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

1er Juillet 2021
Inverse problem

**Direct problem**
- Known
  - source $S$
  - medium $\Sigma$
  - obstacle $\mathcal{X}$
- Unknown
  - observation $\mathcal{Y}$

**Inverse problem**
- Known
  - source $S$
  - medium $\Sigma$
  - observation $\mathcal{Y}$
- Unknown
  - obstacle $\mathcal{X}$
Solving inverse problems iteratively

\[
\text{Parameters } \mathcal{X} \quad \rightarrow \quad \text{Direct model } \mathcal{F} \quad \rightarrow \quad \text{Cost function* } \langle \mathcal{F}(\mathcal{X}) - \mathcal{Y} \rangle \quad \rightarrow \quad \text{Optimal parameters } \mathcal{X}^* \\
\text{Refinement } \mathcal{X} \leftarrow \mathcal{X}^* \\
\]

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

Metamodel : « model of a model »

Usually, metamodel \( \hat{F} \) is defined by

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

Several strategies exist :

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

Main steps to build a metamodel :

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

Univariate case

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

Multivariate \((d)\) case

\[ \hat{F}(X) = \sum_{\alpha \in \mathcal{A}_{n,p}} c_\alpha \Psi_\alpha(X) \]

with

\[ \alpha = (\alpha_1 \ldots \alpha_d) \] a multi-index of dimension \(d\)

\[ \mathcal{A}_{n,p} = \left\{ \alpha = (\alpha_1 \ldots \alpha_d) \in \mathbb{N}^d, \left( \sum_{i=1}^{d} \alpha_i^p \right)^{\frac{1}{p}} \leq n \right\} \] 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**

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

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Polynomials</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>𝑈[𝑎, 𝑏]</td>
<td>Legendre</td>
<td>( pᵢ(x) = \frac{1}{2^n} \sum_{k=0}^{i} \binom{i}{k} (x - 1)^{i-k}(x + 1)^k )</td>
</tr>
<tr>
<td>𝑁[μ, σ]</td>
<td>Hermite</td>
<td>( Hᵢ(x) = \sum_{k=0}^{\lfloor i/2 \rfloor} (-1)^k \frac{i!}{2^k(i - 2k)!} x^{i-2k} )</td>
</tr>
<tr>
<td>Γ[λ, k]</td>
<td>Laguerre</td>
<td>( Lᵢ(\lambda)(x) = \frac{x^{-\lambda}e^x}{i!} \frac{d^i}{dx^i}(e^{-x}x^{i+\lambda}) )</td>
</tr>
<tr>
<td>𝐵[𝑟, 𝑠, 𝛼, 𝛽]</td>
<td>Jacobi</td>
<td>( pᵢ^{(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 )</td>
</tr>
</tbody>
</table>
Cross-validation

Initial set:

\[(X^{(1)}, Y^{(1)}) \ldots (X^{(i)}, Y^{(i)}) \ldots (X^{(M)}, Y^{(M)})\]

Step 1:

Train Train Train Train Train Train Train Train Train Test

Step 2:

Train Train Train Train Train Train Train Test Train

... 

Step k:

Test Train Train Train Train Train Train Train Train Train

The performance of the metamodel is the average performance of all steps.
Tuning metaparameters

\[ \mathcal{A}^{(n,p)} = \left\{ \alpha = (\alpha_1 \ldots \alpha_d) \in \mathbb{N}^d, \left( \sum_{i=1}^{d} \alpha_i^p \right)^{1/p} \leq n \right\} \]

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

**RMS error (log scale)**

**Relative mean error (log scale)**

**Maximal error (log scale)**

**Runtime (s)**
Solving inverse problems with metamodels

Observations $\mathcal{Y}$

Cost function $\langle \hat{F}(\mathcal{X}) - \mathcal{Y} \rangle$

Optimal parameters $\mathcal{X}^*$

Refinement $\mathcal{X} \leftarrow \mathcal{X}^*$

Parameters $\mathcal{X}$

Metamodel $\hat{F}$

Parameters $\mathcal{X}$

Building the metamodel

Inversion

Results

Conclusion

Références

Introduction
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 \(i\):

\[
\mathcal{E}(X_{i,k}) = \frac{\|\hat{\mathcal{F}}(X_{i,k}) - Y\|^2}{\|Y\|^2}
\]

if \(\mathcal{E}(X_{i,k}) < \mathcal{E}(P_i)\), \(X_{i,k} \rightarrow P_i\)

if \(\mathcal{E}(X_{i,k}) < \mathcal{E}(S)\), \(X_{i,k} \rightarrow S\)

\[
V_{i,k+1} = c_0 V_{i,k} + c_s \text{rand}(0, 1)(P_{i,k} - X_{i,k}) + c_s \text{rand}(0, 1)(S - X_{i,k})
\]

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

---

Gradient of the metamodel

From the explicit formulation of the metamodel

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

One can obtain the gradient of the metamodel

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

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

---

Gradient of the cost function

Granted the cost function is

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

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

Differentiating w.r.t. each parameter yields

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

Which eventually leads to

\[ \frac{\partial}{\partial x} \mathcal{E}(x, y, z) = \frac{2}{\| Y \|^2} \text{Re} \left\{ \langle \frac{\partial}{\partial x} \hat{F}(x, y, z), \hat{F}(x, y, z) - Y \rangle \right\} \]
Eddy Current Testing & configuration

Basic principle of Eddy-Current Testing

Example of impedance variation map over a metal plate
In our case ($d = 3$)

\[
X = (x, y, z) \text{ uniformly distributed}
\]
\[
\alpha = (0, 0, 0), (0, 0, 1), (0, 1, 0), (1, 0, 0) \ldots
\]
\[
\mathcal{A}^{(n,p)} = \left\{ \alpha = (i, j, k) \in \mathbb{N}^3, \ (i^p + j^p + k^p)^p \leq n \right\}
\]
\[
\Psi_{000} = \psi_0(x)\psi_0(y)\psi_0(z) = \mathcal{P}_0(x)\mathcal{P}_0(y)\mathcal{P}_0(z),
\Psi_{001} = \psi_0(x)\psi_0(y)\psi_1(z) = \mathcal{P}_0(x)\mathcal{P}_0(y)\mathcal{P}_1(z), \ldots
\]

Making the metamodel as

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

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

---

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Depth</th>
<th>Length</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Std dev.</td>
<td>Mean</td>
</tr>
<tr>
<td>PSO</td>
<td>1.67 \times 10^{-3}</td>
<td>1.58 \times 10^{-3}</td>
<td>2.18 \times 10^{-4}</td>
</tr>
<tr>
<td>gPSO</td>
<td>1.18 \times 10^{-3}</td>
<td>9.85 \times 10^{-4}</td>
<td>2.17 \times 10^{-4}</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Time (s)</th>
<th>Number of iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSO</td>
<td>2.60</td>
<td>105.57</td>
</tr>
<tr>
<td>gPSO</td>
<td>7.74</td>
<td>24.37</td>
</tr>
</tbody>
</table>

(b) Average time and number of iterations to reach a precision of $1 \times 10^{-3}$
Inversion results, noise level 5

<table>
<thead>
<tr>
<th>Method</th>
<th>Depth</th>
<th>Length</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Std dev.</td>
<td>Mean</td>
</tr>
<tr>
<td>PSO</td>
<td>$3.30 \cdot 10^{-2}$</td>
<td>$4.37 \cdot 10^{-2}$</td>
<td>$8.12 \cdot 10^{-3}$</td>
</tr>
<tr>
<td>gPSO</td>
<td>$2.06 \cdot 10^{-2}$</td>
<td>$1.52 \cdot 10^{-2}$</td>
<td>$5.61 \cdot 10^{-3}$</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Time (s)</th>
<th>Number of iterations</th>
<th>Cost function value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSO</td>
<td>2.17</td>
<td>90.98</td>
<td>$2.432 \cdot 10^{-1}$</td>
</tr>
<tr>
<td>gPSO</td>
<td>26.23</td>
<td>80.12</td>
<td>$2.397 \cdot 10^{-1}$</td>
</tr>
</tbody>
</table>

(d) Average time and number of iterations to reach a precision of $1 \cdot 10^{-3}$
### Inversion results, noise level 10

#### Table

<table>
<thead>
<tr>
<th>Method</th>
<th>Depth</th>
<th></th>
<th>Length</th>
<th></th>
<th>Width</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Std dev.</td>
<td>Mean</td>
<td>Std dev.</td>
<td>Mean</td>
<td>Std dev.</td>
</tr>
<tr>
<td>PSO</td>
<td>4.80 $\cdot 10^{-2}$</td>
<td>4.29 $\cdot 10^{-2}$</td>
<td>1.50 $\cdot 10^{-2}$</td>
<td>1.30 $\cdot 10^{-2}$</td>
<td>1.67 $\cdot 10^{-1}$</td>
<td>1.53 $\cdot 10^{-1}$</td>
</tr>
<tr>
<td>gPSO</td>
<td>3.52 $\cdot 10^{-2}$</td>
<td>2.99 $\cdot 10^{-2}$</td>
<td>9.07 $\cdot 10^{-3}$</td>
<td>7.97 $\cdot 10^{-3}$</td>
<td>1.39 $\cdot 10^{-1}$</td>
<td>1.13 $\cdot 10^{-1}$</td>
</tr>
</tbody>
</table>

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

#### Table (continued)

<table>
<thead>
<tr>
<th>Method</th>
<th>Time (s)</th>
<th>Number of iterations</th>
<th>Cost function value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSO</td>
<td>2.07</td>
<td>88.82</td>
<td>$3.877 \cdot 10^{-1}$</td>
</tr>
<tr>
<td>gPSO</td>
<td>29.07</td>
<td>82.89</td>
<td>$3.848 \cdot 10^{-1}$</td>
</tr>
</tbody>
</table>

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

---

![Graphs showing reconstructed vs. true values for depth, length, and width](image-url)
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 reduction
- Sensitivity analysis for *aposteriori* dimension reduction

References


