



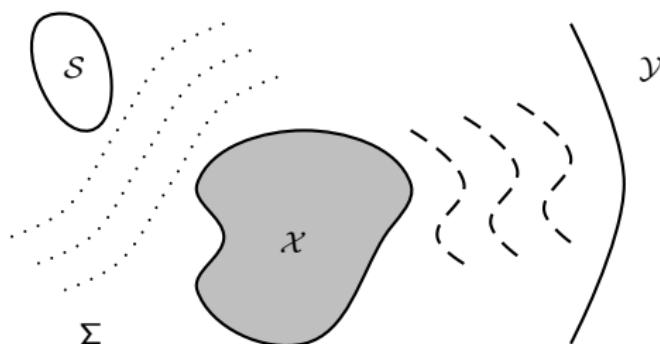
Métamodèles de chaos polynomial en imagerie électromagnétique

Charles Boulitrop

Université Paris-Saclay, CentraleSupélec, CNRS, Laboratoire de Génie Electrique et Electronique de Paris, 91192, Gif-sur-Yvette, France
Sorbonne Université, CNRS, Laboratoire de Génie Electrique et Electronique de Paris, 75252, Paris, France

1^{er} Juillet 2021

Inverse problem



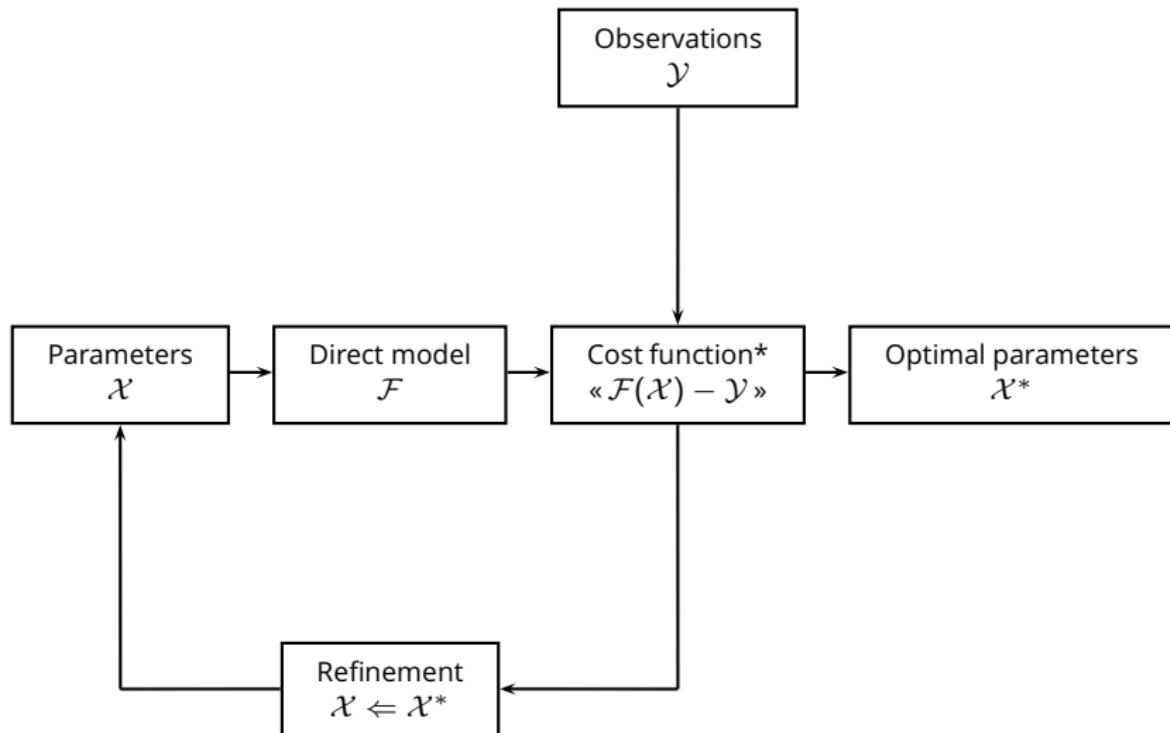
Direct problem

- Known
 - source S
 - medium Σ
 - obstacle X
- Unknown
 - observation y

Inverse problem

- Known
 - source S
 - medium Σ
 - observation y
- Unknown
 - obstacle X

Solving inverse problems iteratively



* Evaluation of the cost function may need numerous evaluations of the direct model

Metamodeling

Metamodel : « model of a model »

Usually, metamodel $\hat{\mathcal{F}}$ is defined by

- a set of metaparameters β
- $\mathcal{F}(X) \approx \hat{\mathcal{F}}_\beta(X)$ for most X in the space
- $\hat{\mathcal{F}}_\beta$ is much less computationally expensive than \mathcal{F}

Several strategies exist :

- kriging (gaussian processes)
- neural networks
- polynomial-based

Main steps to build a metamodel :

- build a metamodel with random set β (or choose it thanks to previous works)
- compute $\hat{\mathcal{F}}_\beta$ over a testing set
- refine β accordingly

Polynomial chaos

Univariate case

$$\hat{\mathcal{F}}(x) = \sum_{i=0}^{\infty} c_i \psi_i(x)$$

Multivariate (d) case

$$\hat{\mathcal{F}}(X) = \sum_{\alpha \in \mathcal{A}^{n,p}} c_{\alpha} \Psi_{\alpha}(X)$$

with

$\alpha = (\alpha_1 \dots \alpha_d)$ a multi-index of dimension d

$$\mathcal{A}^{(n,p)} = \left\{ \alpha = (\alpha_1 \dots \alpha_d) \in \mathbb{N}^d, \left(\sum_{i=1}^d \alpha_i^p \right)^{\frac{1}{p}} \leq n \right\} \text{ the subset of multi-indices}$$

$$\Psi_{\alpha}(X) = \prod_{i=1}^d \psi_{\alpha_i}(X_{\alpha_i})$$

$\beta = (n, p)$ the set of metaparameters

Polynomial chaos basis

ψ_i polynomials are all pre-determined based on the distribution of X

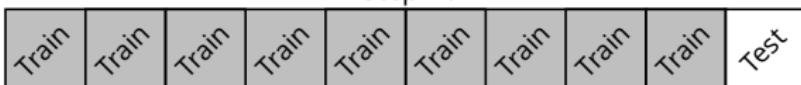
Distribution	Polynomials	Formula
$\mathcal{U}[a, b]$	Legendre	$p_i(x) = \frac{1}{2^n} \sum_{k=0}^i \binom{i}{k}^2 (x - 1)^{i-k} (x + 1)^k$
$\mathcal{N}[\mu, \sigma]$	Hermite	$h_i(x) = \sum_{k=0}^{\lfloor i/2 \rfloor} (-1)^k \frac{i!}{2^k(i-2k)!} x^{i-2k}$
$\Gamma[\lambda, k]$	Laguerre	$l_i^{(\lambda)}(x) = \frac{x^{-\lambda} e^x}{i!} \frac{d^i}{dx^i}(e^{-x} x^{i+\lambda})$
$B[r, s, \alpha, \beta]$	Jacobi	$p_i^{(r,s)}(x) = \frac{\Gamma(\alpha + i + 1)}{j! \Gamma(\alpha + \beta + i + 1)} \sum_{m=0}^i \binom{i}{m} \frac{\Gamma(\alpha + \beta + i + m + 1)}{\Gamma(\alpha + m + 1)} \left(\frac{x - 1}{2}\right)^m$

Cross-validation

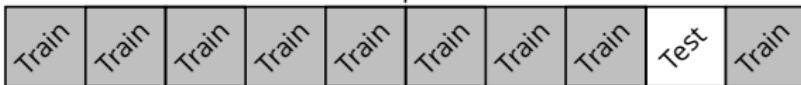
Initial set :

$$(X^{(1)}, Y^{(1)}) \dots (X^{(i)}, Y^{(i)}) \dots (X^{(M)}, Y^{(M)})$$

Step 1 :



Step 2 :



...

Step k :

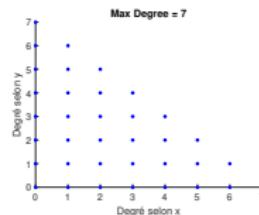
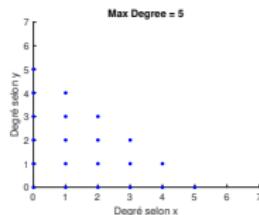
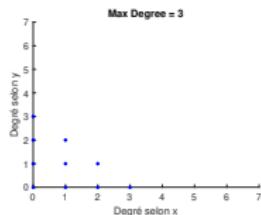


The performance of the metamodel is the average performance of all steps

Tuning metaparameters

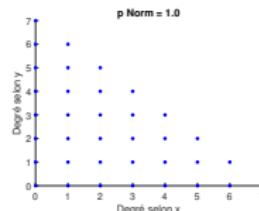
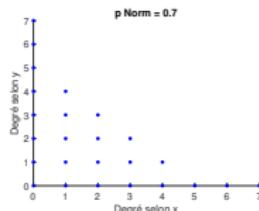
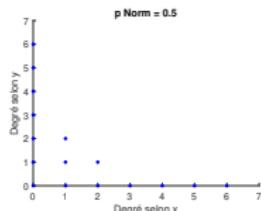
$$\mathcal{A}^{(n,p)} = \left\{ \alpha = (\alpha_1 \dots \alpha_d) \in \mathbb{N}^d, \left(\sum_{i=1}^d \alpha_i^p \right)^{\frac{1}{p}} \leq n \right\}$$

Maximal degree n :



Distribution of coefficients for various maximal degrees

Hyperbolic norm p :



Distribution of coefficients for various norms

Metaparameter study

- Root mean square error

$$\epsilon_{rms} = \frac{1}{N} \sum_{i=1}^N \frac{\|\hat{Y}^{(i)} - Y^{(i)}\|^2}{\|Y^{(i)}\|^2}$$

- Relative mean

$$\epsilon_{rel} = \frac{1}{N} \sum_{i=1}^N \frac{|\hat{Y}^{(i)} - Y^{(i)}|}{Y^{(i)}}$$

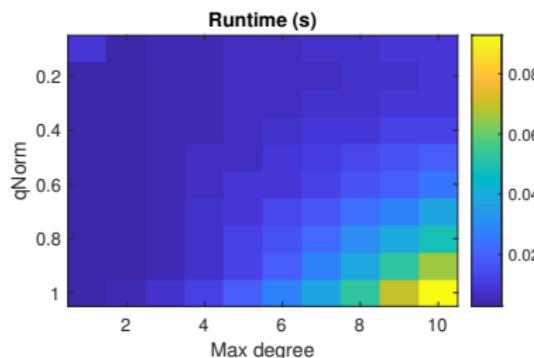
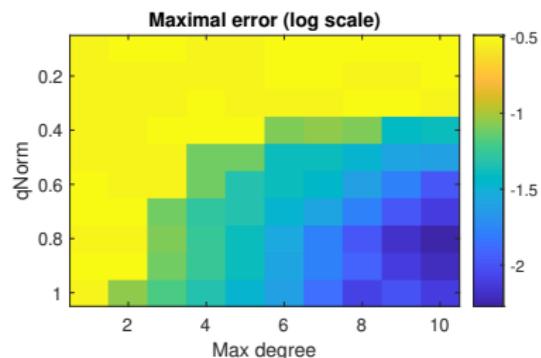
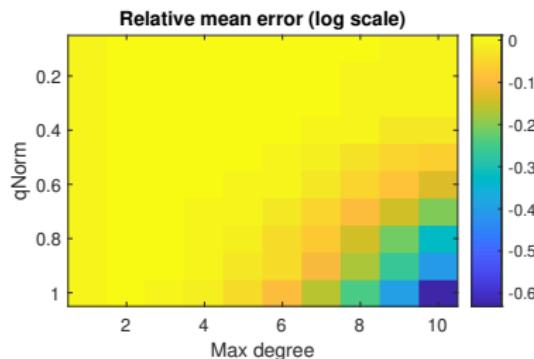
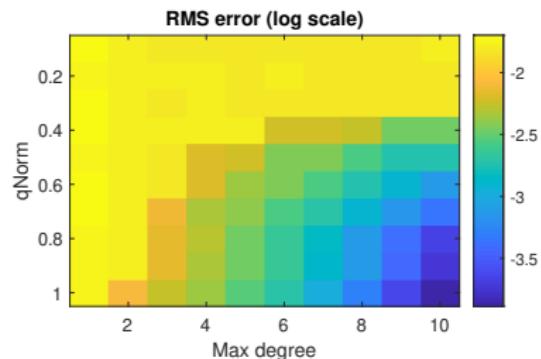
- Max error

$$\epsilon_{max} = \frac{\max_i |\hat{Y}^{(i)} - Y^{(i)}|}{\max_i |Y^{(i)}|}$$

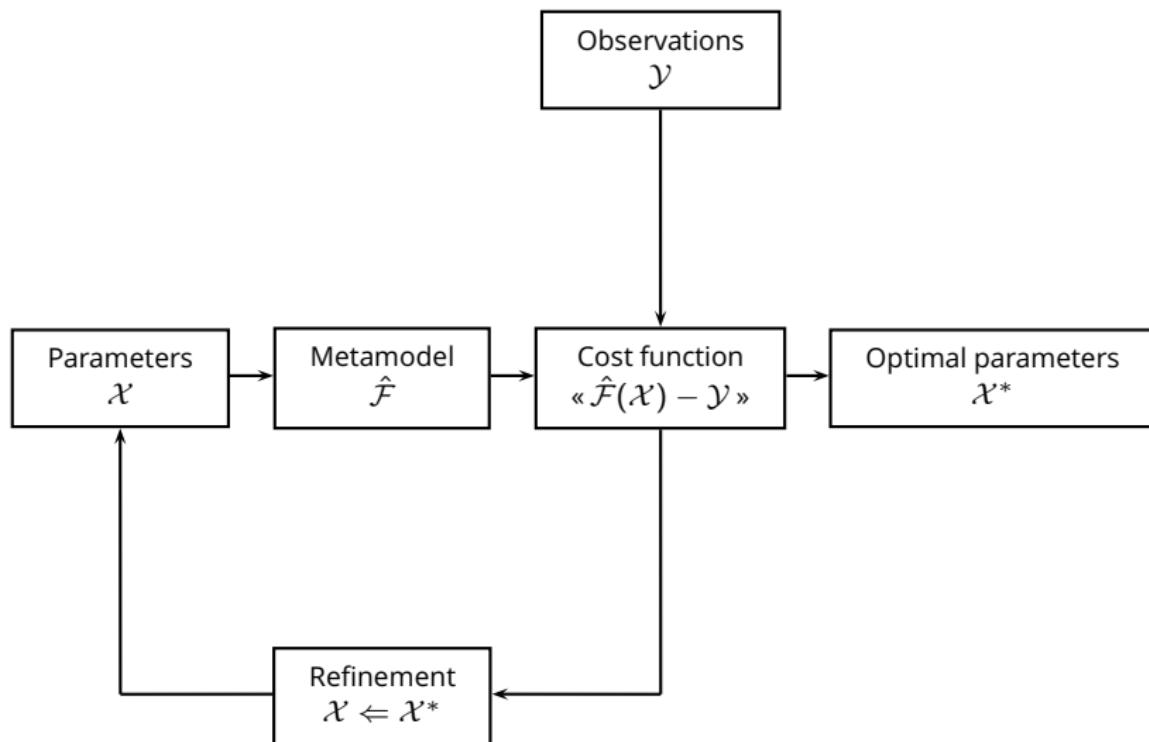
- Runtime

Metaparameter study

10 runs, OLS method, 10-fold validation



Solving inverse problems with metamodels



Particle Swarm Optimization

Iterative optimization algorithm^{1 2} (KENNEDY et EBERHART 1995, CLERC 2010)

Initialization : swarm of particles, « best » position

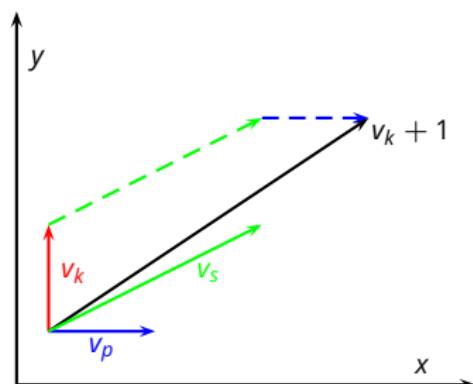
→ S , each point's position → P

Iteration k, Particle i :

- $\mathcal{E}(X_{i,k}) = \frac{\|\hat{\mathcal{F}}(X_{i,k}) - \mathcal{Y}\|^2}{\|\mathcal{Y}\|^2}$
- if $\mathcal{E}(X_{i,k}) < \mathcal{E}(P_i)$, $X_{i,k} \rightarrow P_i$
- if $\mathcal{E}(X_{i,k}) < \mathcal{E}(S)$, $X_{i,k} \rightarrow S$
- $V_{i,k+1} = c_0 V_{i,k} + c_s \text{rand}(0, 1)(P_{ik} - X_{i,k}) + c_s \text{rand}(0, 1)(S - X_{i,k})$
with c_0 : confidence in self, c_s : confidence in swarm
- $X_{i,k+1} = X_{i,k} + V_{i,k+1}$

Stopping criterion :

- no significant variation of $\mathcal{E}(X_{i,k})$ during a given number of iterations
- threshold on $\mathcal{E}(X_{i,k})$
- maximum number of iterations reached



1. J. KENNEDY et R. EBERHART (nov. 1995). « Particle swarm optimization ». In : *Proceedings of ICNN'95 - International Conference on Neural Networks*. T. 4, 1942-1948 vol.4

2. Maurice CLERC (2010). *Particle Swarm Optimization*. T. 93. John Wiley & Sons

Gradient of the metamodel³

From the explicit formulation of the metamodel

$$\hat{\mathcal{F}}(x, y, z) = \sum_{(i,j,k) \in \mathcal{A}^{(n,p)}} c_{ijk} \psi_i(x) \psi_j(y) \psi_k(z)$$

One can obtain the gradient of the metamodel

$$\frac{\partial \hat{\mathcal{F}}}{\partial x}(x, y, z) = \sum_{(i,j,k) \in \mathcal{A}^{(n,p)}} c_{ijk} \frac{d}{dx} [\mathcal{P}_i(x)] \mathcal{P}_j(y) \mathcal{P}_k(z)$$

Which is explicit after using this recurrence formula

$$\frac{d}{dx} \mathcal{P}_{n+1}(x) = (n+1) \mathcal{P}_n(x) + x \frac{d}{dx} \mathcal{P}_n(x)$$

Only derivative w.r.t. one parameter is shown for the sake of clarity

3. B. SUDRET et C.V. MAI (2015). « Computing derivative-based global sensitivity measures using polynomial chaos expansions ». In : *Reliability Engineering & System Safety* 134, p. 241-250

Gradient of the cost function

Granted the cost function is

$$\mathcal{E}(x, y, z) = \frac{\|\hat{\mathcal{F}}(x, y, z) - \mathcal{Y}\|^2}{\|\mathcal{Y}\|^2}$$

with $\|\cdot\| = \langle \cdot, \cdot \rangle$ the L2-norm

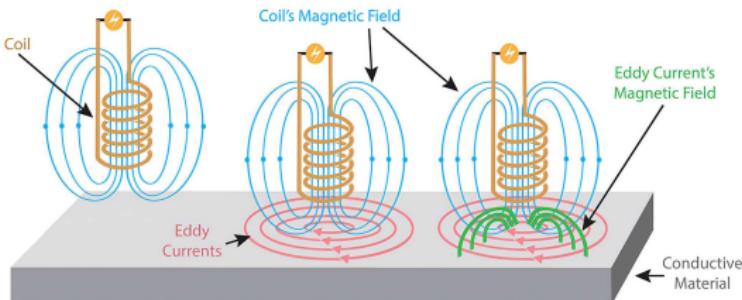
Differentiating w.r.t. each parameter yields

$$\frac{\partial}{\partial x} \mathcal{E}(x, y, z) = \frac{\partial}{\partial x} \frac{\|\hat{\mathcal{F}}(x, y, z) - \mathcal{Y}\|^2}{\|\mathcal{Y}\|^2}$$

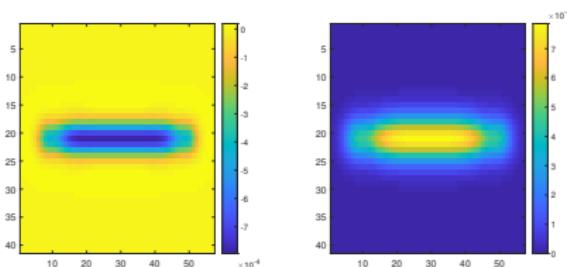
Which eventually leads to

$$\frac{\partial}{\partial x} \mathcal{E}(x, y, z) = \frac{2}{\|\mathcal{Y}\|^2} \operatorname{Re} \left\{ \left\langle \frac{\partial}{\partial x} \hat{\mathcal{F}}(x, y, z), \hat{\mathcal{F}}(x, y, z) - \mathcal{Y} \right\rangle \right\}$$

Eddy Current Testing & configuration



Basic principle of Eddy-Current Testing



Example of impedance variation map over a metal plate

In our case ($d = 3$)

$X = (x, y, z)$ uniformly distributed

$\alpha = (0, 0, 0), (0, 0, 1), (0, 1, 0), (1, 0, 0) \dots$

$$\mathcal{A}^{(n,p)} = \left\{ \alpha = (i, j, k) \in \mathbb{N}^3, (i^p + j^p + k^p)^{\frac{1}{p}} \leq n \right\}$$

$$\Psi_{000} = \psi_0(x)\psi_0(y)\psi_0(z) = \mathcal{P}_0(x)\mathcal{P}_0(y)\mathcal{P}_0(z),$$

$$\Psi_{001} = \psi_0(x)\psi_0(y)\psi_1(z) = \mathcal{P}_0(x)\mathcal{P}_0(y)\mathcal{P}_1(z), \dots$$

Making the metamodel as

$$\hat{\mathcal{F}}(x, y, z) = \sum_{(i,j,k) \in \mathcal{A}^{(n,p)}} c_{ijk} \mathcal{P}_i(x)\mathcal{P}_j(y)\mathcal{P}_k(z)$$

Computation of the c_{ijk} coefficients thanks to *UQLab*⁴ (MARELLI et SUDRET 2019) framework

4. S. MARELLI et B. SUDRET (2019). *UQLab user manual – Polynomial chaos expansions*. Rapp. tech. Report # UQLab-V1.2-104. Chair of Risk, Safety & Uncertainty Quantification, ETH Zurich

Inversion method

Algorithms

- standard PSO with PCE metamodel
- gPSO : PSO-based + gradient-based displacements at each iteration

Stopping criteria

- reaching the desired cost function value
- spending a given number of iterations with no improvement on the cost function value
- reaching the maximal number of iterations

Data

- 100 points sampled by Latin Hypercube in the input space $[-1, 1]^3$
- computation of the full metamodel for each point → impedance variation mappings
- addition of noise on magnitude and phase
- transformation into principal component space

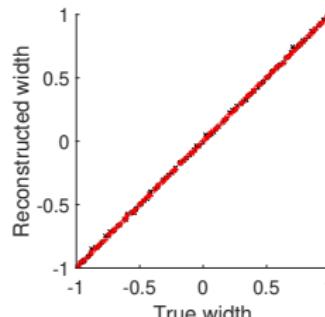
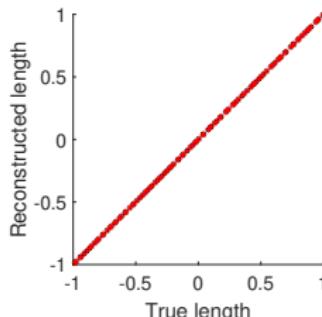
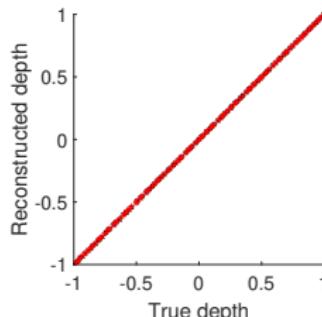
Inversion results, ideal data

Method	Depth		Length		Width	
	Mean	Std dev.	Mean	Std dev.	Mean	Std dev.
PSO	$1.67 \cdot 10^{-3}$	$1.58 \cdot 10^{-3}$	$2.18 \cdot 10^{-4}$	$1.91 \cdot 10^{-4}$	$6.81 \cdot 10^{-3}$	$7.35 \cdot 10^{-3}$
gPSO	$1.18 \cdot 10^{-3}$	$9.85 \cdot 10^{-4}$	$2.17 \cdot 10^{-4}$	$2.56 \cdot 10^{-4}$	$4.53 \cdot 10^{-3}$	$3.83 \cdot 10^{-3}$

(a) Mean and standard deviation of the absolute error for parameter reconstructions

Method	Time (s)	Number of iterations
PSO	2.60	105.57
gPSO	7.74	24.37

(b) Average time and number of iterations to reach a precision of $1 \cdot 10^{-3}$



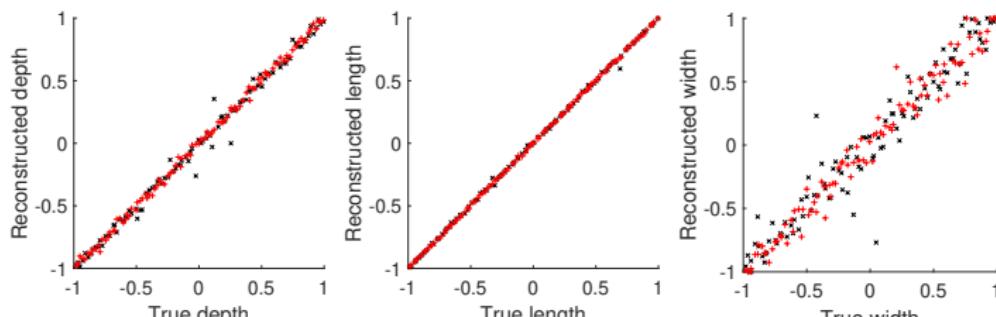
Inversion results, noise level 5

Method	Depth		Length		Width	
	Mean	Std dev.	Mean	Std dev.	Mean	Std dev.
PSO	$3.30 \cdot 10^{-2}$	$4.37 \cdot 10^{-2}$	$8.12 \cdot 10^{-3}$	$1.19 \cdot 10^{-2}$	$1.04 \cdot 10^{-1}$	$1.18 \cdot 10^{-1}$
gPSO	$2.06 \cdot 10^{-2}$	$1.52 \cdot 10^{-2}$	$5.61 \cdot 10^{-3}$	$4.52 \cdot 10^{-3}$	$7.38 \cdot 10^{-2}$	$6.67 \cdot 10^{-2}$

(c) Mean and standard deviation of the absolute error for parameter reconstructions

Method	Time (s)	Number of iterations	Cost function value
PSO	2.17	90.98	$2.432 \cdot 10^{-1}$
gPSO	26.23	80.12	$2.397 \cdot 10^{-1}$

(d) Average time and number of iterations to reach a precision of $1 \cdot 10^{-3}$

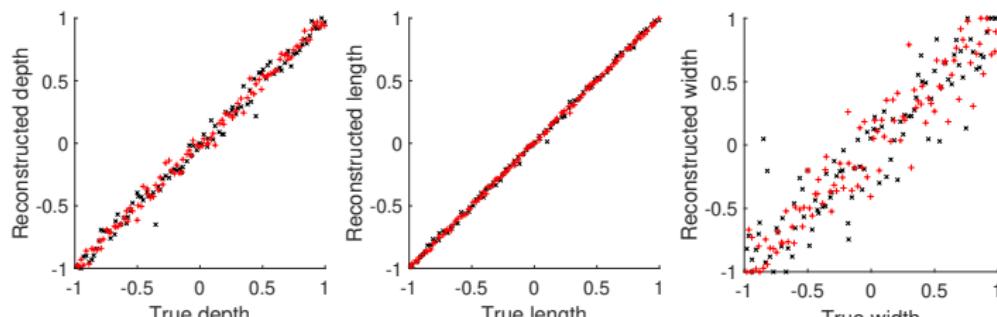


Inversion results, noise level 10

Method	Depth		Length		Width	
	Mean	Std dev.	Mean	Std dev.	Mean	Std dev.
PSO	$4.80 \cdot 10^{-2}$	$4.29 \cdot 10^{-2}$	$1.50 \cdot 10^{-2}$	$1.30 \cdot 10^{-2}$	$1.67 \cdot 10^{-1}$	$1.53 \cdot 10^{-1}$
gPSO	$3.52 \cdot 10^{-2}$	$2.99 \cdot 10^{-2}$	$9.07 \cdot 10^{-3}$	$7.97 \cdot 10^{-3}$	$1.39 \cdot 10^{-1}$	$1.13 \cdot 10^{-1}$

(e) Mean and standard deviation of the absolute error for parameter reconstructions

Method	Time (s)	Number of iterations	Cost function value
PSO	2.07	88.82	$3.877 \cdot 10^{-1}$
gPSO	29.07	82.89	$3.848 \cdot 10^{-1}$

(f) Average time and number of iterations to reach a precision of $1 \cdot 10^{-3}$ 

Conclusions & outlook

Conclusions

- Integration of a Polynomial Chaos-based metamodel inside a Particle Swarm Optimization algorithm
- Explicit gradient computation for both metamodel and cost function
- Improvement of rate of convergence of PSO thanks to metamodel gradient information (25% on mean value, 40% on variability)
- Good performances for both methods

Perspectives

- Improve gradient computation so as to minimize time per iteration
- Tests on real data
- Tests on other datasets with increasing numbers of parameters (5 and 9 parameters)
- More reliable dimension reduction⁵
- Sensitivity analysis for *a posteriori* dimension reduction

5. A. S. LEWIS et G. KNOWLES (1992). « Image compression using the 2-D wavelet transform ». In : *IEEE Transactions on Image Processing* 1.2, p. 244-250

References

-  CLERC, Maurice (2010). *Particle Swarm Optimization*. T. 93. John Wiley & Sons.
-  KENNEDY, J. et R. EBERHART (nov. 1995). « Particle swarm optimization ». In : *Proceedings of ICNN'95 - International Conference on Neural Networks*. T. 4, 1942-1948 vol.4.
-  LEWIS, A. S. et G. KNOWLES (1992). « Image compression using the 2-D wavelet transform ». In : *IEEE Transactions on Image Processing* 1.2, p. 244-250.
-  MARELLI, S. et B. SUDRET (2019). *UQLab user manual – Polynomial chaos expansions*. Rapp. tech. Report # UQLab-V1.2-104. Chair of Risk, Safety & Uncertainty Quantification, ETH Zurich.
-  SUDRET, B. et C.V. MAI (2015). « Computing derivative-based global sensitivity measures using polynomial chaos expansions ». In : *Reliability Engineering & System Safety* 134, p. 241-250.