DE LA RECHERCHE À L'INDUSTRIE

www.cea.fr

Goal-oriented designs for statistical learning in high dimension

G. Perrin[⋆]

[⋆] CEA/DAM/DIF, Arpajon, France

IHP, Séminaire Phimeca
"Appréhender la grande dimension"
| June 6th, 2019

An increasing role for simulation in our society

- Taking advantage of always increasing computational resources, the importance of simulation keeps increasing.
- It is now completely integrated in most of the decision making processes of our society.
- Thus, simulation has not only to be descriptive, but needs to be **predictive**.
- In the following, let us focus on a system $\mathcal{S}$, whose design (dimensions, materials, initial conditions...) is characterized by a vector $\boldsymbol{x} \in \mathbb{X}$, and whose behavior is analyzed through the response function $y(\boldsymbol{x}) \in \mathbb{Y}$.
- To predict the value of $y(\boldsymbol{x})$, we assume we have access to a **parametric simulator**, y^{sim}.

Uncertainty quantification to generate trust

$$y^{\text{true}}\left(\boldsymbol{x}^{\text{true}}\right) = ?$$

Uncertainty quantification to generate trust

$$y^{\mathsf{sim}}(\boldsymbol{x}^{\mathsf{mes}}; \boldsymbol{\beta}^{\mathsf{nom}}, \boldsymbol{\nu}^{\mathsf{nom}}) = 0.42367589235124$$

$$y^{\mathsf{true}}\left(\boldsymbol{x}^{\mathsf{true}}\right) = 0.42367589235124?$$

Uncertainty quantification to generate trust

$$y^{\mathsf{sim}}(\boldsymbol{x}^{\mathsf{mes}}; \boldsymbol{\beta}^{\mathsf{nom}}, \boldsymbol{\nu}^{\mathsf{nom}} + \boldsymbol{\varepsilon}_{\nu}) = 0.423675892351\mathbf{24}$$

$$y^{\mathsf{true}}(\boldsymbol{x}^{\mathsf{true}}) \in [0.42367589235 \pm 10^{-11}]?$$

$\Rightarrow$ numerical uncertainties (finite arithmetic, compilation, resolution schemes,...),

Uncertainty quantification to generate trust

$$y^{\mathsf{sim}}(\boldsymbol{x}^{\mathsf{mes}}; \boldsymbol{\beta}^{\mathsf{nom}} + \boldsymbol{\varepsilon}_\beta, \boldsymbol{\nu}^{\mathsf{nom}} + \boldsymbol{\varepsilon}_\nu) = 0.423675892351\mathbf{24}$$

$$y^{\mathsf{true}}\left(\boldsymbol{x}^{\mathsf{true}}\right) \in \left[0.4236 \ \pm 10^{-4}\right]?$$

⇒ numerical uncertainties (finite arithmetic, compilation, resolution schemes,...),

⇒ parametric uncertainties (physical parameters, code thresholds...),

Uncertainty quantification to generate trust

$$(y^{\mathsf{sim}})(x^{\mathsf{mes}} + \varepsilon_x^{\mathsf{mes}}; \beta^{\mathsf{nom}} + \varepsilon_\beta, \nu^{\mathsf{nom}} + \varepsilon_\nu) = 0.42367589235124$$

$$y^{\mathsf{true}}(x^{\mathsf{true}}) \in [0.42 \pm 10^{-2}]?$$

- $\Rightarrow$ numerical uncertainties (finite arithmetic, compilation, resolution schemes,...),
- $\Rightarrow$ parametric uncertainties (physical parameters, code thresholds...),
- $\Rightarrow$ experimental uncertainties (construction tolerance, boundary and initial conditions...).

Uncertainty quantification to generate trust

$$(y^{\text{sim}} + \varepsilon_{\text{mod}})(x^{\text{mes}} + \varepsilon_x^{\text{mes}}; \beta^{\text{nom}} + \varepsilon_\beta, \nu^{\text{nom}} + \varepsilon_\nu) = 0.42367589235124$$

$$y^{\text{true}}(x^{\text{true}}) \in [0.4 \pm 10^{-1}]?$$

- $\Rightarrow$ numerical uncertainties (finite arithmetic, compilation, resolution schemes,...),
- $\Rightarrow$ parametric uncertainties (physical parameters, code thresholds...),
- $\Rightarrow$ experimental uncertainties (construction tolerance, boundary and initial conditions...).
- $\Rightarrow$ model uncertainties (simplifications, missing phenomena...).

Uncertainty quantification to generate trust

$$(y^{\text{sim}} + \varepsilon_{\text{mod}})(x^{\text{mes}} + \varepsilon_x^{\text{mes}}; \beta^{\text{nom}} + \varepsilon_\beta, \nu^{\text{nom}} + \varepsilon_\nu) = 0.42367589235124$$

$$y^{\text{true}}(x^{\text{true}}) \in [0.4 \pm 10^{-1}]?$$

⇒ numerical uncertainties (finite arithmetic, compilation, resolution schemes,...),

⇒ parametric uncertainties (physical parameters, code thresholds...),

⇒ experimental uncertainties (construction tolerance, boundary and initial conditions...).

⇒ model uncertainties (simplifications, missing phenomena...).

To be predictive, simulation needs to be able to take into account the different sources of uncertainty !

Some challenges of VV-UQ approaches

$$
\begin{aligned}
y^{\text{true}}(\boldsymbol{x}^{\text{true}}) &= (y^{\text{sim}} + \varepsilon_{\text{mod}})(\boldsymbol{x}^{\text{mes}} + \boldsymbol{\varepsilon}_x^{\text{mes}}; \boldsymbol{\beta}^{\text{nom}} + \boldsymbol{\varepsilon}_\beta, \boldsymbol{\nu}^{\text{nom}} + \boldsymbol{\varepsilon}_\nu) \\
&= (y^{\text{mes}} + \varepsilon_y^{\text{mes}})(\boldsymbol{x}^{\text{mes}} + \boldsymbol{\varepsilon}_x^{\text{mes}})
\end{aligned}
$$

1. **Sensitivity analysis** : find the elements of $\boldsymbol{x}$, $\boldsymbol{\beta}$ and $\boldsymbol{\nu}$ that play the most important roles on the variability of $y^{\text{sim}}(\boldsymbol{x}; \boldsymbol{\beta}, \boldsymbol{\nu})$.
2. **Code validation** : given a set of (noisy) experimental measurements, identify $\boldsymbol{\varepsilon}_\beta$, $\boldsymbol{\varepsilon}_\nu$ and ε_{mod}.
3. **Robust optimization** : find $\boldsymbol{x}^\star$ (using stochastic simulator y^{sim}) such that $\mathcal{C}(y^{\text{true}}(\boldsymbol{x}^\star))$ is minimal, with $\mathcal{C}$ a cost function.
4. **System certification** : guarantee (using stochastic simulator y^{sim}) that the risk that $y^{\text{true}}(\boldsymbol{x}^\star)$ exceeds S is lower than α (with a satisfying confidence).

The crucial role of statistical learning in VV-UQ

- To solve the former problems, we generally need a huge number of code evaluations.
- Hence, when confronted to very costly simulators, the code response has to be replaced by a surrogate model (or metamodel).

$$y^{\text{sim}} = y^{\text{meta}} + \varepsilon_{\text{meta}}.$$

⇒ **Warning !** Replacing the true code by its surrogate introduces an other source of uncertainty that also needs to be taken into account.

DE LA RECHERCHE À L'INDUSTRIE

cea

Outline

Theoretical framework

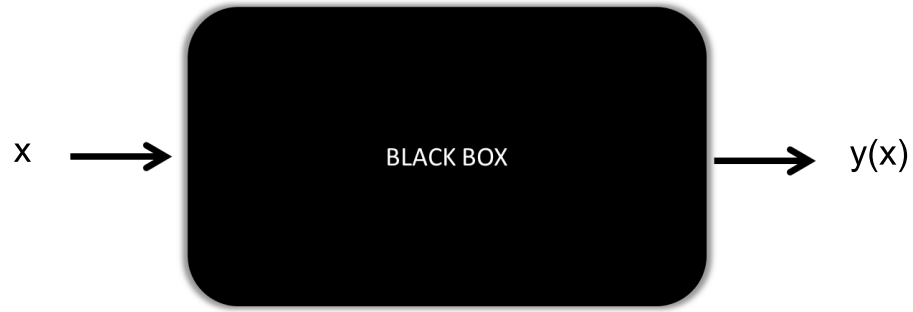
- We consider a function $y(x_1, \ldots, x_d)$ in the space $\mathbb{H} = L^2_\mu(\mathbb{X}, \mathbb{R}^{d_y})$ of square-integrable functions with values in $\mathbb{R}^{d_y}$ defined on $\mathbb{X} = \mathbb{X}_1 \times \cdots \times \mathbb{X}_d$ equipped with a probability measure $\mu = \mu_1 \otimes \cdots \otimes \mu_d$.
- The inner product and the norm in $\mathbb{H}$ are respectively written $\langle \cdot, \cdot \rangle_{\mathbb{H}}$ and $\|\cdot\|_{\mathbb{H}}$, such that for all $u, v \in \mathbb{H}$,

$$\langle u, v \rangle_{\mathbb{H}} = \int_{\mathbb{X}} u(\boldsymbol{x})^T v(\boldsymbol{x}) d\mu(\boldsymbol{x}), \quad \|u\|^2_{\mathbb{H}} = \langle u, u \rangle_{\mathbb{H}}.$$

- Each evaluation of y is supposed to be **time consuming**.

In the following, $\mathbb{X} = [-1, 1]^d$, $d\mu(\boldsymbol{x}) = \frac{d\boldsymbol{x}}{2^d}$.

Reference framework for statistical learning in computer experiment



Objective: based on $\mathcal{X} = (\boldsymbol{x}^{(n)}, y(\boldsymbol{x}^{(n)}))_{n=1}^{N}$ (constraint on the maximal budget), construct a predictor $\widehat{y}$ such that $\|\widehat{y} - y\|_{L^2}$ is minimal.

Classical formalism

- The code is modeled by a black-box code (point-wise approach), whose response is supposed to be square integrable over $\mathbb{X}$.
- The points $\boldsymbol{x}_n$, $1 \leq n \leq N$, generally correspond to a set of iid elements or to a deterministic space filling design.
- Metamodelling tools often rely on the **distance** between training and prediction points.

The curse of dimensionality

If $\mathbb{X} = [0,1]^d$, $\boldsymbol{x}_1, \boldsymbol{x}_2 \sim \mathcal{U}(\mathbb{X})$, $Z = \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^2$,

$$\mathbb{E}[Z] = \frac{d}{6} \to_{d\to+\infty} +\infty, \quad \delta^2(Z) = \frac{\mathrm{Var}(Z)}{\mathbb{E}[Z]^2} = \frac{1.4}{d} \to_{d\to+\infty} 0.$$

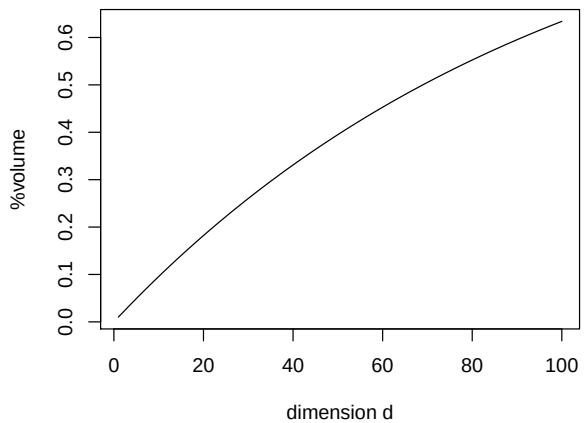
Problem 1

When the dimension of the input space increases, all points are far the ones from the others.

The curse of dimensionality

If $\mathbb{X} = [-1, 1]^d$, $\mathbb{X}^{\text{skin}} = ([-1, -0.99] \cup [0.99, 1])^d$, $\boldsymbol{x} \sim \mathcal{U}(\mathbb{X})$,

$$\text{Volume}(\mathbb{X}) = 2^d, \quad \text{Volume}(\mathbb{X}^{\text{skin}}) = 2^d - (2 \times 0.99)^d.$$

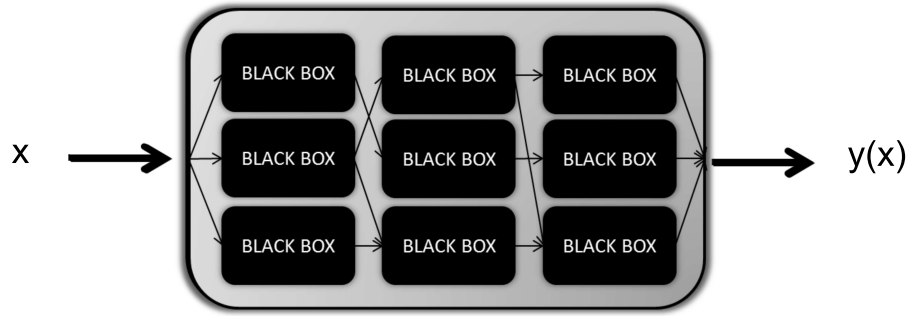


Problem 2

When the dimension of the input space increases, an important portion of the predictions have to be made in extrapolation.

The curse of dimensionality

Problematic : which (non-space filling) strategy can be proposed to improve the prediction capacity of $\widehat{y}$ when d increases ?



Direction 1

Complex codes often aggregate several "black-box codes"
⇒ how to integrate information about the code inner structure to improve the prediction capacity of $\widehat{y}$?
⇒ how to adapt the sampling strategy to this inner structure, while integrating potentially different computational costs for each code?

⇒ Gaussian process regression (GPR) for nested networks.

The curse of dimensionality

Problematic : which (non-space filling) strategy can be proposed to improve the prediction capacity of $\widehat{y}$ when d increases ?

Direction 2

When the code inner structure is hidden, or when it is not possible to separately call each code of the network, how to generalize the Principal Component Analysis to dimensions higher than 2 to identify low dimensional projection subspaces ?

$\Rightarrow$ Tree based composed functions (TBCF).

DE LA RECHERCHE À L'INDUSTRIE

cea

Outline

Existence of a code structure

- We assume that the code associated with $x \mapsto y(x)$ is constituted of a series of nested codes.
- The inner structure is supposed to be known (without loop), and the intermediary results are available.
- The intermediary quantities may be scalar or vectors.
- It is possible to launch separately each code of the network.

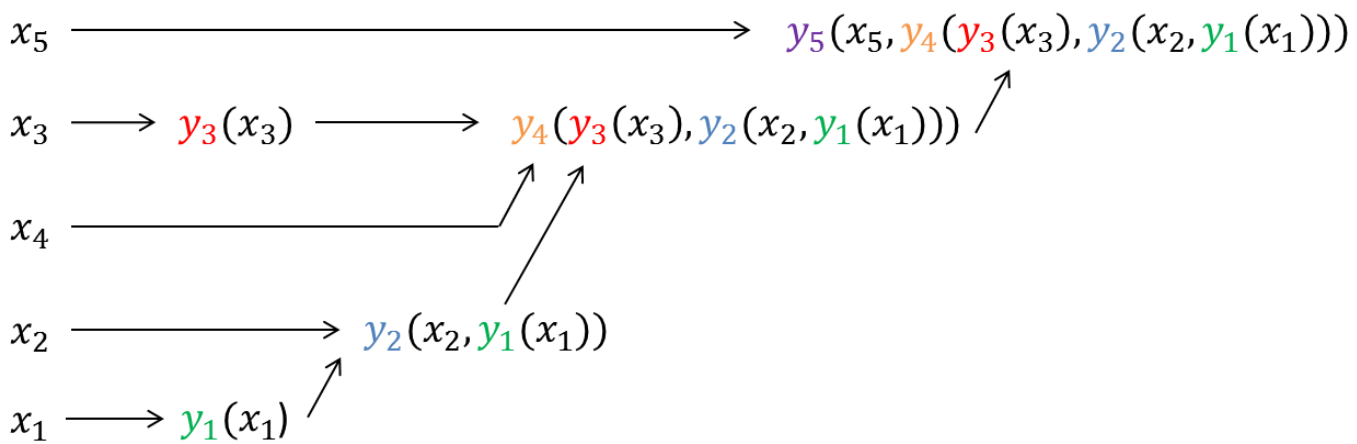


Figure: Example of a code network

GP approximation of the code network

- Each function y_i is considered as a particular realization of a GP Y_i.
- Each GP is conditioned by a series of N_i code evaluations, and we write $\mu_i + \varepsilon_i = Y_i | (\boldsymbol{x}_i^{(n_i)}, y_i(\boldsymbol{x}_i^{(n_i)}))_{n_i=1}^{N_i}$, with ε_i the (non-stationary) centered GP characterizing the metamodel uncertainty.

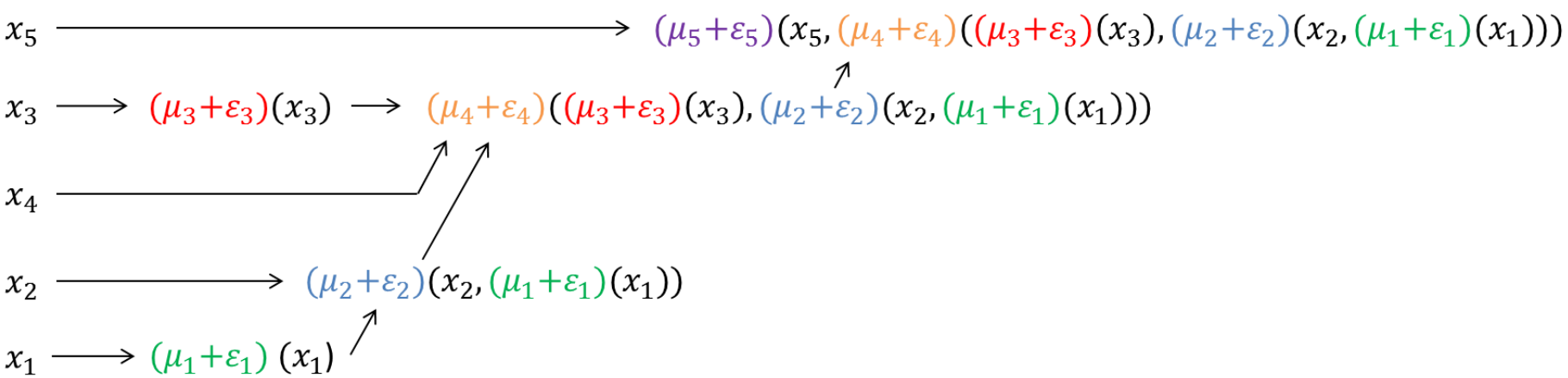


Figure: Replacement of the code functions by their GP surrogates

Construction of the predictor

- The combination of Gaussian processes is not Gaussian. Let $Y_{\text{nest}}|\mathcal{X}$ be the nested random process that aggregates all these conditioned GPs.
- Using sampling techniques (such as MCS), it is possible to propagate the surrogate uncertainties through the GP network to compute an empirical estimate of the mean of $Y_{\text{nest}}|\mathcal{X}$ based on M independent realizations of each GP, written $\{Y_{\text{nest}}(\omega_m)|\mathcal{X},\ 1 \leq m \leq M\}$.
- It comes:

$$\widehat{y} = \frac{1}{M} \sum_{m=1}^{M} Y_{\text{nest}}(\omega_m)|\mathcal{X}.$$

Design of experiments and error control

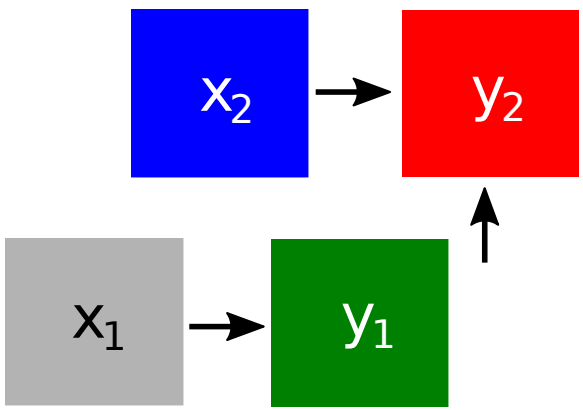
- Until now, each GP Y_i is supposed to be conditioned by the evaluations of y_i in the points $\boldsymbol{x}^{(n_i)}$, $1 \leq n_i \leq N_i$.
- The choice of these N_i points is generally motivated by the fact that the pointwise error between y_i and its GP approximation is controlled by the minimax criterion C^{mM} (up to a normalization of the inputs), which writes:

$$C^{\mathsf{mM}}(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N_i)}) = \max_{\boldsymbol{x} \in \mathbb{X}_i} \min_{1 \leq n_i \leq N_i} \left\| \boldsymbol{x} - \boldsymbol{x}^{(n_i)} \right\|_2 .$$

- Hence, the points $\boldsymbol{x}^{(n_i)}$, $1 \leq n_i \leq N_i$, most of the time correspond to **space-filling designs** (including or not constraints on the projections in subspaces of $\mathbb{X}_i$, see [Perrin and Cannamela, 2017]).

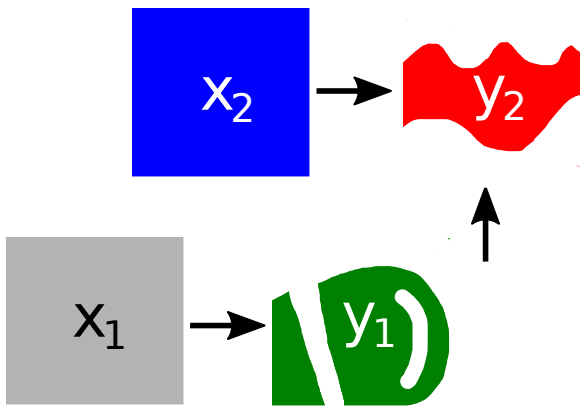
Code networks add errors ($\Rightarrow$ local enrichment)

- To reduce the prediction error on the final output, a first idea consists in minimizing the errors in each node of the network.
- Hence, the objective is to add new points in the empty regions of the different input spaces.



Code networks add errors (⇒ local enrichment)

- To reduce the prediction error on the final output, a first idea consists in minimizing the errors in each node of the network.
- Hence, the objective is to add new points in the empty regions of the different input spaces.



- To avoid useless code evaluations, the surrogate models before enrichment can be used to generate a huge number of points that are likely to be in the output spaces at the different nodes. Clustering methods can then allow selecting interesting batches of points for the enrichment.

Code networks compensate errors (⇒ global enrichment)

As an alternative, sequential designs based on a stepwise uncertainty reduction (SUR) can be proposed [Marque-Pucheu et al., 2018]:

- If the codes cannot be run separately → **Chained I-optimal**,

$$\boldsymbol{x}^{\text{new}} = \underset{\boldsymbol{x}\in\mathbb{X}}{\operatorname{argmin}} \int_{\mathbb{X}} \mathbb{V}(Y_{\text{nest}}(\boldsymbol{x}')|\mathcal{X}\cup\boldsymbol{x})d\boldsymbol{x}',$$

- If the codes can be run separately → **Best I-optimal**,

$$(i^{\text{new}}, \widetilde{\boldsymbol{x}}_{i^{\text{new}}}^{\text{new}}) = \underset{\widetilde{\boldsymbol{x}}_i\in\widetilde{\mathbb{X}}_i,\ i\leq d}{\operatorname{argmax}} \frac{1}{\tau_i} \times \int_{\mathbb{X}} \left[\mathbb{V}\left(Y_{\text{nest}}(\boldsymbol{x}')|\mathcal{X}\right) - \mathbb{V}\left(Y_{\text{nest}}(\boldsymbol{x}')|\mathcal{X}\cup\widetilde{\boldsymbol{x}}_i\right)\right] d\boldsymbol{x}',$$

where

$$(\widetilde{\boldsymbol{x}}_i, \widetilde{\mathbb{X}}_i) := \begin{cases} (\boldsymbol{x}_i, \mathbb{X}_i) \text{ if we are at a leaf of the code network,} \\ ((\boldsymbol{x}_i, y_j), \mathbb{X}_i \times \mu_j(\mathbb{X}_j)) \text{ if we are at a node,} \end{cases}$$

and τ_i is the computational cost of code i.

Remarks on the optimization problem

$$\boldsymbol{x}^{\text{new}} = \underset{\boldsymbol{x}\in\mathbb{X}}{\operatorname{argmin}}\ \mathbb{E}\left[\mathbb{V}(Y_{\text{nest}}(\boldsymbol{x}')|\mathcal{X}\cup\boldsymbol{x})\right],\ \boldsymbol{x}'\sim\mathcal{U}(\mathbb{X}).$$

☹ There is no reason for $\boldsymbol{x}\mapsto\mathbb{E}\left[\mathbb{V}(Y_{\text{nest}}(\boldsymbol{x}')|\mathcal{X}\cup\boldsymbol{x})\right]$ to be convex.

☹ As Y_{nest} is not Gaussian, there is a priori no closed-form expression for $\mathbb{E}\left[\mathbb{V}(Y_{\text{nest}}(\boldsymbol{x}')|\mathcal{X}\cup\boldsymbol{x})\right]$.

Remarks on the optimization problem

$$\boldsymbol{x}^{\text{new}} = \underset{\boldsymbol{x}\in\mathbb{X}}{\operatorname{argmin}}\ \mathbb{E}\left[\mathbb{V}(Y_{\text{nest}}^{\text{lin}}(\boldsymbol{x}')|\mathcal{X}\cup\boldsymbol{x})\right],\ \boldsymbol{x}'\sim\mathcal{U}(\mathbb{X}).$$

☹ There is no reason for $\boldsymbol{x}\mapsto\mathbb{E}\left[\mathbb{V}(Y_{\text{nest}}(\boldsymbol{x}')|\mathcal{X}\cup\boldsymbol{x})\right]$ to be convex.

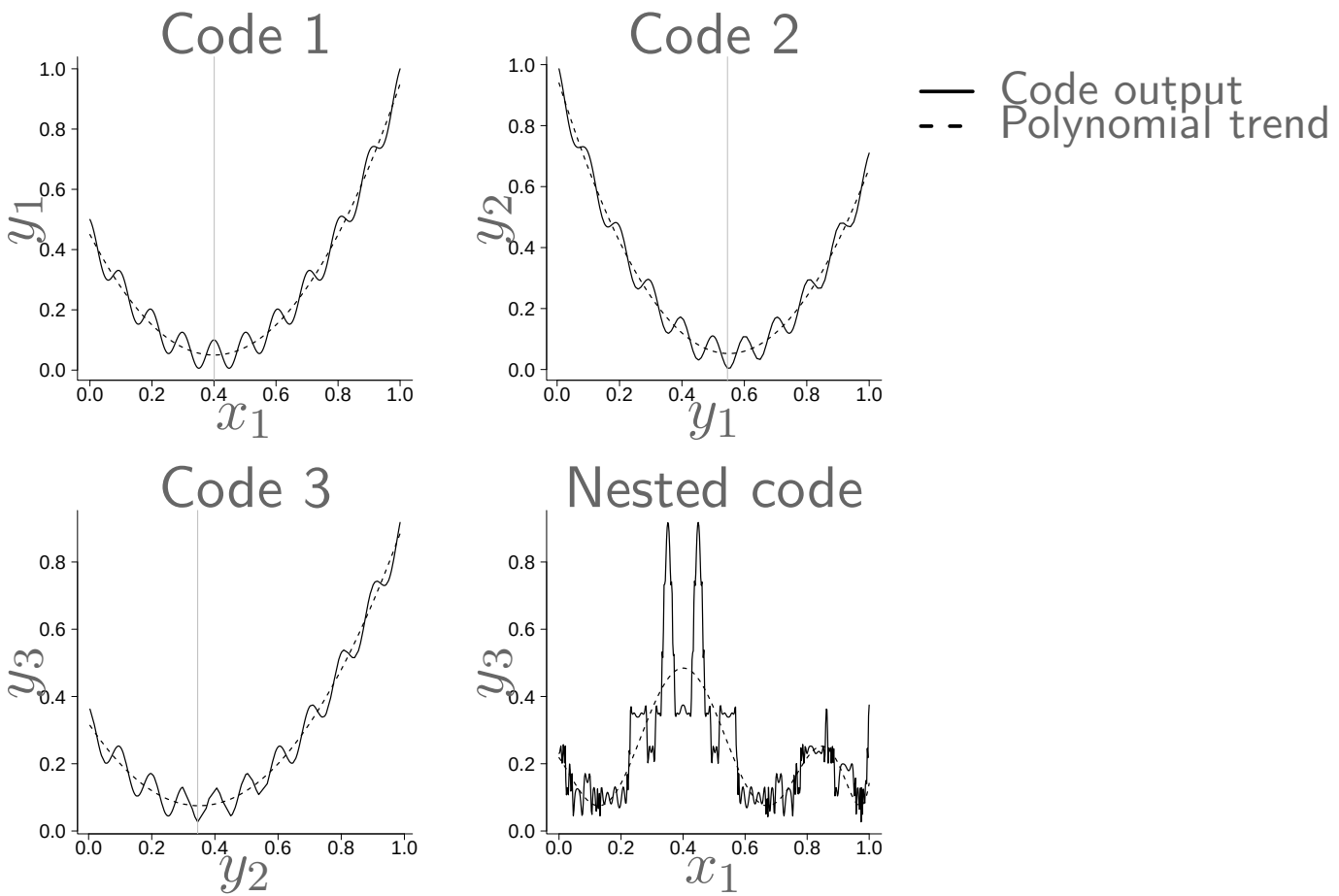
☺ $Y_{\text{nest}}^{\text{lin}}$ is the Gaussian approximation of Y_{nest} based on a series of Taylor expansions, such that :

$$\begin{aligned}&(\mu_{i+1}+\varepsilon_{i+1})(x_{i+1},(\mu_i+\varepsilon_i)(x_i))\\ &\quad\approx \mu_{i+1}(x_{i+1},\mu_i(x_i)) + \frac{\partial\mu_{i+1}}{\partial y_i}(x_{i+1},\mu_i(x_i))\varepsilon_i(x_i) + \varepsilon_{i+1}(x_{i+1},\mu_i(x_i)).\end{aligned}$$

Its variance is therefore explicitly known, and the former optimization problem becomes a classical robust optimization problem.

☺ No code evaluation is required to compute $\mathbb{E}\left[\mathbb{V}(Y_{\text{nest}}^{\text{lin}}(\boldsymbol{x}')|\mathcal{X}\cup\boldsymbol{x})\right]$ for any $\boldsymbol{x}\in\mathbb{X}$.

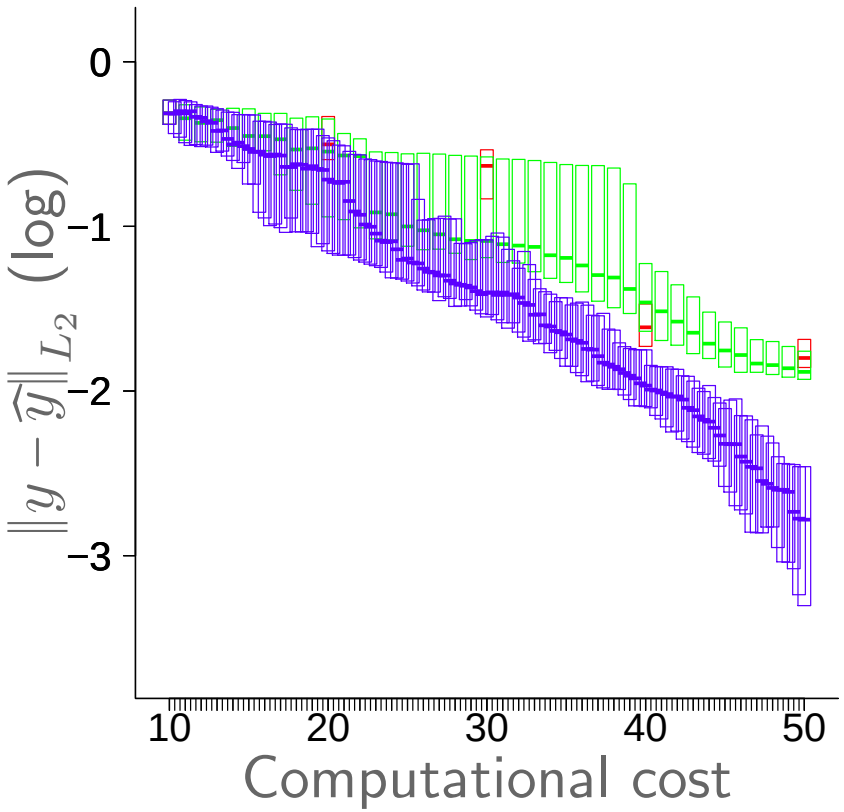
Analytical example (1/2)



$$y(\boldsymbol{x}) = y_3(y_2(y_1(x_1))).$$

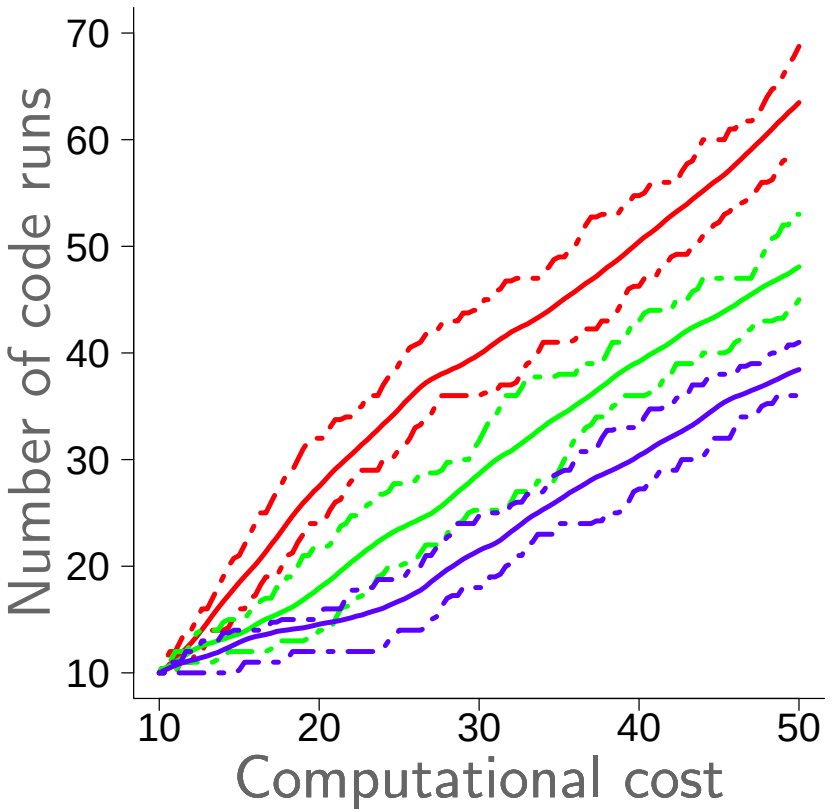
Analytical example (2/2)

Sequential vs clustering-based designs



Clustering-based
Chained I-optimal
Best I-optimal

Distribution (Best)

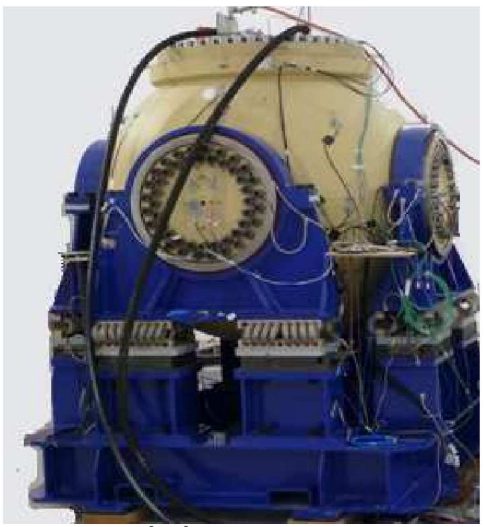


Code 1
Code 2
Code 3

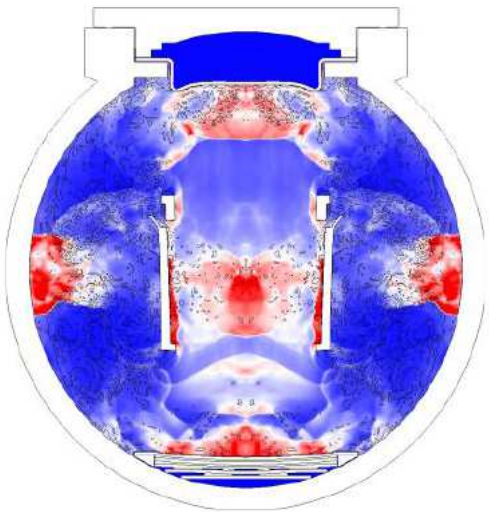
Industrial example (1/3)

Physical problem

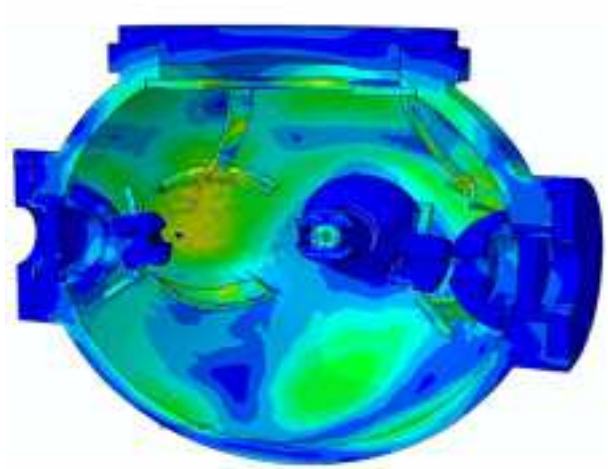
$y(\boldsymbol{x}) = y_2(\boldsymbol{x}_2, y_1(\boldsymbol{x}_1))$: coupling of a detonation code with a structural dynamics code $\Rightarrow$ functional outputs.



(a) Tank



(b) First code

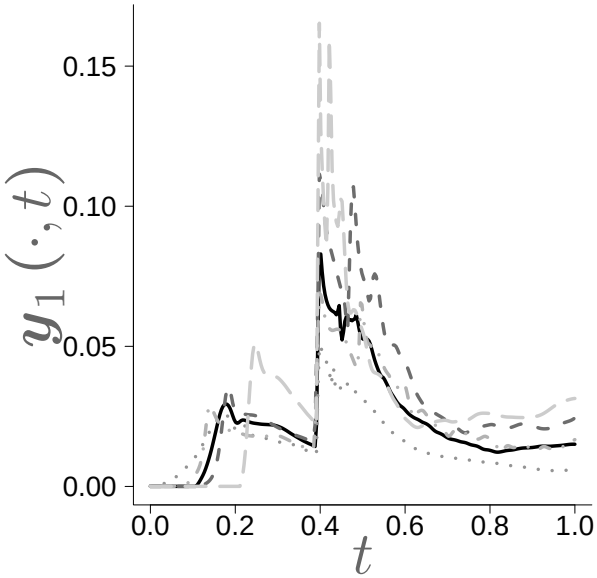


(c) Second code

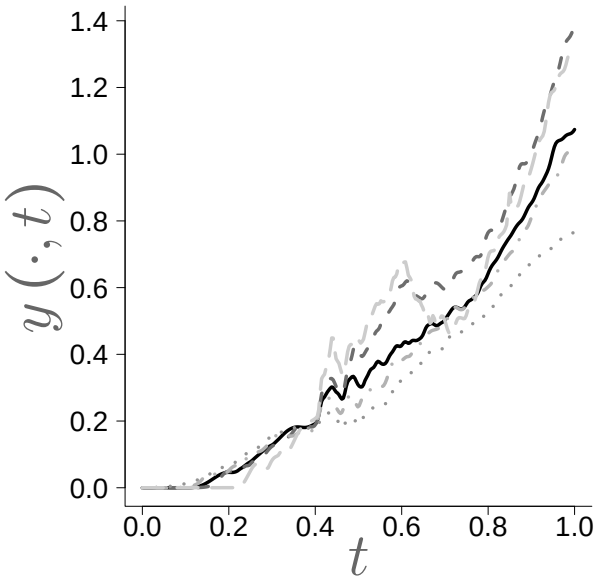
"Blind-box" $\leftrightarrow$ the code inner structure is not taken into account.

Industrial example (2/3)

- $(x_1)_1$ Radius of the explosive charge (m)
- $(x_1)_2$ Temporal dilatation parameter (-)
- $(x_1)_3$ Shock magnitude parameter (-)
- $(x_1)_4$ Attenuation parameter (-)
- y_1 Pressure generated by the shock wave

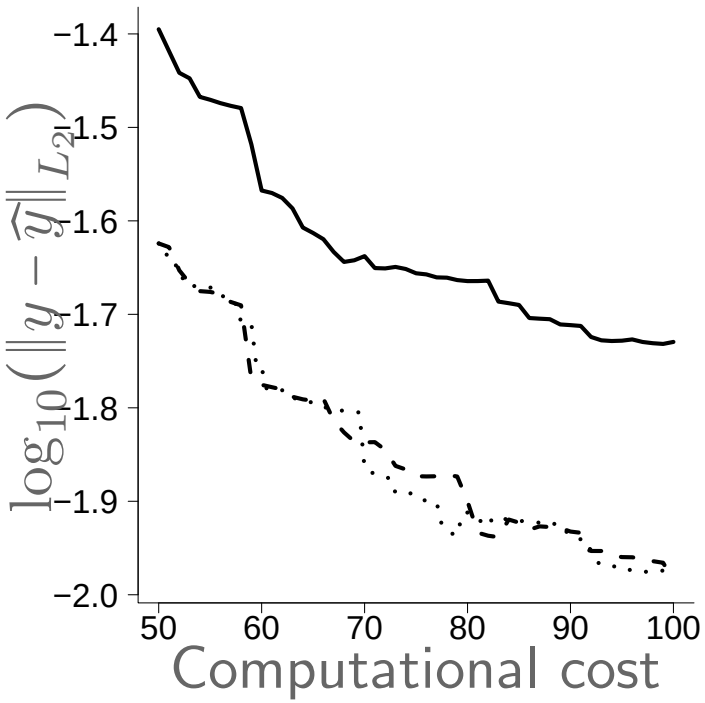


- $(x_2)_1$ Internal radius of the tank (m)
- $(x_2)_2$ Thickness (m)
- $(x_2)_3$ Young modulus of the tank (Pa)
- $(x_2)_4$ Young modulus of the tap (Pa)
- y_2 Maximal Von Mises stress over the inner surface of the tank



Industrial example (3/3)

Prediction accuracy along the sequential designs:



Blind-box I-optimal
Linearized Chained I-optimal
Linearized Best I-optimal

DE LA RECHERCHE À L'INDUSTRIE

cea

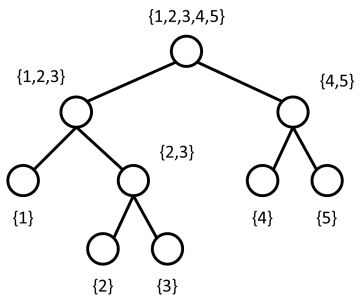
Outline

Formalism

⇒ Let T be a given dimension tree over $D = \{1, \ldots, d\}$.

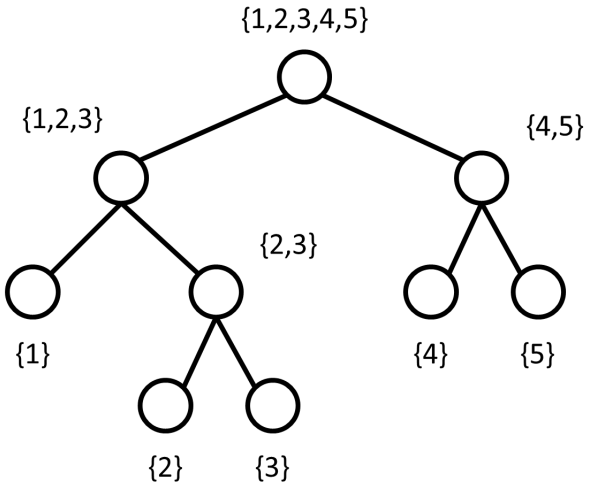
⇒ Let $\mathcal{F}^T$ be the set of functions written as a series of compositions of functions defined on subspaces of $\mathbb{X}$ controlled by T. For instance:

$$y(\boldsymbol{x}) = f_{\{1,2,3,4,5\}}(f_{\{1,2,3\}}(x_1, f_{\{2,3\}}(x_2, x_3)), f_{\{4,5\}}(x_4, x_5))$$



⇒ $\widehat{y}$ is the (empirical) projection (using $\mathcal{X}$) of y on $\mathcal{F}^T$.

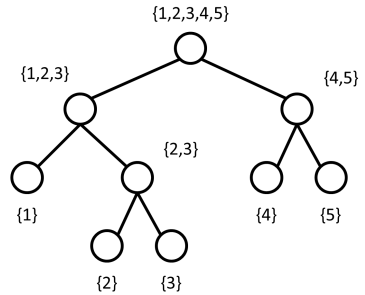
Choice of the dimension tree



- The dimension tree can be adapted to the code structure if it is known.
- Expert judgments can be used to aggregate inputs.
- The development of data-driven construction of dimension tree remains an open question.

Choice of the projection space

$$y(\boldsymbol{x}) = f_{\{1,2,3,4,5\}}(f_{\{1,2,3\}}(x_1, f_{\{2,3\}}(x_2, x_3)), f_{\{4,5\}}(x_4, x_5))$$



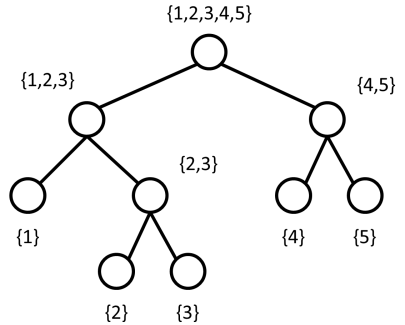
- **In theory**, the set $\mathcal{F}^T$ can gather any functions under the constraint that y is in $\mathbb{H} = L^2_\mu(\mathbb{X}, \mathbb{R}^{d_y})$.
- **In practice**, as a compromise between complexity and error control, we focus on the composition of linear functions :

$$f_{\{i,j,k\}}(x_i, x_j, x_k) = \sum_{\ell=1}^{r_{i,j,k}} c_\ell h_\ell^{(i)}(x_i) h_\ell^{(j)}(x_j) h_\ell^{(k)}(x_k),$$

where $(h_\ell^{(i)}, h_\ell^{(j)}, h_\ell^{(k)})$ are "well-chosen" functions.

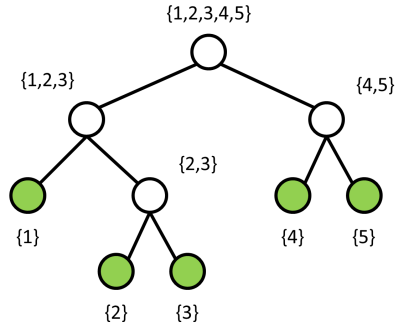
Leaves to root active learning strategy [Nouy, 2019]

$$y(\boldsymbol{x}) = f_{\{1,2,3,4,5\}}(f_{\{1,2,3\}}(x_1, f_{\{2,3\}}(x_2, x_3)), f_{\{4,5\}}(x_4, x_5))$$



Leaves to root active learning strategy [Nouy, 2019]

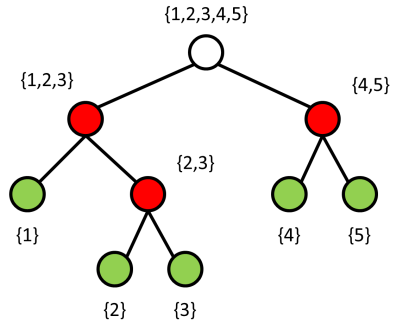
$$y(\boldsymbol{x}) = f_{\{1,2,3,4,5\}}(f_{\{1,2,3\}}(x_1, f_{\{2,3\}}(x_2, x_3)), f_{\{4,5\}}(x_4, x_5))$$



- For each leaf i (here, $i \in \{1,2,3,4,5\}$) of the tree, we introduce an approximation space $\widehat{H}_i$ (Legendre polynomials for instance). We then search U_i as the r_i-dimensional principal subspace of $\widehat{H}_i$ for y.

Leaves to root active learning strategy [Nouy, 2019]

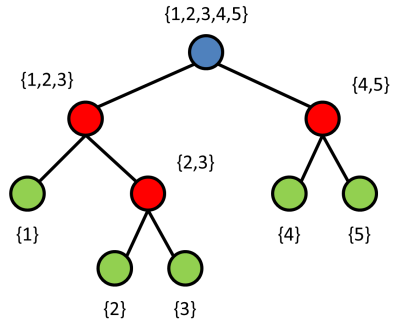
$$y(\boldsymbol{x}) = f_{\{1,2,3,4,5\}}(f_{\{1,2,3\}}(x_1, f_{\{2,3\}}(x_2, x_3)), f_{\{4,5\}}(x_4, x_5))$$



- For each leaf i (here, $i \in \{1,2,3,4,5\}$) of the tree, we introduce an approximation space $\widehat{H}_i$ (Legendre polynomials for instance). We then search U_i as the r_i-dimensional principal subspace of $\widehat{H}_i$ for y.
- For each intermediary node α (here, $\alpha \in \{\{1,2,3\}, \{2,3\}, \{4,5\}\}$), we introduce the approximation space $\widehat{H}_\alpha = \otimes_{\beta \in S(\alpha)} U_\beta$, where $S(\alpha)$ denotes the "sons" of α. We then search U_α as the r_α-dimensional principal subspace of $\widehat{H}_\alpha$ for y.

Leaves to root active learning strategy [Nouy, 2019]

$$y(\boldsymbol{x}) = f_{\{1,2,3,4,5\}}(f_{\{1,2,3\}}(x_1, f_{\{2,3\}}(x_2, x_3)), f_{\{4,5\}}(x_4, x_5))$$



- For each leaf i (here, $i \in \{1,2,3,4,5\}$) of the tree, we introduce an approximation space $\widehat{H}_i$ (Legendre polynomials for instance). We then search U_i as the r_i-dimensional principal subspace of $\widehat{H}_i$ for y.
- For each intermediary node α (here, $\alpha \in \{\{1,2,3\},\{2,3\},\{4,5\}\}$), we introduce the approximation space $\widehat{H}_\alpha = \otimes_{\beta \in S(\alpha)} U_\beta$, where $S(\alpha)$ denotes the "sons" of α. We then search U_α as the r_α-dimensional principal subspace of $\widehat{H}_\alpha$ for y.
- The final approximation $\widehat{y}$ is given by the projection of y onto the tensor product space of the principal subspaces:

$$\widehat{y} = \Pi_{U_D} y, \quad U_D = \otimes_{\beta \in S(\{1,\ldots,d\})} U_\beta.$$

Practical identification of the principal subspaces

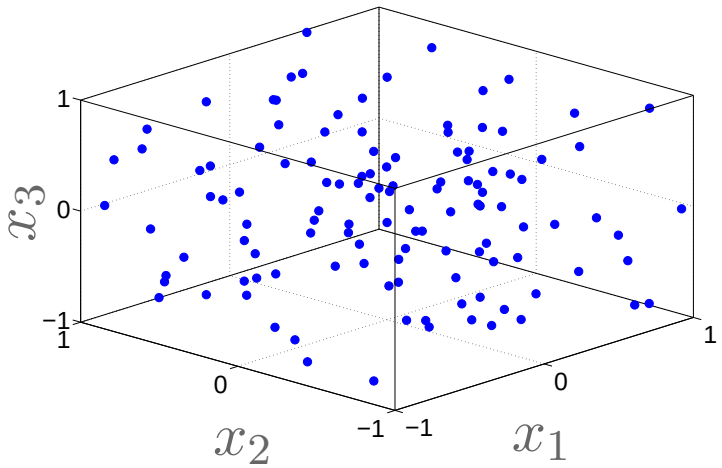
For each node $\alpha \subset \{1, \dots, d\}$ (which can be a leaf or not), the approximation of the r_α-dimensional principal subspace is based on the identification of y with a (bivariate) function z of the groups of variables $\boldsymbol{x}_\alpha$ and $\boldsymbol{x}_{\alpha^c}$, which admits the following singular value decomposition:

$$y(\boldsymbol{x}) = z(\boldsymbol{x}_\alpha, \boldsymbol{x}_{\alpha^c}) = \sum_{i=1}^{+\infty} \sqrt{\lambda_i} u_i^\alpha(\boldsymbol{x}_\alpha) u_i^{\alpha^c}(\boldsymbol{x}_{\alpha^c}), \;\; \lambda_{i+1} \leq \lambda_i.$$

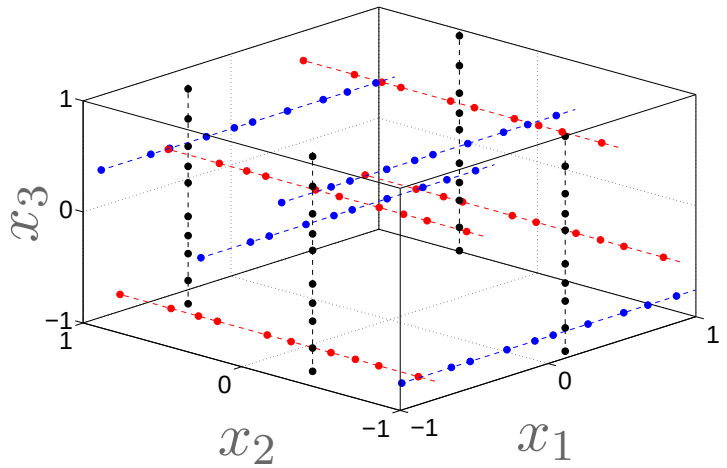
(Simplified) algorithm

- Let $\boldsymbol{x}_{\alpha^c}^{(1)}, \dots, \boldsymbol{x}_{\alpha^c}^{(M_\alpha)}$ be M_α values of $\boldsymbol{x}_{\alpha^c}$ chosen at random.
- Let $v_1^\alpha, \dots, v_{Q_\alpha}^\alpha$ be an orthonormal basis of approximation space V_α.
- Let $\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$ be the SVD-decomposition of the $(Q_\alpha \times M_\alpha)$ matrix $\widehat{\boldsymbol{W}}$ gathering the LS coefficients of the projection of $\boldsymbol{x}_\alpha \mapsto z(\boldsymbol{x}_\alpha, \boldsymbol{x}_{\alpha^c}^{(m)})$ on $v_1^\alpha, \dots, v_{Q_\alpha}^\alpha$ using P_α (**well-chosen**) evaluations of y.
- $U_\alpha \approx \widehat{U}_\alpha := \text{span}\left\{\sum_{q=1}^{Q_\alpha} (\boldsymbol{U})_{q,1} v_q^\alpha, \dots, \sum_{q=1}^{Q_\alpha} (\boldsymbol{U})_{q,r_\alpha} v_q^\alpha\right\}$.

Goal-oriented designs vs. space-filling designs



(d) Classical space-filling design



(e) Goal-oriented design for TBCF

- Focusing on the leaves, we notice an important difference in the positions where the code is evaluated between GPR and TBCF.

☺ Even if d is high, the different approximations are only made in small dimensional spaces.

☺ The algorithm is quick and stable as it is only based on a series of SVD.

Analytical example

We consider the Borehole function:

$$y(\boldsymbol{x}) = \frac{2\pi x_3(x_4 - x_6)}{(x_2 - \log(x_1))(1 + \frac{2x_7x_3}{(x_2 - \log(x_1))x_1^2x_8}) + \frac{x_3}{x_5}}$$

- $x_1 \sim \mathcal{N}(0.100, 0.0162)$,
- $x_2 \sim \mathcal{N}(7.71, 1.01)$,
- $x_3 \sim \mathcal{U}(6.31 \times 10^4, 1.16 \times 10^5)$,
- $x_4 \sim \mathcal{U}(9.90 \times 10^2, 1.11 \times 10^3)$,
- $x_5 \sim \mathcal{U}(631, 116)$,
- $x_6 \sim \mathcal{U}(700, 820)$,
- $x_7 \sim \mathcal{U}(1120, 1680)$,
- $x_8 \sim \mathcal{U}(9.86 \times 10^3, 1.20 \times 10^4)$.

Analytical example

Number of evaluations	IC90% for $\|y - \widehat{y}\|_{L_2} / \|y\|_{L_2}$
88	$[2.4 \times 10^{-2} - 2.7 \times 10^{-2}]$
308	$[1.4 \times 10^{-3} - 1.4 \times 10^{-2}]$
660	$[1.8 \times 10^{-5} - 4.9 \times 10^{-5}]$
1144	$[2.9 \times 10^{-6} - 3.5 \times 10^{-6}]$
1760	$[5.2 \times 10^{-7} - 6.1 \times 10^{-7}]$
2508	$[9.0 \times 10^{-8} - 1.3 \times 10^{-7}]$
3388	$[5.7 \times 10^{-8} - 9.2 \times 10^{-8}]$

Figure: TCBF for the approximation of the Borehole function. The approximation spaces associated with the leaves are spanned by polynomial functions of degrees less than 10.

Outline

Conclusion

- There are still many challenges for statistical learning in a small data context ($N \sim 10 - 100d$).
- In particular, we believe that important challenges lie in the introduction of effective sparse representations.
- By exploiting information about the structure of the functions to be approximated, nested GPR and TCBF are two methods that can allow us to construct such representations.
- Working on hybrid methods, which can take advantage of both techniques, remains on open subject.

References I

Marque-Pucheu, S., Perrin, G., and Garnier, J. (2018).
Efficient sequential experimental design for surrogate modeling of nested codes.
ESAIM: Probability and Statistics.

Nouy, A. (2019).
Higher-order principal component analysis for the approximation of tensors in tree-based low-rank formats.
Numerische Mathematik, 141(3):743–789.

Perrin, G. and Cannamela, C. (2017).
A repulsion-based method for the definition and the enrichment of opotimized space filling designs in constrained input spaces.
Journal de la Société Française de Statistique, 158(1):37–67.

Thank you for your attention!

Commissariat à l'énergie atomique et aux énergies alternatives
Centre de Saclay | 91191 Gif-sur-Yvette Cedex

Établissement public à caractère industriel et commercial | RCS Paris B 775 685 019