x_n, t, \omega) = \sum_{i=1}^{\infty} \sqrt{\lambda_i} * \phi_i(x_1, x_2, ...x_n, t) * \xi_i(\omega)$$



Truncate dimension d

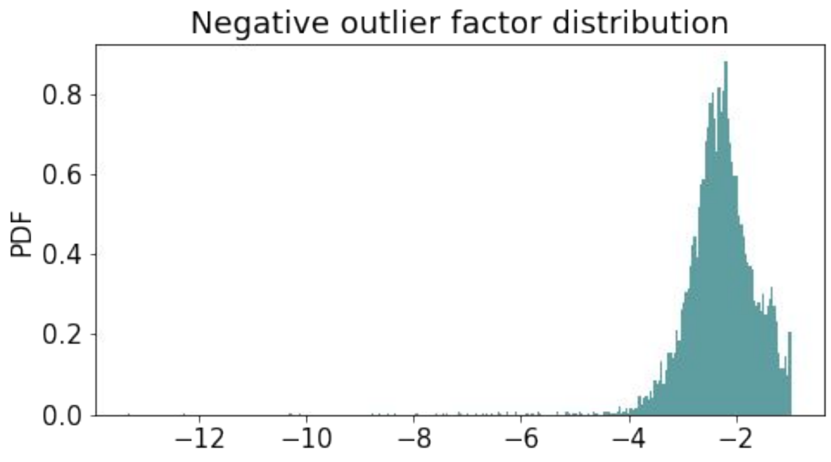


Methodology : Anomaly detection

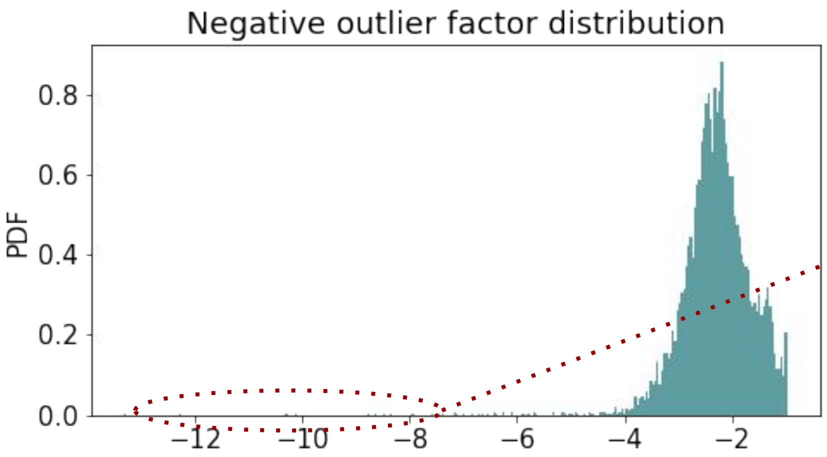
- Build an ordering on the reduced space of dimension d such that abnormal observations obtain lower anomaly scores

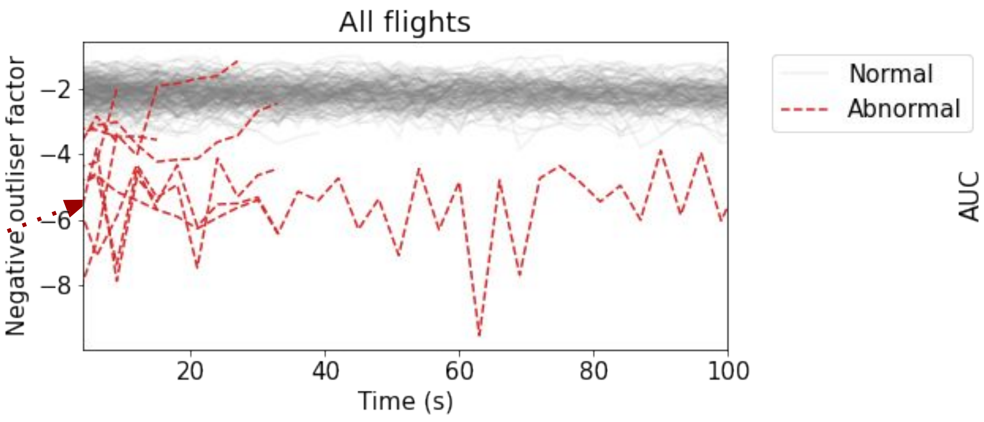
$$\mathbb{R}^d \to \mathbb{R} : \boldsymbol{x} \mapsto \boldsymbol{g}(\boldsymbol{x})$$

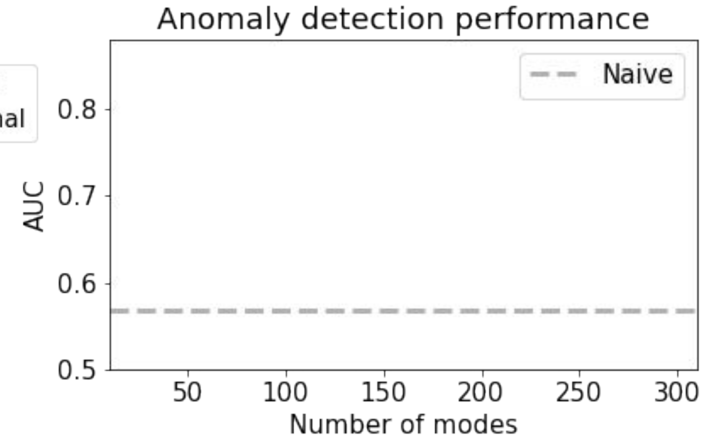
- Vectors in the compressed space are order by the Negative Outlier Factor (NOF)
- A vector is abnormal if it is ‘far’ from the NOF distribution
- A flight trajectory is flagged if at least one segment is abnormal

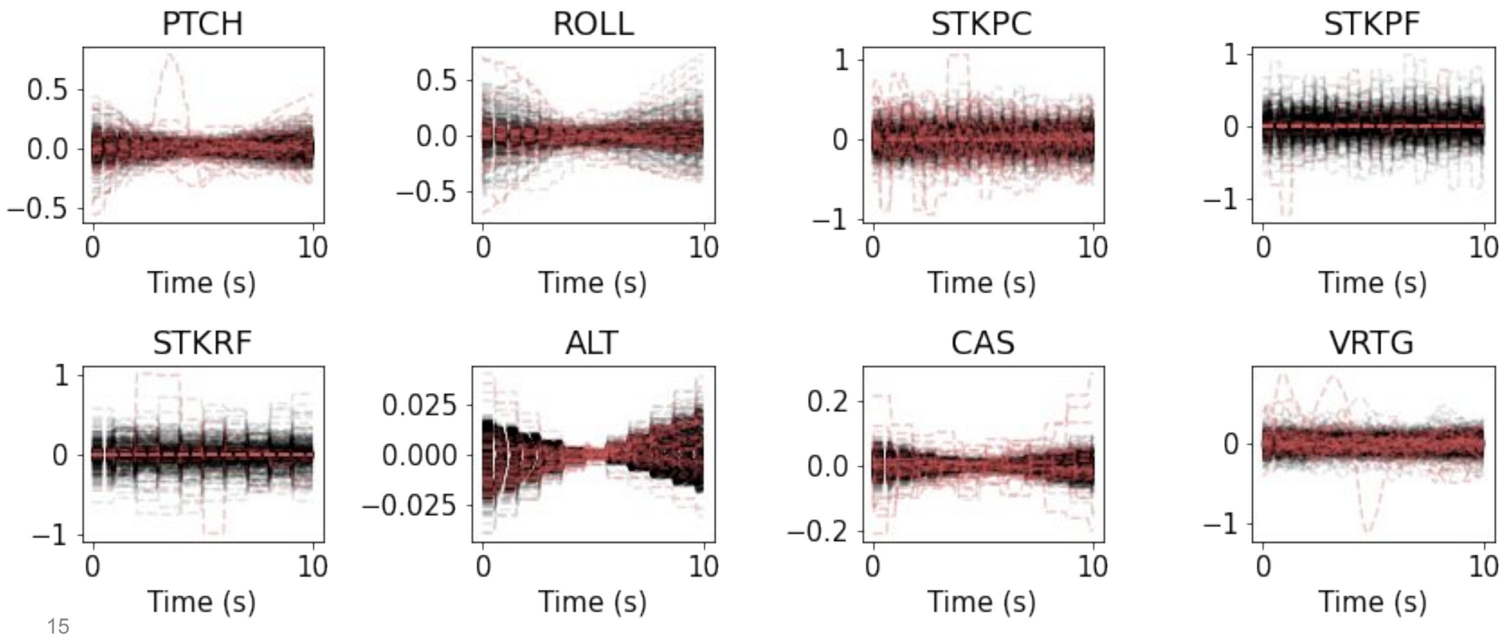


Results : Anomaly detection



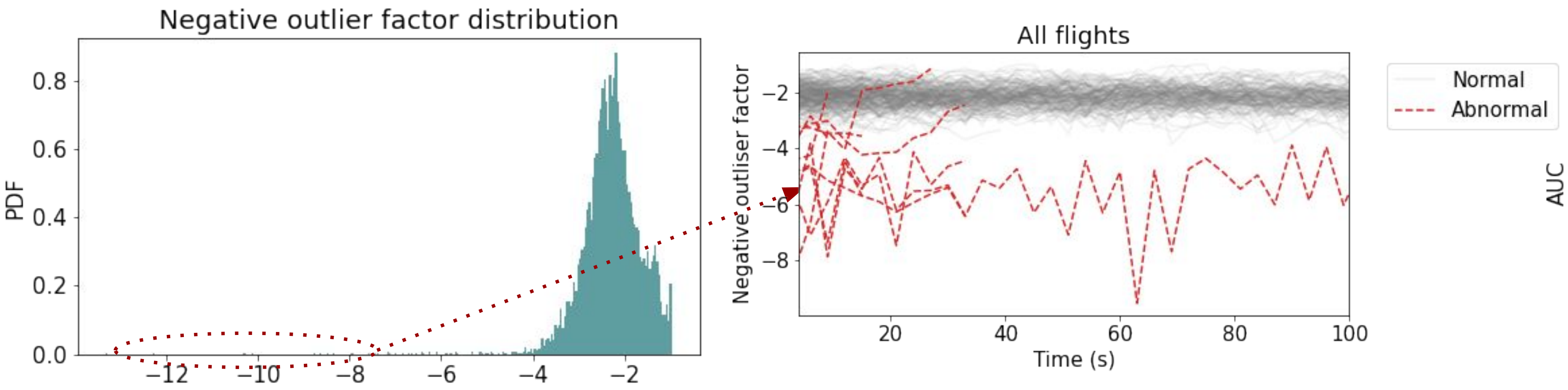


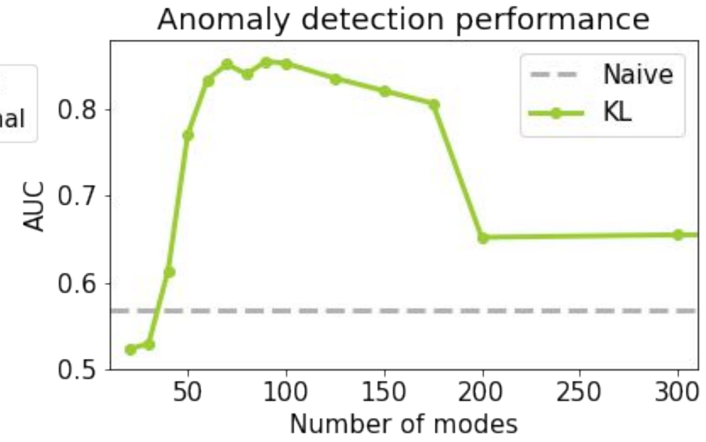


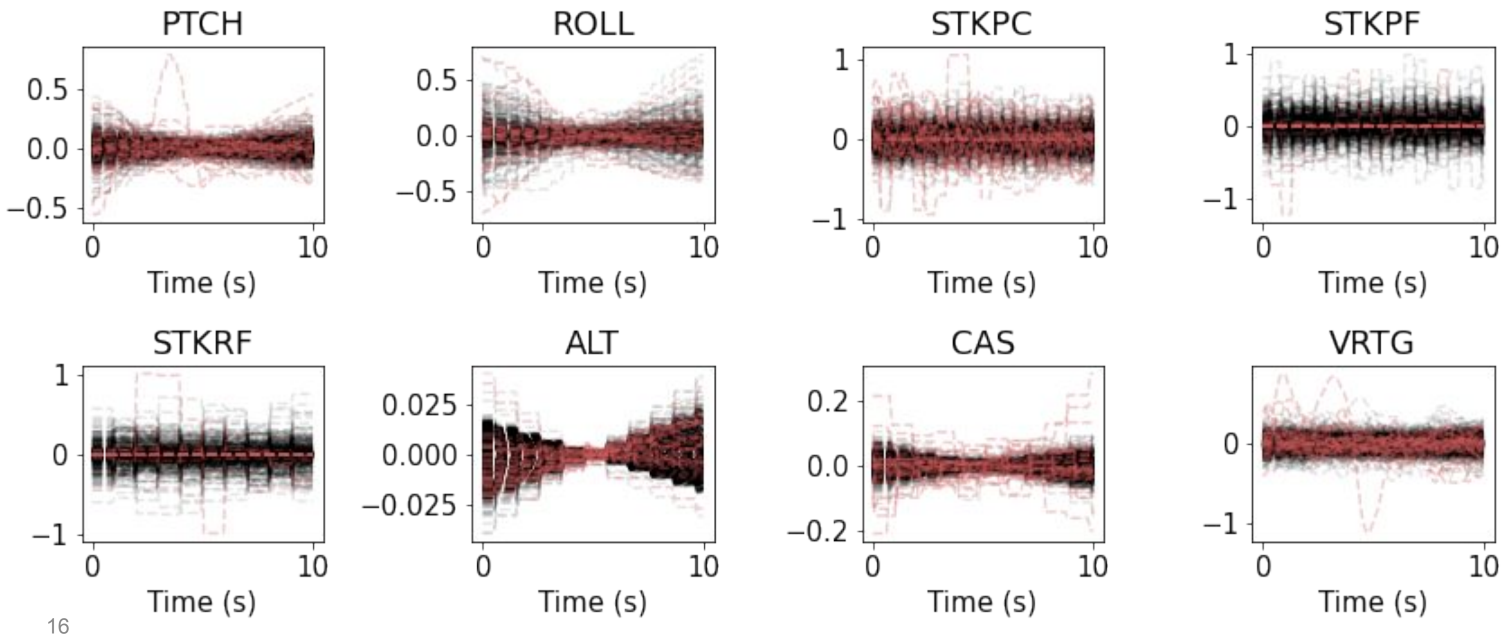


➢ Quantify the influence of each variable on the NOF score -> variance based sensitivity analysis

Results : Anomaly detection







➢ Quantify the influence of each variable on the NOF score -> variance based sensitivity analysis

AIRBUS

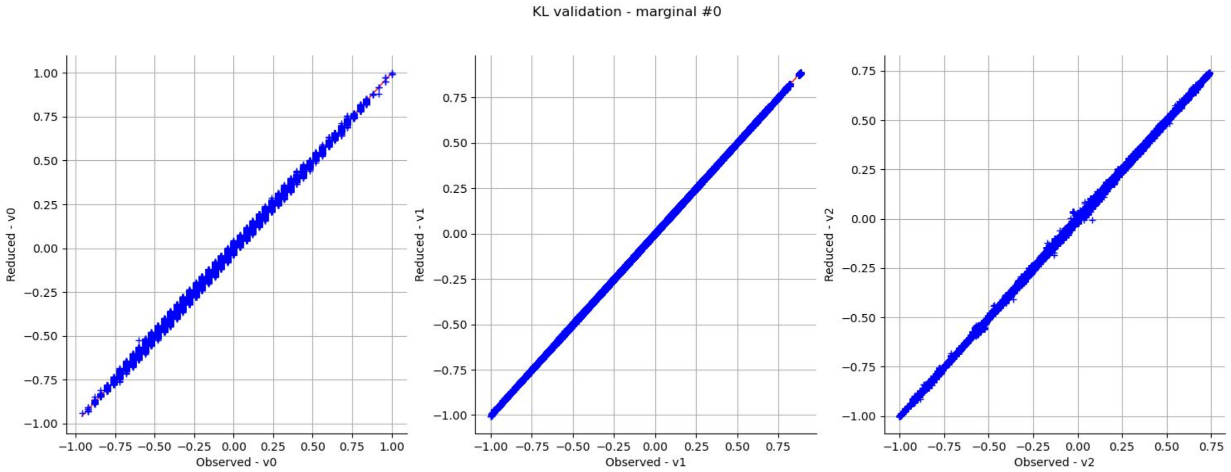
Variance-based sensitivity analysis implementation

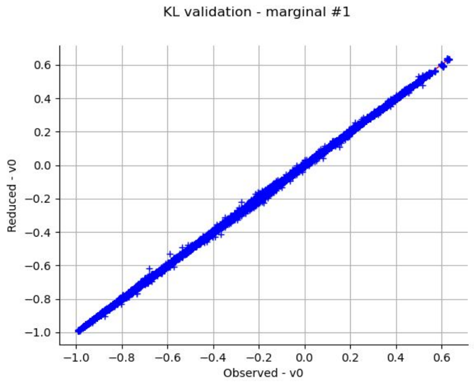
```
algo = ot.FieldToPointFunctionalChaosAlgorithm(x, y)  # x~ProcessSample, y~Sample

# 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) # cha
algo.setSparse(True)

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

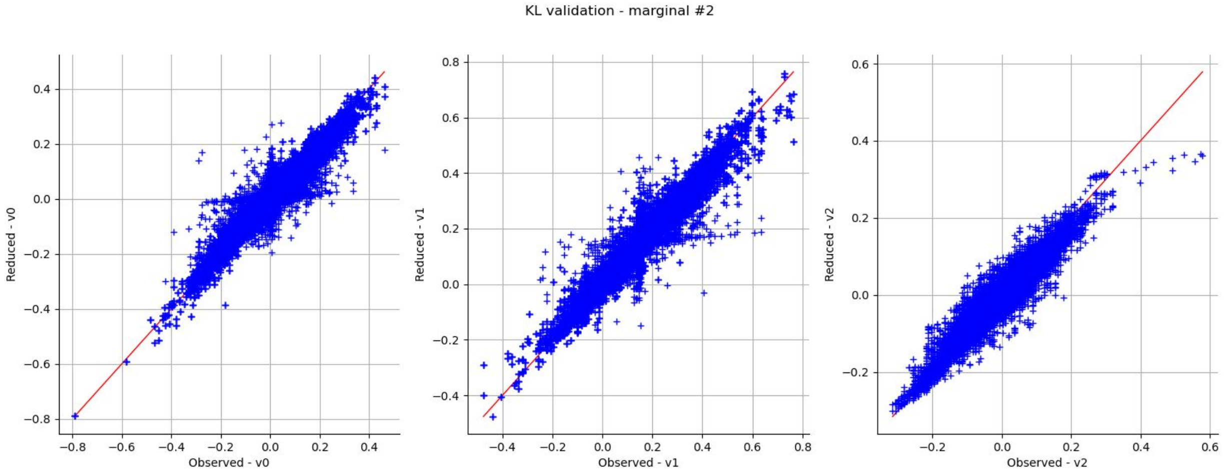
Variance-based sensitivity analysis implementation

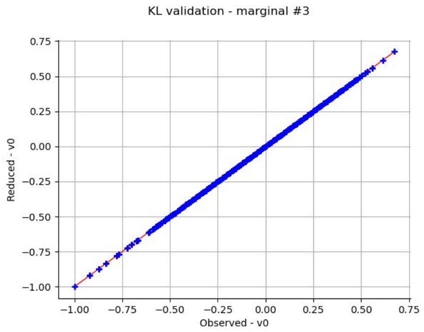
```
result = algo.getResult()

# 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())
```

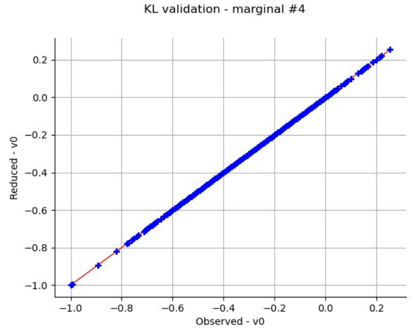
Results











Validation of KL decomposition in independent blocs for nModes= 20, recompress = False

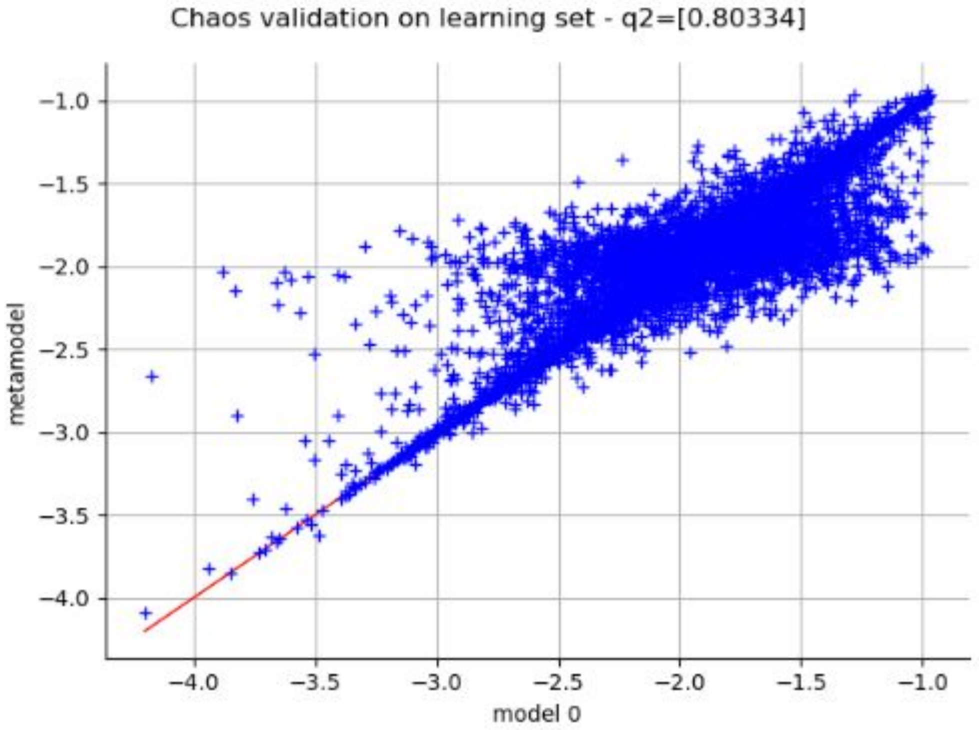
Results

Chaos considerations:

- Basis size : determines, together with number of modes, the order of the polynomial and affects execution time
  38 modes, basis size 13000, degree 3

- Sparse vs full chaos
  Sparse : 20 sec vs full: 240 sec

- Metamodel learnt in a bounded support but test data can fall outside this support

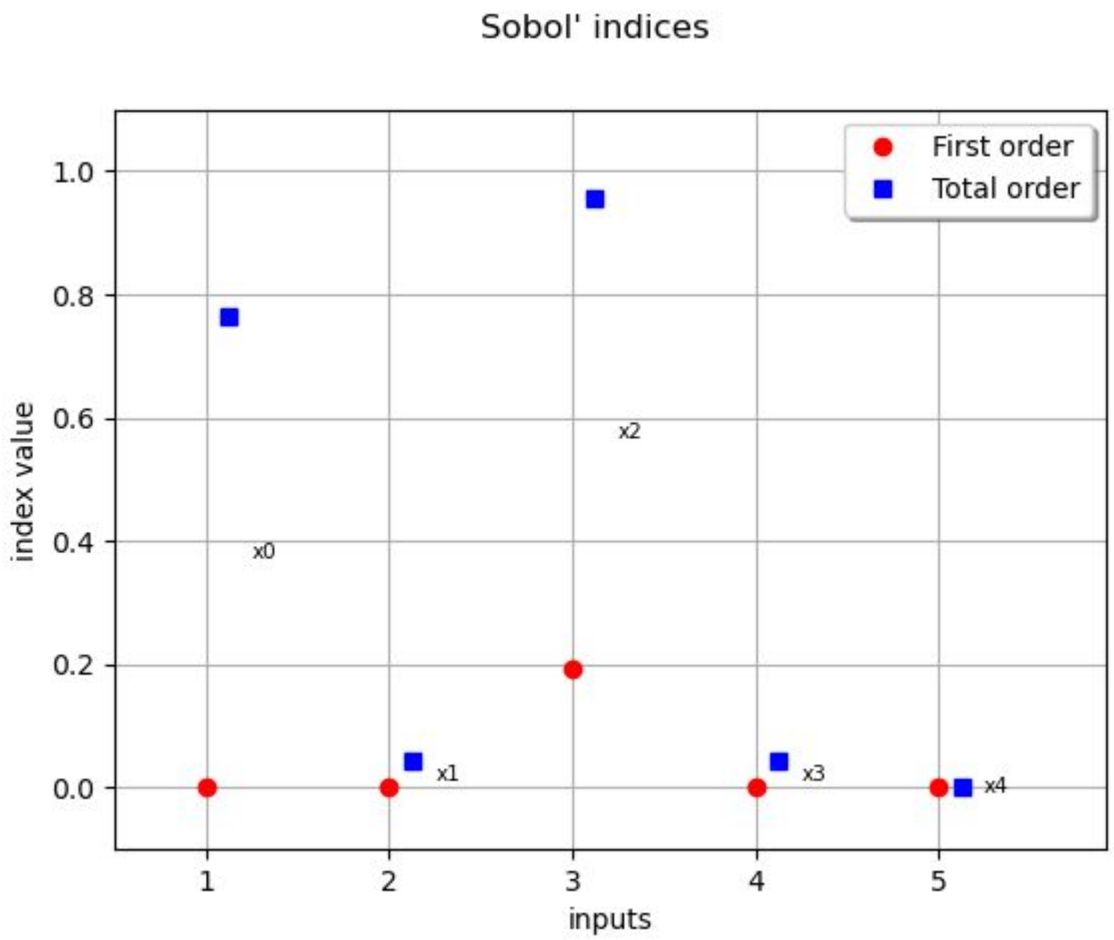


AIRBUS

Variance-based sensitivity analysis implementation

```
# 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])
```

Results



Independent components

Outlook

- Tailor strategy behind chaos algorithm to the use case
- Development is settling down
- Expected to land in OpenTURNS 1.20 (fall 2022)
- Extension to Vector -> Field, Field-> Field ?

Thank you