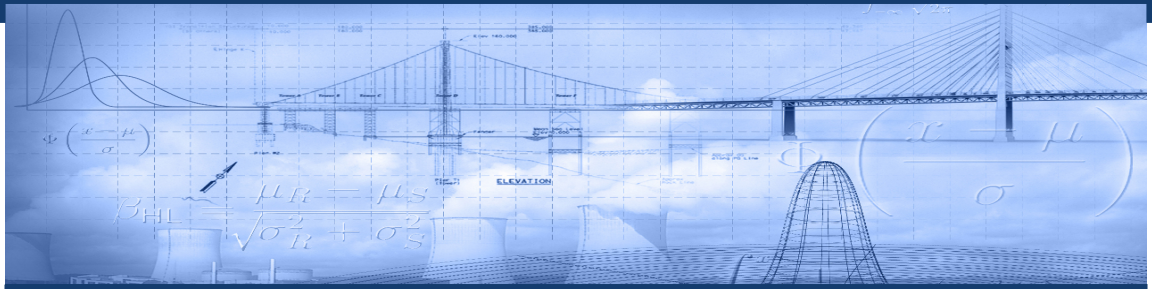
# Sparse polynomial chaos expansions for uncertainty quantification in applied sciences

ETH zürich




# Sparse polynomial chaos expansions for uncertainty quantification in applied sciences

Bruno Sudret

# How to cite?

This presentation is an invited lecture given at the Workshop on "*Uncertainty Quantification for High-Dimensional Problems*" at CWI, Amsterdam (The Netherlands) on November 11, 2024.

## How to cite

Sudret, B. *Sparse polynomial chaos expansions for uncertainty quantification in applied sciences*, Workshop on Uncertainty Quantification for High-Dimensional Problems, CWI, Amsterdam (The Netherlands), Invited Lecture, November 11, 2024.

**Impressions of Amsterdam**

# Computational models in engineering

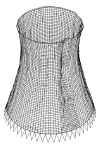
Complex engineering systems are designed and assessed using computational models, a.k.a simulators

A computational model combines:

- A mathematical description of the physical phenomena (governing equations), *e.g.* mechanics, electromagnetism, fluid dynamics, etc.

$$\begin{aligned}\text{div}\ \boldsymbol{\sigma} + \boldsymbol{f} &= \mathbf{0} \\ \boldsymbol{\sigma} &= \mathbf{D} \cdot \boldsymbol{\varepsilon} \\ \boldsymbol{\varepsilon} &= \frac{1}{2}\left(\nabla \boldsymbol{u} + {}^{\mathsf{T}}\nabla \boldsymbol{u}\right)\end{aligned}$$

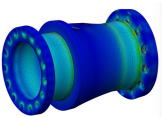
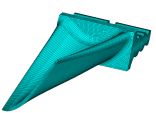
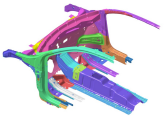
- Discretization techniques which transform continuous equations into linear algebra problems
- Algorithms to solve the discretized equations



# Computational models in engineering

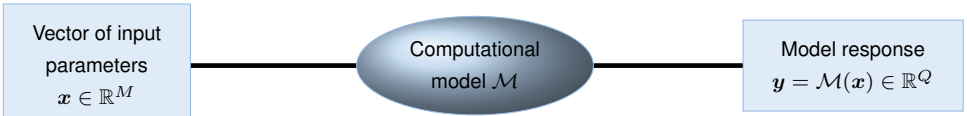
Computational models are used:

- To explore the design space (“virtual prototypes”)
- To optimize the system (*e.g.* minimize the mass) under performance constraints
- To assess its robustness w.r.t uncertainty and its reliability
- Together with experimental data for calibration purposes

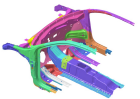





# Computational models: the abstract viewpoint

A computational model may be seen as a black box program that computes quantities of interest (QoI) (a.k.a. model responses) as a function of input parameters



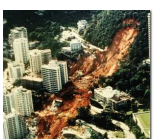
# Real world is uncertain



- Differences between the designed and the real system:
  - Dimensions (tolerances in manufacturing)
  - Material properties (*e.g.* variability of the stiffness or resistance)

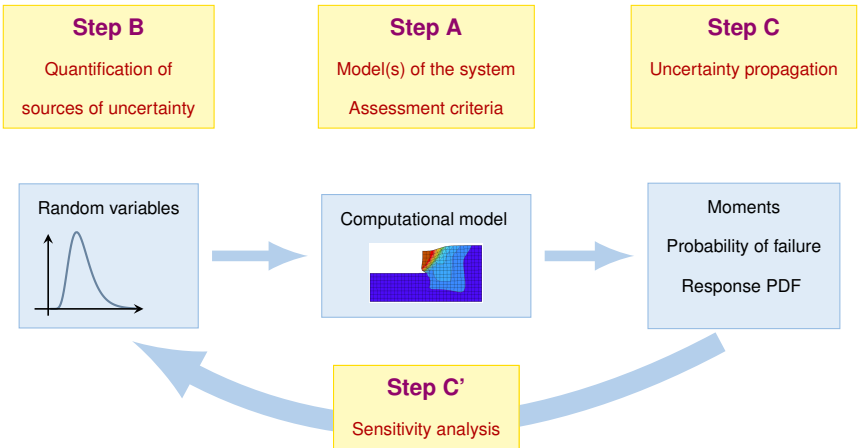
- Unforecast exposures: exceptional service loads, natural hazards (earthquakes, floods, landslides), climate loads (hurricanes, snow storms, etc.), accidental human actions (explosions, fire, etc.)






# Outline

# Global framework for uncertainty quantification




B. Sudret, Uncertainty propagation and sensitivity analysis in mechanical models – contributions to structural reliability and stochastic spectral methods (2007)

## Surrogate models for uncertainty quantification

A surrogate model $\tilde{\mathcal{M}}$ is an approximation of the original computational model $\mathcal{M}$ with the following features:

- It assumes some regularity of the model $\mathcal{M}$ and some general functional shape
- It is built from a limited set of runs of the original model $\mathcal{M}$ called the experimental design $\mathcal{X} = \left\{\boldsymbol{x}^{(i)},\, i = 1,\, \ldots\, , n\right\}$

Simulated data

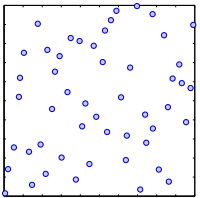
- It is fast to evaluate!

# Surrogate models for uncertainty quantification

| Name | Shape | Parameters |
|---|---|---|
| Polynomial chaos expansions | $\tilde{\mathcal{M}}(\boldsymbol{x}) = \sum_{\boldsymbol{\alpha} \in \mathcal{A}} a_{\boldsymbol{\alpha}} \, \Psi_{\boldsymbol{\alpha}}(\boldsymbol{x})$ | $\boldsymbol{a_\alpha}$ |
| Low-rank tensor approximations | $\tilde{\mathcal{M}}(\boldsymbol{x}) = \sum_{l=1}^{R} b_l \left( \prod_{i=1}^{M} v_l^{(i)}(x_i) \right)$ | $b_l, \, z_{k,l}^{(i)}$ |
| Kriging (a.k.a Gaussian processes) | $\tilde{\mathcal{M}}(\boldsymbol{x}) = \boldsymbol{\beta}^{\mathsf{T}} \cdot \boldsymbol{f}(\boldsymbol{x}) + Z(\boldsymbol{x}, \omega)$ | $\boldsymbol{\beta}$ , $\sigma_Z^2$ , $\boldsymbol{\theta}$ |
| Support vector machines | $\tilde{\mathcal{M}}(\boldsymbol{x}) = \sum_{i=1}^{m} a_i \, K(\boldsymbol{x}_i, \boldsymbol{x}) + b$ | $\boldsymbol{a}$ , $b$ |
| (Deep) Neural networks | $\tilde{\mathcal{M}}(\boldsymbol{x}) = f_n \left( \cdots f_2 \left( b_2 + f_1 \left( b_1 + \boldsymbol{w}_1 \cdot \boldsymbol{x} \right) \cdot \boldsymbol{w}_2 \right) \right)$ | $\boldsymbol{w}, \boldsymbol{b}$ |

# Ingredients for building a surrogate model

- Select an experimental design $\mathcal{X}$ that covers at best the domain of input parameters:
  - (Monte Carlo simulation)
  - Latin hypercube sampling (LHS)
  - Low-discrepancy sequences



- Run the computational model $\mathcal{M}$ onto $\mathcal{X}$ exactly as in Monte Carlo simulation

## Ingredients for building a surrogate model

- Smartly post-process the data $\{\mathcal{X}, \mathcal{M}(\mathcal{X})\}$ through a learning algorithm

| Name | Learning method |
|---|---|
| Polynomial chaos expansions | sparse grid integration, least-squares, compressive sensing |
| Low-rank tensor approximations | alternate least squares |
| Kriging | maximum likelihood, Bayesian inference |
| Support vector machines | quadratic programming |

- Validate the surrogate model, *e.g.* estimate a global error $\varepsilon = \mathbb{E}\left[\left(\mathcal{M}(\boldsymbol{X}) - \tilde{\mathcal{M}}(\boldsymbol{X})\right)^2\right]$

# Advantages of surrogate models

## Usage

$$\mathcal{M}(\boldsymbol{x}) \quad \approx \quad \tilde{\mathcal{M}}(\boldsymbol{x})$$

hours per run — seconds for $10^6$ runs

## Advantages

- Non-intrusive methods: based on runs of the computational model, exactly as in Monte Carlo simulation
- Suited to high performance computing: "embarrassingly parallel"

## Challenges

- Need for rigorous validation
- Communication: advanced mathematical background

## Efficiency

- 6-8 orders of magnitude (!) less CPU for a single run
- 2-3 orders of magnitude less runs compared to a full Monte Carlo simulation

## Surrogate modelling vs. machine learning

| Features | Supervised learning | Surrogate modelling |
|---|---|---|
| Computational model $\mathcal{M}$ | ✗ | ✔ |
| Probabilistic model of the input $\boldsymbol{X} \sim f_{\boldsymbol{X}}$ | ✗ | ✔ |
| Training data: $\mathcal{X} = \{(\boldsymbol{x}_i, y_i),\, i = 1, \ldots, n\}$ | ✔ Training data set (big data) | ✔ Experimental design (small data) |
| Prediction goal: for a new $\boldsymbol{x} \notin \mathcal{X}$, $y(\boldsymbol{x})$ ? | $\sum_{i=1}^{m} y_i\, K(\boldsymbol{x}_i, \boldsymbol{x}) + b$ | $\sum_{\boldsymbol{\alpha} \in \mathcal{A}} y_{\boldsymbol{\alpha}}\, \Psi_{\boldsymbol{\alpha}}(\boldsymbol{x})$ |
| Validation (resp. cross-validation) | ✔ Validation set | ✔ Leave-one-out CV |

# Outline

# Polynomial chaos expansions in a nutshell

Ghanem & Spanos (1991; 2003); Xiu & Karniadakis (2002); Soize & Ghanem (2004)

- We assume here for simplicity that the input parameters are independent with $X_i \sim f_{X_i}$, $i = 1, \ldots, M$
- PCE is also applicable in the general case using an isoprobabilistic transform $\boldsymbol{X} \mapsto \boldsymbol{\Xi}$

The polynomial chaos expansion of the (random) model response reads:

$$Y = \sum_{\boldsymbol{\alpha} \in \mathbb{N}^M} y_{\boldsymbol{\alpha}} \, \Psi_{\boldsymbol{\alpha}}(\boldsymbol{X})$$

where:

- $\Psi_{\boldsymbol{\alpha}}(\boldsymbol{X})$ are basis functions (multivariate orthonormal polynomials)
- $y_{\boldsymbol{\alpha}}$ are coefficients to be computed (coordinates)

# Multivariate polynomial basis

## Univariate polynomials

- For each input variable $X_i$, univariate orthogonal polynomials $\{P_k^{(i)},\ k \in \mathbb{N}\}$ are built:

$$\left\langle P_j^{(i)}, P_k^{(i)} \right\rangle = \int P_j^{(i)}(u)\ P_k^{(i)}(u)\ f_{X_i}(u)\, du = \ \gamma_j^{(i)}\ \delta_{jk}$$

*e.g.*, Legendre polynomials if $X_i \sim \mathcal{U}(-1, 1)$, Hermite polynomials if $X_i \sim \mathcal{N}(0, 1)$

- Normalization: $\Psi_j^{(i)} = P_j^{(i)} / \sqrt{\gamma_j^{(i)}} \quad i = 1, \ldots, M, \quad j \in \mathbb{N}$

## Tensor product construction

$$\Psi_{\boldsymbol{\alpha}}(\boldsymbol{x}) \overset{\text{def}}{=} \prod_{i=1}^{M} \Psi_{\alpha_i}^{(i)}(x_i) \qquad \mathbb{E}\left[\Psi_{\boldsymbol{\alpha}}(\boldsymbol{X})\Psi_{\boldsymbol{\beta}}(\boldsymbol{X})\right] = \delta_{\boldsymbol{\alpha}\boldsymbol{\beta}}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_M)$ are multi-indices (partial degree in each dimension)

# Computing the coefficients by least-square minimization

Isukapalli (1999); Berveiller, Sudret & Lemaire (2006)

## Principle

The exact (infinite) series expansion is considered as the sum of a truncated series and a residual:

$$Y = \mathcal{M}(\boldsymbol{X}) = \sum_{\boldsymbol{\alpha} \in \mathcal{A}} y_{\boldsymbol{\alpha}} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{X}) + \varepsilon_P \equiv \mathbf{Y}^{\mathsf{T}} \boldsymbol{\Psi}(\boldsymbol{X}) + \varepsilon_P(\boldsymbol{X})$$

where : $\mathbf{Y} = \{y_{\boldsymbol{\alpha}}, \boldsymbol{\alpha} \in \mathcal{A}\} \equiv \{y_0, \ldots, y_{P-1}\}$ ($P$ unknown coefficients)

$$\boldsymbol{\Psi}(\boldsymbol{x}) = \{\Psi_0(\boldsymbol{x}), \ldots, \Psi_{P-1}(\boldsymbol{x})\}$$

## Least-square minimization

The unknown coefficients are estimated by minimizing the mean square residual error:

$$\hat{\mathbf{Y}} = \arg\min \mathbb{E}\left[\left(\mathbf{Y}^{\mathsf{T}} \boldsymbol{\Psi}(\boldsymbol{X}) - \mathcal{M}(\boldsymbol{X})\right)^2\right]$$

## Discrete (ordinary) least-square minimization

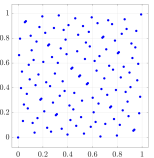
An estimate of the mean square error (sample average) is minimized:

$$\hat{\mathbf{Y}} = \arg\min_{\mathbf{Y}\in\mathbb{R}^P} \frac{1}{n}\sum_{i=1}^{n} \left(\mathbf{Y}^\mathsf{T}\boldsymbol{\Psi}(\boldsymbol{x}^{(i)}) - \mathcal{M}(\boldsymbol{x}^{(i)})\right)^2$$

### Procedure

- Select a truncation scheme, *e.g.* $\mathcal{A}^{M,p} = \left\{\boldsymbol{\alpha}\in\mathbb{N}^M \ : \ |\boldsymbol{\alpha}|_1 \leq p\right\}$
- Select an experimental design and evaluate the model response

$$\mathbf{M} = \left\{\mathcal{M}(\boldsymbol{x}^{(1)}), \ldots, \mathcal{M}(\boldsymbol{x}^{(n)})\right\}^\mathsf{T}$$




- Compute the experimental matrix

$$\mathbf{A}_{ij} = \Psi_j\left(\boldsymbol{x}^{(i)}\right) \quad i = 1, \ldots, n \ ; \ j = 0, \ldots, P-1$$

- Solve the resulting linear system

$$\hat{\mathbf{Y}} = (\mathbf{A}^\mathsf{T}\mathbf{A})^{-1}\mathbf{A}^\mathsf{T}\mathbf{M}$$

## Error estimators

- In least-squares analysis, the generalization error is defined as:

$$E_{gen} = \mathbb{E}\left[\left(\mathcal{M}(\boldsymbol{X}) - \mathcal{M}^{\mathsf{PC}}(\boldsymbol{X})\right)^2\right] \qquad \mathcal{M}^{\mathsf{PC}}(\boldsymbol{X}) = \sum_{\boldsymbol{\alpha}\in\mathcal{A}} y_{\boldsymbol{\alpha}}\, \Psi_{\boldsymbol{\alpha}}(\boldsymbol{X})$$

- The empirical error based on the experimental design $\mathcal{X}$ is a poor estimator in case of overfitting

$$E_{emp} = \frac{1}{n}\sum_{i=1}^{n}\left(\mathcal{M}(\boldsymbol{x}^{(i)}) - \mathcal{M}^{\mathsf{PC}}(\boldsymbol{x}^{(i)})\right)^2$$

### Leave-one-out cross validation

- From statistical learning theory, model validation shall be carried out using independent data

$$E_{LOO} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\mathcal{M}(\boldsymbol{x}^{(i)}) - \mathcal{M}^{PC}(\boldsymbol{x}^{(i)})}{1-h_i}\right)^2$$

where $h_i$ is the $i$-th diagonal term of matrix $\mathbf{A}(\mathbf{A}^{\mathsf{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathsf{T}}$

# Outline

# Post-processing PC expansions

## Statistical moments

- Due to the orthogonality of the basis functions ($\mathbb{E}\left[\Psi_{\boldsymbol{\alpha}}(\boldsymbol{X})\Psi_{\boldsymbol{\beta}}(\boldsymbol{X})\right] = \delta_{\boldsymbol{\alpha}\boldsymbol{\beta}}$) and using $\mathbb{E}\left[\Psi_{\boldsymbol{\alpha}\neq\mathbf{0}}\right] = 0$ the statistical moments read:

$$\text{Mean:} \qquad \hat{\mu}_Y = y_0$$

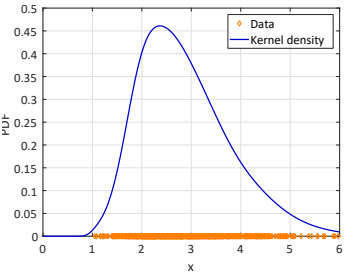
$$\text{Variance:} \qquad \hat{\sigma}^2_Y = \sum_{\boldsymbol{\alpha}\in\mathcal{A}\setminus\mathbf{0}} y^2_{\boldsymbol{\alpha}}$$

## Distribution of the QoI

- The PCE can be used as a response surface for sampling:

$$\mathfrak{y}_j = \sum_{\boldsymbol{\alpha}\in\mathcal{A}} y_{\boldsymbol{\alpha}}\, \Psi_{\boldsymbol{\alpha}}(\boldsymbol{x}_j) \quad j = 1, \ldots, n_{big}$$

- The PDF of the response is estimated by histograms or kernel smoothing



# Sensitivity analysis

## Goal

Sobol' (1993); Saltelli *et al.* (2008)

Global sensitivity analysis aims at quantifying which input parameter(s) (or combinations thereof) influence the most the response variability (variance decomposition)

## Hoeffding-Sobol' decomposition

$(\boldsymbol{X} \sim \mathcal{U}([0,1]^M))$

$$\mathcal{M}(\boldsymbol{x}) = \mathcal{M}_0 + \sum_{i=1}^{M} \mathcal{M}_i(x_i) + \sum_{1 \le i < j \le M} \mathcal{M}_{ij}(x_i, x_j) + \cdots + \mathcal{M}_{12\ldots M}(\boldsymbol{x})$$

$$= \mathcal{M}_0 + \sum_{\mathbf{u} \subset \{1, \ldots, M\}} \mathcal{M}_{\mathbf{u}}(\boldsymbol{x}_{\mathbf{u}}) \qquad (\boldsymbol{x}_{\mathbf{u}} \overset{\text{def}}{=} \{x_{i_1}, \ldots, x_{i_s}\})$$

- The summands satisfy the orthogonality condition:

$$\int_{[0,1]^M} \mathcal{M}_{\mathbf{u}}(\boldsymbol{x}_{\mathbf{u}})\, \mathcal{M}_{\mathbf{v}}(\boldsymbol{x}_{\mathbf{v}})\, d\boldsymbol{x} = 0 \qquad \forall\, \mathbf{u} \neq \mathbf{v}$$

## Sobol' indices

Total variance: $$D \equiv \mathrm{Var}\left[\mathcal{M}(\boldsymbol{X})\right] = \sum_{\mathbf{u} \subset \{1, \dots, M\}} \mathrm{Var}\left[\mathcal{M}_{\mathbf{u}}(\boldsymbol{X}_{\mathbf{u}})\right]$$

- Sobol' indices:
$$S_{\mathbf{u}} \overset{\text{def}}{=} \frac{\mathrm{Var}\left[\mathcal{M}_{\mathbf{u}}(\boldsymbol{X}_{\mathbf{u}})\right]}{D}$$
- First-order Sobol' indices:
$$S_i = \frac{D_i}{D} = \frac{\mathrm{Var}\left[\mathcal{M}_i(X_i)\right]}{D}$$
Quantify the additive effect of each input parameter separately
- Total Sobol' indices:
$$S_i^T \overset{\text{def}}{=} \sum_{\mathbf{u} \supset i} S_{\mathbf{u}}$$
Quantify the total effect of $X_i$, including interactions with the other variables.

# Link with PC expansions

## Sobol decomposition of a PC expansion

Sudret, CSM (2006); RESS (2008)

Obtained by reordering the terms of the (truncated) PC expansion $\mathcal{M}^{\text{PC}}(\boldsymbol{X}) \overset{\text{def}}{=} \sum_{\boldsymbol{\alpha} \in \mathcal{A}} y_{\boldsymbol{\alpha}} \, \Psi_{\boldsymbol{\alpha}}(\boldsymbol{X})$

## Interaction sets

For a given $\mathbf{u} \overset{\text{def}}{=} \{i_1, \ldots, i_s\}$ : $\quad \mathcal{A}_{\mathbf{u}} = \{\boldsymbol{\alpha} \in \mathcal{A} \, : \, k \in \mathbf{u} \Leftrightarrow \alpha_k \neq 0\}$

$$\mathcal{M}^{\text{PC}}(\boldsymbol{x}) = \mathcal{M}_0 + \sum_{\mathbf{u} \subset \{1, \ldots, M\}} \mathcal{M}_{\mathbf{u}}(\boldsymbol{x}_{\mathbf{u}}) \qquad \text{where} \qquad \mathcal{M}_{\mathbf{u}}(\boldsymbol{x}_{\mathbf{u}}) \overset{\text{def}}{=} \sum_{\boldsymbol{\alpha} \in \mathcal{A}_{\mathbf{u}}} y_{\boldsymbol{\alpha}} \, \Psi_{\boldsymbol{\alpha}}(\boldsymbol{x})$$

## PC-based Sobol' indices

$$S_{\mathbf{u}} = D_{\mathbf{u}}/D = \sum_{\boldsymbol{\alpha} \in \mathcal{A}_{\mathbf{u}}} y_{\boldsymbol{\alpha}}^2 / \sum_{\boldsymbol{\alpha} \in \mathcal{A} \setminus \mathbf{0}} y_{\boldsymbol{\alpha}}^2$$

The Sobol' indices are obtained analytically, at any order from the coefficients of the PC expansion

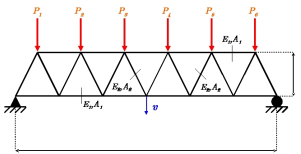
# Outline

## Curse of dimensionality

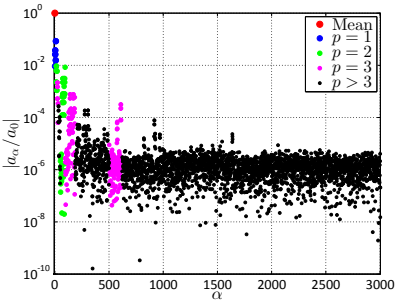
- The cardinality of the truncation scheme $\mathcal{A}^{M,p}$ is $P = \dfrac{(M+p)!}{M!\,p!}$
- Typical computational requirements: $n = OSR \cdot P$ where the oversampling rate is $OSR = 2 - 3$

However ... most coefficients are close to zero !

Example




- Elastic truss structure with $M = 10$ independent input variables
- PCE of degree $p = 5$ ($P = 3,003$ coefficients)



## Compressive sensing approaches

Blatman & Sudret (2011); Doostan & Owhadi (2011); Sargsyan *et al.* (2014); Jakeman *et al.* (2015)

- Sparsity in the solution can be induced by $\ell_1$-regularization:

$$\boldsymbol{y}_{\boldsymbol{\alpha}} = \arg\min \frac{1}{n} \sum_{i=1}^{n} \left( \mathrm{Y}^{\mathsf{T}} \boldsymbol{\Psi}(\boldsymbol{x}^{(i)}) - \mathcal{M}(\boldsymbol{x}^{(i)}) \right)^2 + \lambda \parallel \boldsymbol{y}_{\boldsymbol{\alpha}} \parallel_1$$

- Different algorithms: LASSO, orthogonal matching pursuit, LARS, Bayesian compressive sensing, subspace pursuit, etc.
- State-of-the-art-review and comparisons available in:

Lüthen, N., Marelli, S. & Sudret, B. *Sparse polynomial chaos expansions: Literature survey and benchmark*, SIAM/ASA J. Unc. Quant., 2021, 9, 593-649 https://doi.org/10.1137/20M1315774

–, *Automatic selection of basis-adaptive sparse polynomial chaos expansions for engineering applications*, Int. J. Uncertainty Quantification, 2022, 12, 49-74 https://doi.org/10.1615/Int.J.UncertaintyQuantification.2021036153

# Outline

# Sparse regression solvers (1/4)

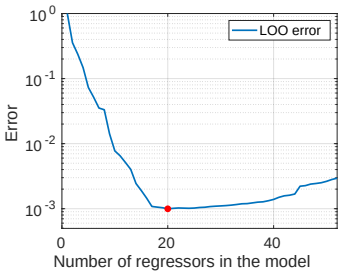
## Orthogonal matching pursuit (OMP)

- Also known as forward stepwise regression
- Greedy technique to solve

$$\min_{\boldsymbol{c}} \| \boldsymbol{\Psi c} - \boldsymbol{y} \|_2 + \lambda \| \boldsymbol{c} \|_0$$

with $\| \boldsymbol{c} \|_0 := \sum_i \mathbf{1}_{\{c_i \neq 0\}}$

- Relevant regressors are added one-by-one depending on their correlation with the residual
- Coefficients are computed by OLS
- Best solution selected by leave-one-out cross-validation



# Sparse regression solvers (2/4)

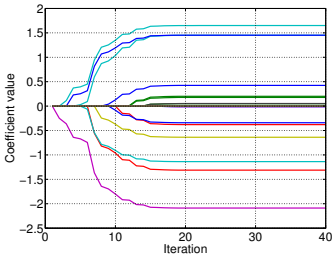
## Least Angle Regression (LARS)

Blatman & Sudret, J. Comput. Phys. (2011)

- Approximates the solution to LASSO

$$\min_{\boldsymbol{c}} \| \boldsymbol{\Psi c} - \boldsymbol{y} \|_2 \text{ s.t. } \| \boldsymbol{c} \|_1 \leq \tau$$

- Generates a path of solutions for increasing $\tau$ by adding relevant regressors one-by-one
- Best solution selected by leave-one-out cross-validation
- [Hybrid]: Final coefficients computed by OLS



# Sparse regression solvers (3/4)

## Subspace pursuit (SP)

Dai & Milenkovic, IEEE Transactions on Inform. Theory (2009)

- Solves

$$\min_{\boldsymbol{c}} \| \boldsymbol{\Psi c} - \boldsymbol{y} \|_2 \text{ s.t. } \| \boldsymbol{c} \|_0 = K$$

- Enlarges and shrinks the set of active basis functions iteratively and greedily
- Maintains at all times a basis of size $K$
- Based on OLS and correlation with residual
- Various values of $K$ are tested: best model chosen by LOO cross-validation

# Sparse regression solvers (4/4)

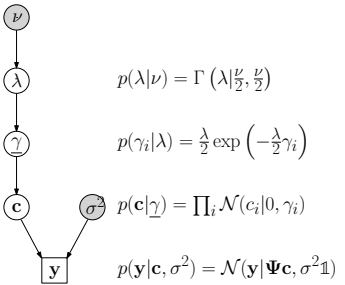
## Bayesian Compressive Sensing (BCS)

Babacan, Molina and Katsaggelos, IEEE Transactions on Image Processing (2010)

- Bayesian framework to estimate the PCE coefficients:

  Maximize posterior $p(\boldsymbol{c}|\boldsymbol{y})$

- Gaussian prior on $\boldsymbol{c}$, Laplace prior on $\lambda$
- Choose the distribution of $\lambda$ so that $p(\boldsymbol{c})$ is encouraging sparsity



# Basis adaptivity schemes

## Principle

- Compute several PCEs, each associated to a different basis size
- Choose the PCE associated to the lowest leave-one-out error

## Degree (p)- and q-norm adaptivity

Blatman & Sudret, J. Comput. Phys. (2011)

- Try a range of $p$'s and $q$'s

$$\mathcal{A}^{p,q} = \{\boldsymbol{\alpha} \in \mathbb{N}^d : \| \boldsymbol{\alpha} \|_q \leq p\} \quad \text{where} \quad \| \boldsymbol{\alpha} \|_q = \left( \sum_{i=1}^{d} |\alpha_i|^q \right)^{1/q}$$

# Sparse PCE: wrap up

**Algorithm 1:** Adaptive Sparse Polynomial Chaos Expansions

**Input:** Computational budget $n$
**Initialization**
  Sample experimental design $\mathcal{X} = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}\}$
  Evaluate model response $\mathcal{Y} = \{\mathcal{M}(\boldsymbol{x}^{(1)}), \ldots, \mathcal{M}(\boldsymbol{x}^{(n)})\}$
**PCE construction**
  **for** $p = p_{\min} : p_{\max}$ **do**
    **for** $q \in \mathcal{Q}$ **do**
      Select candidate basis $\mathcal{A}_q^{M,p}$
      Run LAR/OMP/SP/BCS for extracting the optimal sparse basis $\mathcal{A}^*(p,q)$
      Compute coefficients $\{y_{\boldsymbol{\alpha}},\ \boldsymbol{\alpha} \in \mathcal{A}^*(p,q)\}$ by OLS
      Compute $e_{\text{LOO}}(p,q)$
    **end**
  **end**
  $(p^*, q^*) = \arg\min e_{\text{LOO}}(p,q)$
**Return** Optimal sparse basis $\mathcal{A}^*(p,q)$, PCE coefficients, $e_{\text{LOO}}(p^*,q^*)$

# Outline

# Candidate bases and error metrics

## PCE truncation

- Degree adaptivity: 2:15 (with early stop)
- $q$-norm adaptivity: 0.4:0.1:1
- Max. interaction $r = 5$ in small dimension (resp. $r = 2$ in large dimension)

## Relative mean-square error on a validation set

$$\text{rMSE} = \frac{\sum_{i=1}^{n} \left(\mathcal{M}(\boldsymbol{x}_i) - \tilde{\mathcal{M}}(\boldsymbol{x}_i)\right)^2}{\sum_{i=1}^{n} \left(\mathcal{M}(\boldsymbol{x}_i) - \mu_{\mathcal{X}_{\text{val}}}\right)^2}$$

# First benchmark: Small dimensionality

**Functions available at** https://uqworld.org/c/uq-resources/benchmarks/

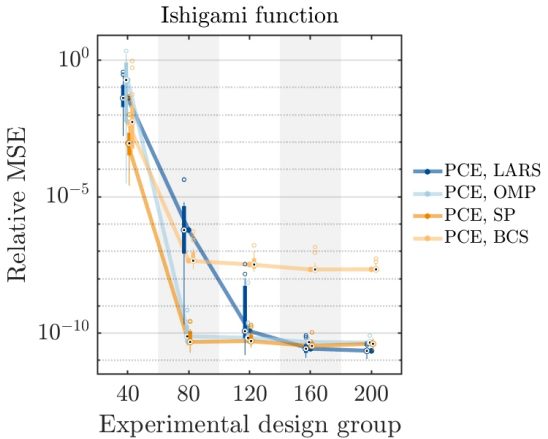
## Case studies

| No. | Bench case | Dim. | Exp. design size | | | | |
|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 |
| 1 | Ishigami function | 3 | | | | | |
| 2 | Undamped oscillator (Gayton) | 6 | 40 | 80 | 120 | 160 | 200 |
| 3 | Borehole function | 8 | | | | | |
| 4 | Wing weight function | 10 | | | | | |
| 5 | Truss model | 10 | 100 | 200 | 300 | 400 | 500 |
| 6 | Damped oscillator (Dubourg) | 8 | 400 | 800 | 1200 | 1600 | 2000 |

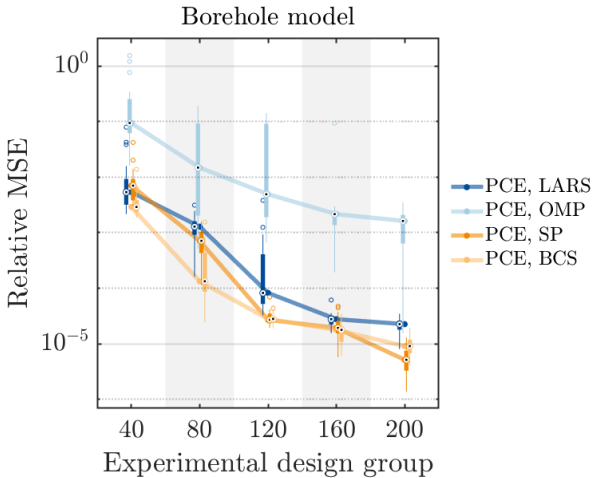
## Settings

- LHS designs (20 replications per ED size)
- Validation with $10^5$ Monte Carlo samples

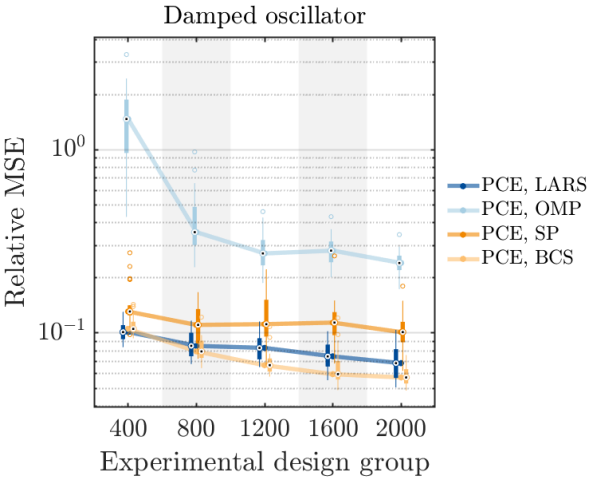
## Convergence curves



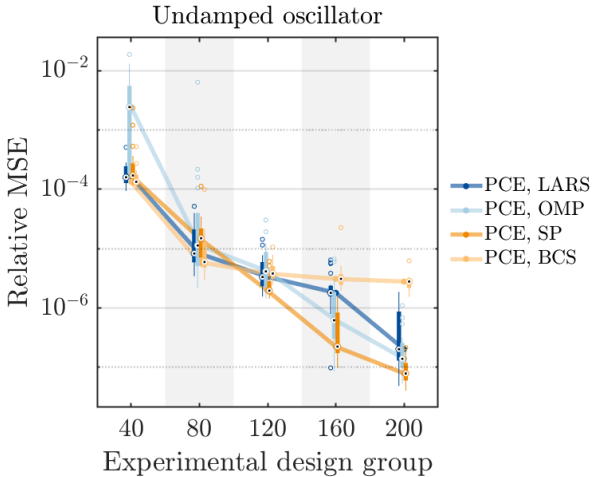
## Convergence curves



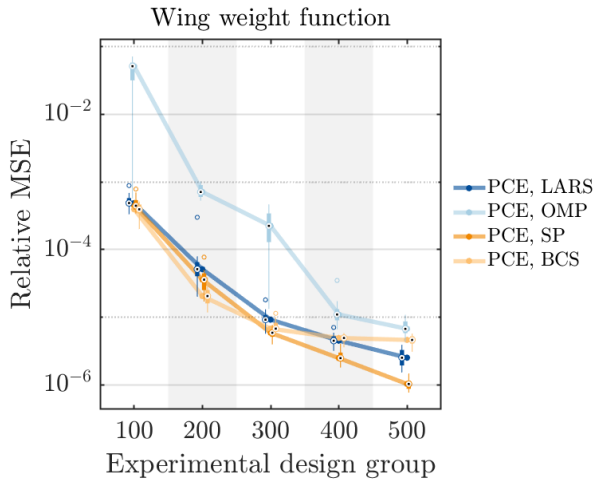
## Convergence curves



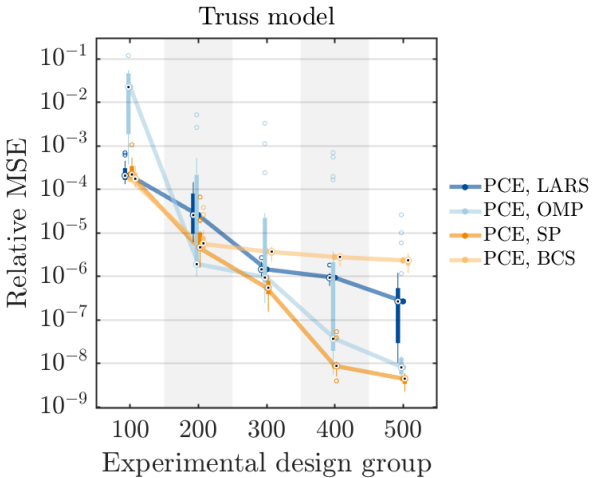
## Convergence curves



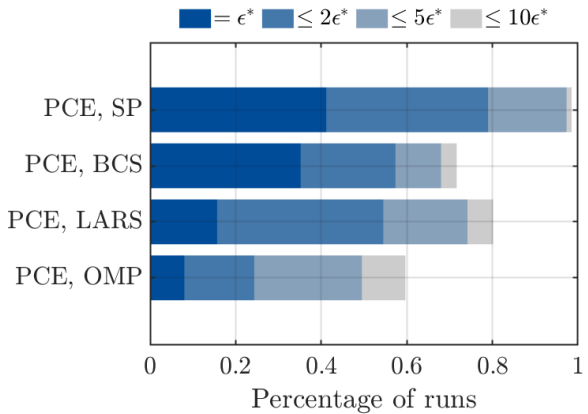
## Convergence curves



## Convergence curves

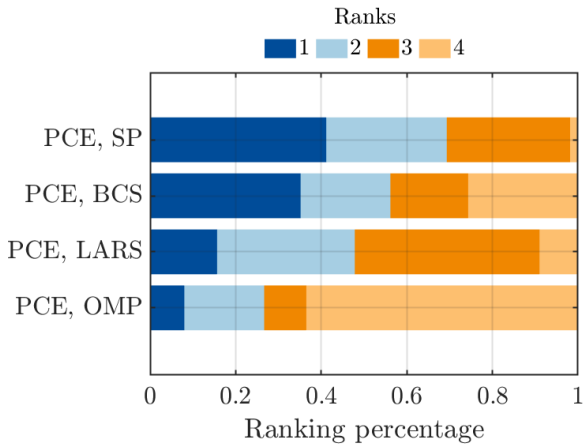


## Aggregated results




- Comparison of errors on 600 analyses with 4 solvers
- Ranking according to #runs with $\epsilon \leq 2\epsilon^*$

## Aggregated results




- Comparison of errors on 600 analyses with 4 solvers
- Ranking according to #runs when a solver gives the lowest error

# Second benchmark: Large dimensionality

**Functions available at** https://uqworld.org/c/uq-resources/benchmarks/

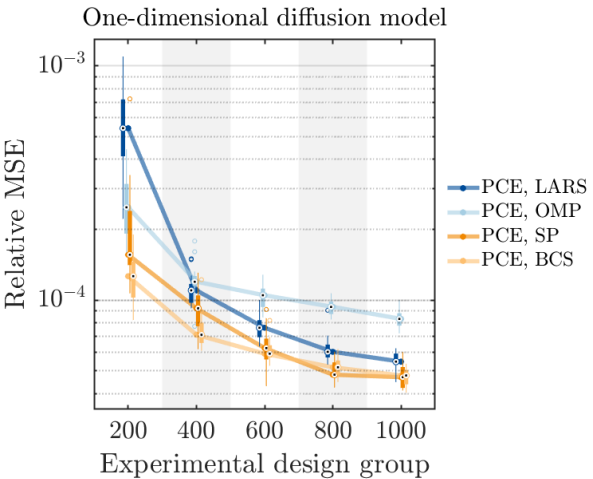
## Case studies

| No. | Bench case | Dim. | Exp. design size | | | | |
|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 |
| 1 | One-dim. diffusion model | 62 | 200 | 400 | 600 | 800 | 1000 |
| 2 | 100D function | 100 | 400 | 800 | 1200 | 1600 | 2000 |
| 3 | Two-dim. heat diffusion model | 53 | | | | | |

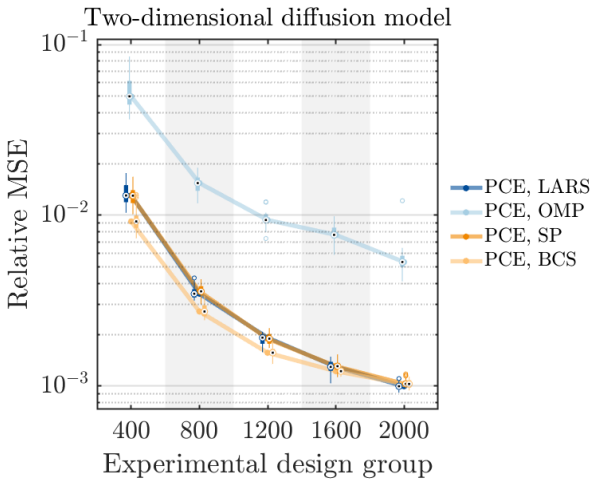
## Settings

- LHS designs (20 replications per ED size)
- Validation with $10^5$ Monte Carlo samples

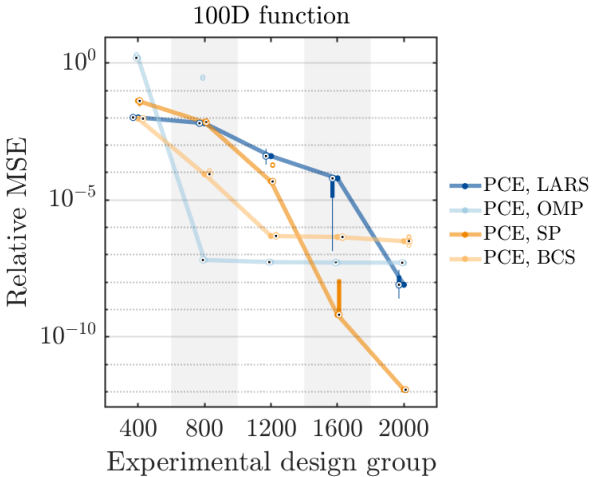
## Convergence curves



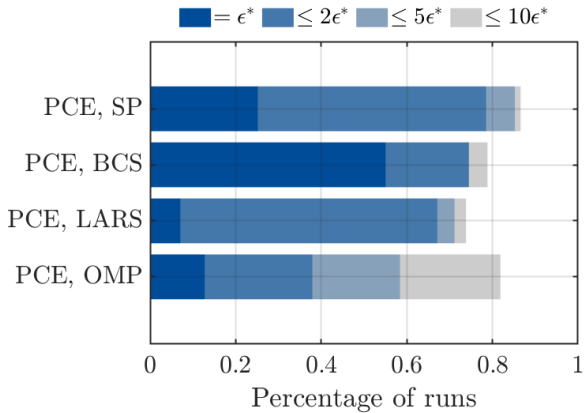
## Convergence curves



## Convergence curves

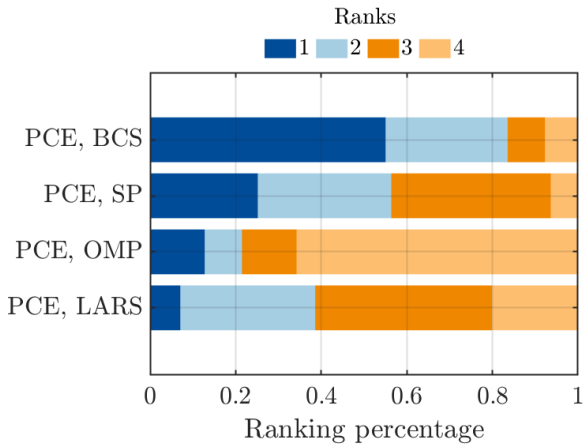


## Aggregated results




- Comparison of errors on 300 analyses with 4 solvers
- Ranking according to #runs with $\epsilon \leq 2\epsilon^*$

## Aggregated results




- Comparison of errors on 300 analyses with 4 solvers
- Ranking according to #runs when a solver gives the lowest error

# Outline

# Models with time-dependent outputs

## Problem statement

- Consider a computational model of a dynamical system:

$$\mathcal{D}_{\boldsymbol{\Xi}} \times [0,T]: \ (\boldsymbol{\xi}, t) \mapsto \mathcal{M}(\boldsymbol{\xi}, t)$$

  where $\boldsymbol{\Xi}$ is a random vector of uncertain parameters with given PDF $f_{\boldsymbol{\Xi}}$

- Uncertainties may be in:
  - The excitation, denoted by $x(\boldsymbol{\xi}_x, t)$
  - And/or in the system's characteristics ($\boldsymbol{\xi}_s$):

  i.e.:

$$\mathcal{M}(\boldsymbol{\xi}, t) \equiv \mathcal{M}(x(\boldsymbol{\xi}_x, t), \ \boldsymbol{\xi}_s)$$

Point-in-time PCE does not work!

# Example: Duffing oscillator

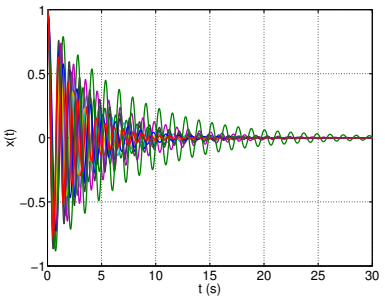
Non-linear SDOF Duffing oscillator:

$$\ddot{x}(t) + 2\,\omega\,\zeta\,\dot{x}(t) + \omega^2\,\left(x(t) + \varepsilon\,x^3(t)\right) = 0$$

Initial conditions: $x(0) = 1, \quad \dot{x}(0) = 0$

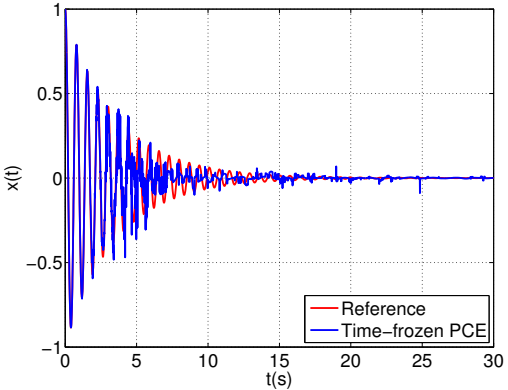
Input: 3 uniform random variables

| RV | Distribution | Values |
|---|---|---|
| $\zeta$ | Uniform | $\mathcal{U}[0.015,\, 0.045]$ |
| $\omega$ | Uniform | $\mathcal{U}[\pi,\, 3\,\pi]$ |
| $\varepsilon$ | Uniform | $\mathcal{U}[-0.25,\, -0.75]$ |

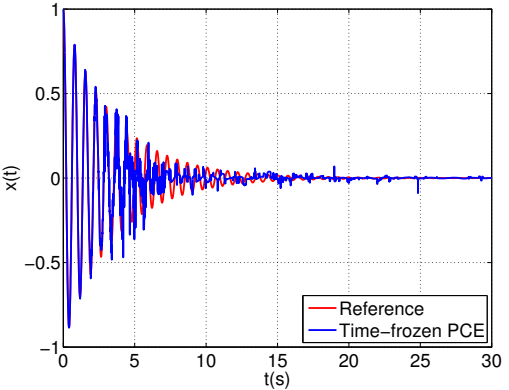



**Samples of trajectories**

# Time-frozen PCE



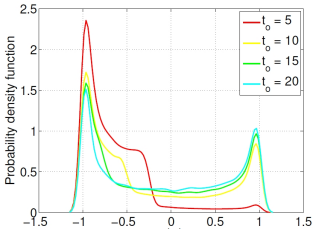

$(\zeta, \omega, \varepsilon) = (0.03, 8.92, -0.34)$

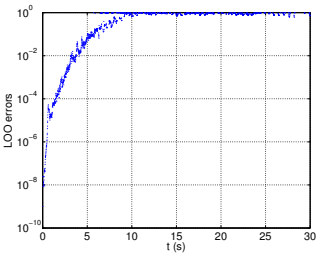



$(\zeta, \omega, \varepsilon) = (0.04, 3.18, -0.33)$**)**

## Why time-frozen PCE does not work?

- The map $\boldsymbol{\xi} \mapsto \mathcal{M}(\boldsymbol{\xi}, t)$ becomes increasingly non linear with time
- The time-frozen distribution of the output at time $t_0$ becomes more complex (*e.g.* multimodal)
- Expansions of higher degree would be required to keep sufficient accuracy with time
- For a fixed experimental design, the LOO error blows up



# Stochastic time warping

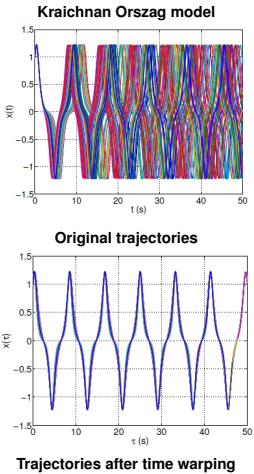
## Problem

Mai & Sudret, SIAM J. Unc. Quant. (2017)

The various trajectories are “similar” yet not in phase, thus the complex point-in-time response

## Principles of the method

- A specific warped time scale $\tau$ is introduced for each trajectory so that they become “in phase”
- Point-in-time PCE is carried out in the warped time scale using reduced-order modelling (principal component analysis)
- Predictions are carried out in the warped time scale and back-transformed in the real time line



# Example: Oregonator model

The Oregonator model represents a well-stirred, homogeneous chemical system governed by a three species coupled mechanism

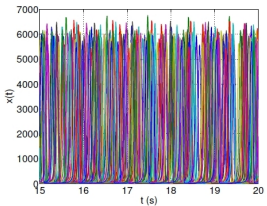
## Governing equations

$$\dot{x}(t) = k_1\, y(t) - k_2\, x(t)\, y(t) + k_3\, x(t) - k_4\, x(t)^2$$
$$\dot{y}(t) = -k_1\, y(t) - k_2\, x(t)\, y(t) + k_5\, z(t)$$
$$\dot{z}(t) = k_3\, x(t) - k_5\, z(t)$$

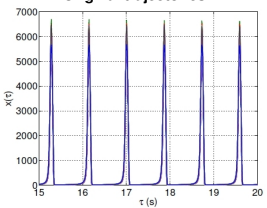
## Input reaction parameters

| Parameter | Distribution | Values |
|---|---|---|
| $k_1$ | Uniform | $\mathcal{U}[1.8,\ 2.2]$ |
| $k_2$ | Uniform | $\mathcal{U}[0.095,\ 0.1005]$ |
| $k_3$ | Gaussian | $\mathcal{N}(104, 1.04)$ |
| $k_4$ | Uniform | $\mathcal{U}[0.0076, 0.0084]$ |
| $k_5$ | Uniform | $\mathcal{U}[23.4, 28.6]$ |

Le Maître et al. (2010)




**Original trajectories**

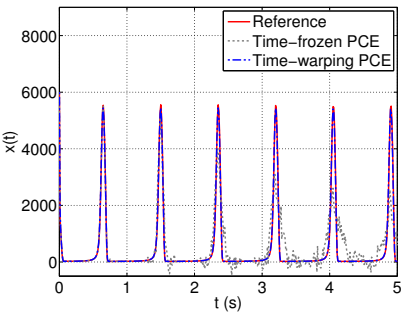



**Trajectories after time warping**

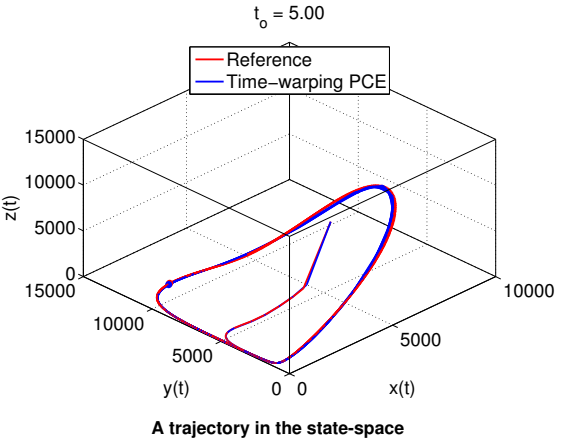
# Oregonator model: prediction

## Surrogate model

- Experimental design of size $n = 50$
- Validation set of size $n_{val} = 10,000$

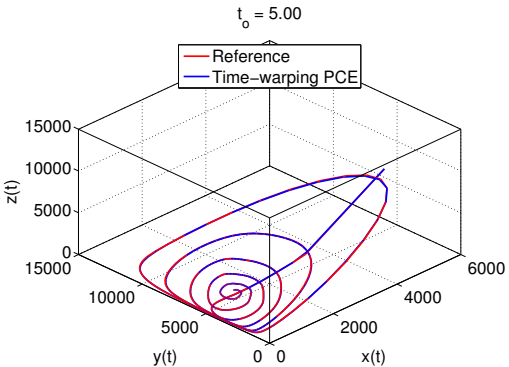



**A specific trajectory** ($\varepsilon = 0.0294$)

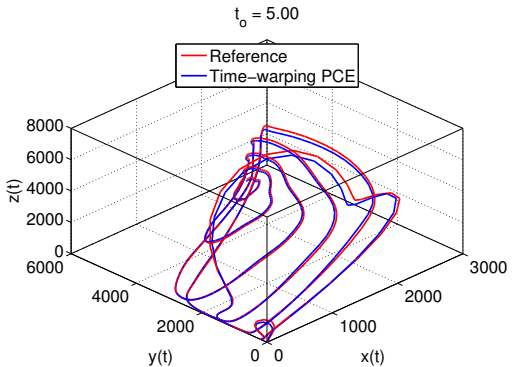



**A trajectory in the state-space**

# Oregonator model: mean and std trajectories




**Mean trajectory ($\varepsilon \approx 10^{-4}$)**




**Standard deviation ($\varepsilon \approx 10^{-3}$)**

# Bouc-Wen nonlinear oscillator

## Governing equations

$$\ddot{y}(t) + 2\,\zeta\,\omega\,\dot{y}(t) + \omega^2(\rho\,y(t) + (1-\rho)\,z(t)) = -x(t)$$

$$\dot{z}(t) = \gamma\dot{y}(t) - \alpha\;|\dot{y}(t)|\;|z(t)|^{n-1}\,z(t) - \beta\,\dot{y}(t)\;|z(t)|^n$$

$$x(t) = A\,\sin(\omega_x\,t)$$

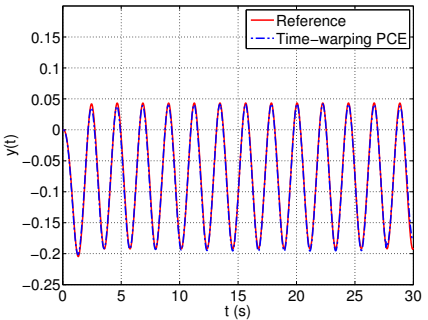
## Input parameters

| Parameter | Distribution | Mean | Standard deviation | COV |
|---|---|---|---|---|
| $\zeta$ | Uniform | 0.02 | 0.002 | 0.1 |
| $\omega$ | Uniform | $2\,\pi$ | $0.2\,\pi$ | 0.1 |
| $\alpha$ | Uniform | $50$ | $5$ | 0.1 |
| $A$ | Uniform | $1$ | $0.1$ | 0.1 |
| $\omega_x$ | Uniform | $\pi$ | $0.1\,\pi$ | 0.1 |

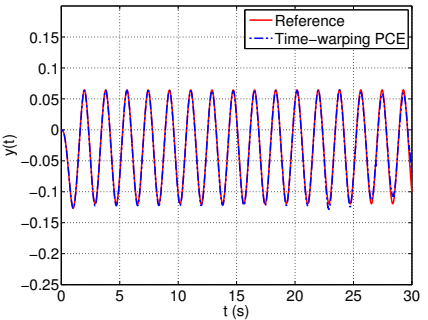
# Bouc-Wen model: two particular predictions

## Surrogate model

- Experimental design of size $n = 100$
- Validation set of size $n_{val} = 10,000$

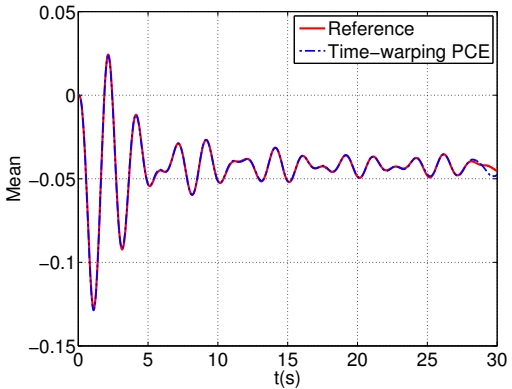



**Trajectory #1** ($\varepsilon = 3.1 \cdot 10^{-3}$)

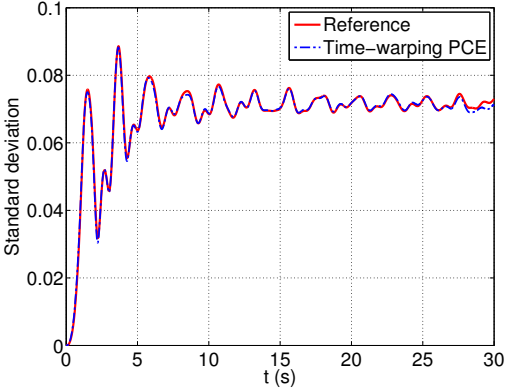



**Trajectory #2** ($\varepsilon = 3.9 \cdot 10^{-3}$)

# Bouc-Wen model: statistical moments

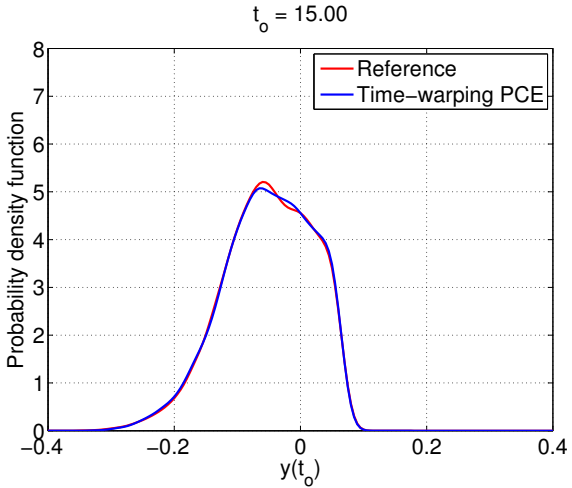



**Mean trajectory** ($\varepsilon = 2.4 \cdot 10^{-3}$)




**Standard deviation** ($\varepsilon = 2.4 \cdot 10^{-3}$)

## Bouc-Wen model: evolution of the PDF



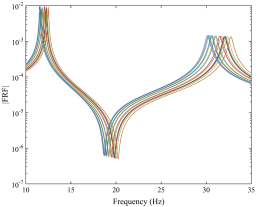

Time-warping PCEs capture not only the mean and standard deviation but also the entire PDF

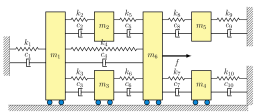
# Dynamics in the frequency domain

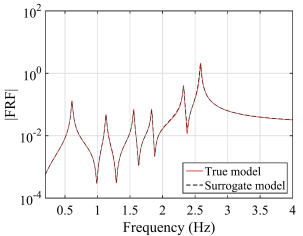
## Premise

Vaghoubi, Marelli & Sudret, Prob. Eng. Mech. (2017)

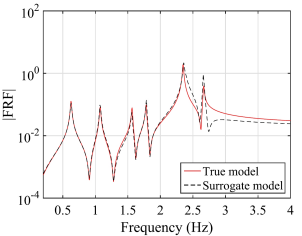
- Frequency response functions (FRF) allow one to compute the response to harmonic excitation
- In case of uncertain system properties (masses, stiffness coefficients) the resonance frequencies are shifted






(a) Typical FRF prediction




(b) Worst FRF prediction

# Nonlinear transient models: PC-NARX

## Goal

Mai, Spiridonakos, Chatzi & Sudret, Int. J. Uncer. Quant. (2016)

Address uncertainty quantification problems for earthquake engineering, which involves transient, strongly non-linear mechanical models

## PC-NARX

- Use of non linear autoregressive with exogenous input models (NARX) to capture the dynamics:

$$y(t) = \mathcal{F}\left(x(t), \ldots, x(t-n_x), y(t-1), \ldots, y(t-n_y)\right) + \epsilon_t \equiv \mathcal{F}\left(\boldsymbol{z}(t)\right) + \epsilon_t$$

- Expand the NARX coefficients of different random trajectories onto a PCE basis

$$y(t, \boldsymbol{\xi}) = \sum_{i=1}^{n_g} \sum_{\boldsymbol{\alpha} \in \mathcal{A}_i} \vartheta_{i,\boldsymbol{\alpha}}\, \psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi})\, g_i(\boldsymbol{z}(t)) + \epsilon(t, \boldsymbol{\xi})$$

# Bouc-Wen model

## Governing equations

Kafali & Grigoriu (2007), Spiridonakos & Chatzi (2015)

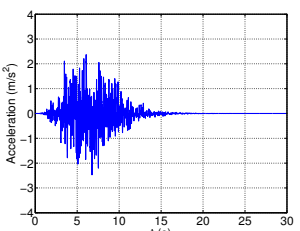
$$\ddot{y}(t) + 2\,\zeta\,\omega\,\dot{y}(t) + \omega^2(\rho\,y(t) + (1-\rho)\,z(t)) = -x(t),$$
$$\dot{z}(t) = \gamma\dot{y}(t) - \alpha\;|\dot{y}(t)|\;|z(t)|^{n-1}\,z(t) - \beta\,\dot{y}(t)\;|z(t)|^{n}\,,$$

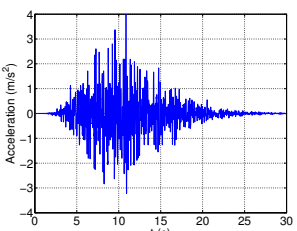
## Excitation

$x(t)$ is generated by a probabilistic ground motion model

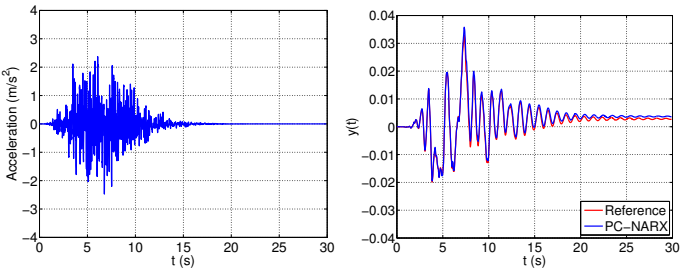
Rezaeian & Der Kiureghian (2010)

$$x(t) = q(t,\boldsymbol{\alpha}) \sum_{i=1}^{n} s_i\,(t, \boldsymbol{\lambda}(t_i))\;U_i$$

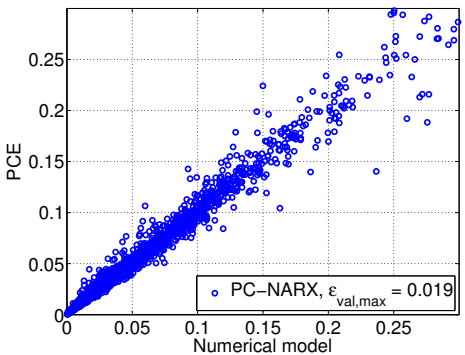


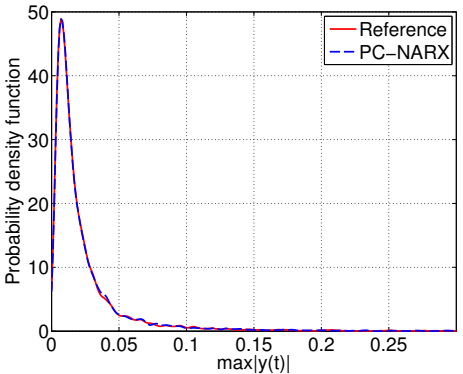
# Bouc-Wen model: prediction



# Bouc-Wen model: prediction



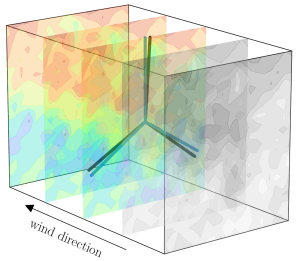

Maximal displacements



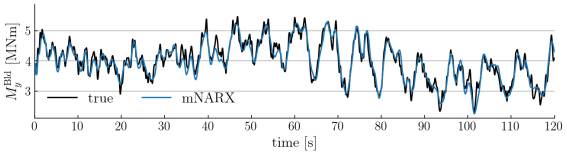

PDF of maximal displacements

# Wind turbine simulations: mNARX surrogate

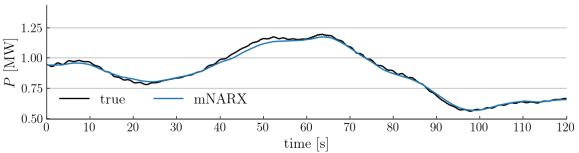
## Movie-to-time series surrogate




**Blade flapwise bending moment**

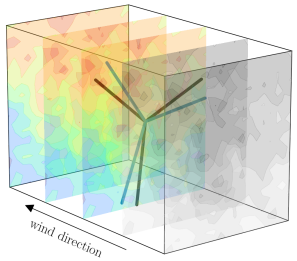



**Generated power**

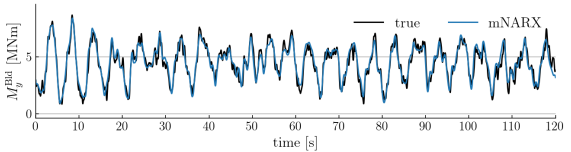


# Wind turbine simulations: mNARX surrogate

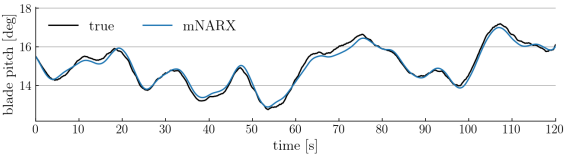
## Movie-to-time series surrogate




**Blade flapwise bending moment**




**Blade pitch**



## Conclusions

- Surrogate models are unavoidable for solving uncertainty quantification problems involving costly computational models (*e.g.* finite element models)
- Sparse PCE and its extensions (time warping, PC-NARX, PC-Kriging, DRSM, etc.) allow us to address a wide range of engineering problems
- Techniques for constructing surrogates are versatile, general-purpose and field-independent
- All the presented algorithms are available in the general-purpose uncertainty quantification software UQLab

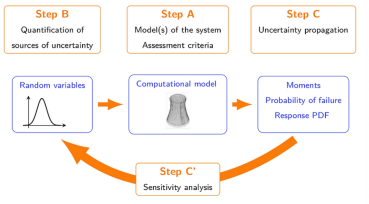
# UQLab

## The Framework for Uncertainty Quantification


OVERVIEW FEATURES DOCUMENTATION DOWNLOAD/INSTALL ABOUT COMMUNITY

**"Make uncertainty quantification available for anybody, in any field of applied science and engineering"**

**www.uqlab.com**




- MATLAB®-based Uncertainty Quantification framework
- State-of-the art, highly optimized open source algorithms
- Fast learning curve for beginners
- Modular structure, easy to extend
- Exhaustive documentation

# UQLab: The Uncertainty Quantification Software


- BSD 3-Clause license:

  **Free access to academic, industrial, governmental and non-governmental users**

- ∼7,800+ registered users from 94 countries since 2015 (1078 so far in 2024)

`http://www.uqlab.com`


UQ[py]Lab

- The cloud version of UQLab, accessible via an API (SaaS)
- Available with python bindings for beta testing

`https://uqpylab.uq-cloud.io/`

| Country | # Users |
|---|---|
| China | 1336 |
| United States | 1022 |
| Germany | 571 |
| France | 552 |
| Switzerland | 458 |
| United Kingdom | 295 |
| India | 289 |
| Brazil | 260 |
| Italy | 261 |
| Canada | 140 |
| Belgium | 127 |
| The Netherlands | 119 |

As of October 7, 2024

# Questions ?

**Chair of Risk, Safety & Uncertainty Quantification**

www.rsuq.ethz.ch

**Thank you very much for your attention !**

**The Uncertainty Quantification Software**

www.uqlab.com

www.uqpylab.uq-cloud.io

UQ[py]Lab

**The Uncertainty Quantification Community**

www.uqworld.org