

Metamodels in structural reliability and sensitivity analysis

Jean-Marc Bourinet

LaMI / IFMA - Clermont-Ferrand, France

Based on C. Mattrand and F. Deheeger's PhD works

Surrogate model - Introduction

▲ Context

- ↳ Computationally demanding simulations (one single simulation takes mins, hours)
- ↳ Many design parameters
- ↳ Applications: design optimization, design space approximation, sensitivity analysis, reliability analysis, ...

▲ Objective

Build an **approximation model** (surrogate model / metamodel) that:

- ↳ **mimics** the behavior of the simulation model **with an acceptable accuracy** (*w.r.t.* the objective / value of interest of the study)
- ↳ is **cheaper to evaluate** (ideally numerical cost close to zero)
- ↳ requires **as few calls to the numerical models as possible**, based on a **limited number of data points intelligently selected in the design space**

Applications of surrogate models

▲ **Design space approximation** $y = f(\mathbf{x})$ for $\mathbf{x} \in D$

▲ **Optimization / Design space exploration** $\mathbf{x}^* = \arg \min_{\mathbf{x} \in D} y = f(\mathbf{x})$ s.t. $\begin{cases} g_j(\mathbf{x}) \leq 0 \\ h_k(\mathbf{x}) = 0 \end{cases}$

▲ **(Global) Sensitivity analysis (variance-based method)**

$$S_i = \frac{V(\mathbb{E}[Y|X_i])}{V(Y)}$$

$$S_{ij} = \frac{V_{ij}}{V(Y)} = \frac{V(\mathbb{E}[Y|X_i, X_j]) - V_i - V_j}{V(Y)}$$

▲ **Reliability analysis** $p_f = E_{f_{\mathbf{x}}} [I(\mathbf{X})] = \int_{g(\mathbf{x}) \leq 0} f_{\mathbf{x}}(\mathbf{x}) d\mathbf{x} = \int_{G(\mathbf{u}) \leq 0} \phi_n(\mathbf{u}) d\mathbf{u}$

where $I(\mathbf{x}) = \begin{cases} 1 & \text{if } g(\mathbf{x}) \leq 0 \\ 0 & \text{otherwise} \end{cases}$



Challenges of surrogate models

▲ **Main objective**

Model as accurate as possible, using as few simulation evaluations as possible

▲ **Main objective**

- ↳ Selection of sample points in design space (number of points, sampling strategy)
- ↳ "Passive" learning / Active learning
(selection of an initial and supposed optimal set of points / adding sequentially new points based on evaluation of previous ones)
- ↳ Construction of the surrogate model (choice of a model, eg. polynomial, ANN, MLS, Kriging, **SVM**, ... , and optimization of its parameters)
- ↳ Appraisal of the accuracy of the surrogate model (bias/variance trade-off), convergence of the whole construction process



Response Surface Method

▲ (Polynomial) Response Surface Method (RSM)

Most widely used form of surrogate model in engineering design
see e.g. [Veneziano *et al.* 1983, Faravelli 1989, Box & Draper 1987]

Assume observed data set of N points in the design space $(x_{(j)}, y_{(j)})$, $j = 1, 2, \dots, N$

Model surrogate $\hat{f}(x, M, \mathbf{a}) = a_0 + a_1x + a_2x^2 + \dots + a_Mx^M = \sum_{m=0}^M a_m x^m$

$\mathbf{a} = \langle a_0 \ a_1 \ \dots \ a_M \rangle^T$ estimated through a least-square regression of $\Phi \mathbf{a} = \mathbf{y}$

where $\Phi = \begin{bmatrix} 1 & x_{(1)} & x_{(1)}^2 & \dots & x_{(1)}^M \\ 1 & x_{(2)} & x_{(2)}^2 & \dots & x_{(2)}^M \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{(N)} & x_{(N)}^2 & \dots & x_{(N)}^M \end{bmatrix}$, $\mathbf{y} = \begin{Bmatrix} y_{(1)} \\ y_{(2)} \\ \vdots \\ y_{(N)} \end{Bmatrix}$ (vector of observed responses)

Maximum likelihood estimate of \mathbf{a} : $\mathbf{a} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{y}$



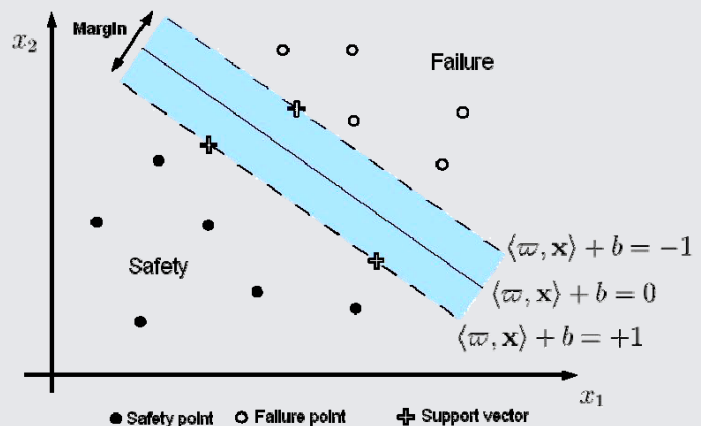
SVM classification - Linear case

↳ N training data pairs
 $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N) \in D \times \{\pm 1\}$

↳ We consider here:
 $y_j = \text{sgn}[g(\mathbf{x}_j)]$ for $j = 1, 2, \dots, N$

$y_j \in \{-1, 1\}$ called labels

Binary classification in reliability analysis: **failure / safety domain**



↳ **Objective:** find an hyperplane which is the optimal data classifier
(linearly separable classes)

$\hat{g}(\mathbf{x}) = \langle \varpi, \mathbf{x} \rangle + b = 0$

▶ The optimal classifier is obtained by maximizing the margin

ϖ : normal vector to the hyperplane



SVM classification - Corresponding optimization problem

- After a few mathematical manipulations, the problem of **maximizing the margin** is transformed to another one which consists in **minimizing the norm of ϖ** , the optimization problem is written in the form:

$$\min_{\varpi, b} \frac{\|\varpi\|^2}{2} \quad \text{subject to} \quad y_j (\langle \varpi, \mathbf{x}_j \rangle + b) \geq 1 \quad \text{for } j=1, 2, \dots, N$$

Constraints for a good classification

where $y_j \in \{-1, 1\}$ are labels corresponding to \mathbf{x}_j points

- Lagrangian formulation**

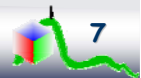
$$L(\varpi, b, \alpha) = \frac{\|\varpi\|^2}{2} - \sum_{j=1}^N \alpha_j [y_j (\langle \varpi, \mathbf{x}_j \rangle + b) - 1]$$

Convex quadratic function

- Karush-Kuhn-Tucker conditions**

► SQP algorithm (global optimum)

$$\frac{\partial L(\varpi, b, \alpha)}{\partial b} = 0 = \sum_{j=1}^N \alpha_j y_j \quad \text{and} \quad \frac{\partial L(\varpi, b, \alpha)}{\partial \varpi} = 0 = \varpi - \sum_{j=1}^N \alpha_j y_j \mathbf{x}_j$$

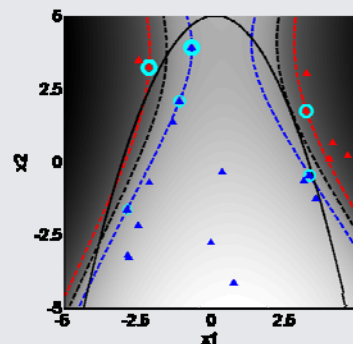


Support Vectors

- The non null α_j correspond to points **on the margin**, they are called the **Support Vectors**
- The separator can be defined only starting from these points:

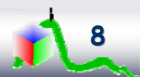
$$\varpi = \sum_{j=1}^{N_{SV}} \alpha_j y_j \mathbf{x}_j$$

where N_{SV} is the number of support vectors



- Main properties of SVM**

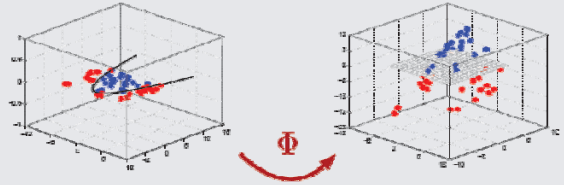
Only points added in the margin are useful to improve the accuracy of the separator



The "Kernel trick" for nonlinear classifiers

↳ For non linear cases, a **space transformation** is applied

↳ A non linear projector Φ transforms the initial space into a space of larger size, possibly infinite dimensional, called the **feature space**



↳ The non linear classification problem in the standard space becomes a linear problem after projection

▶ **Known as the "kernel trick"**

$$\omega = \sum_{j=1}^{N_{SV}} \alpha_j y_j \mathbf{x}_j \quad \text{and} \quad \hat{g}(\mathbf{x}) = \langle \omega, \mathbf{x} \rangle + b = 0 \quad \blacktriangleright \quad \hat{g}(\mathbf{x}) = \sum_{j=1}^{N_{SV}} \alpha_j y_j \langle \mathbf{x}_j, \mathbf{x} \rangle + b$$

Φ function does not need to explicitly known, only kernel function K is required

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$$

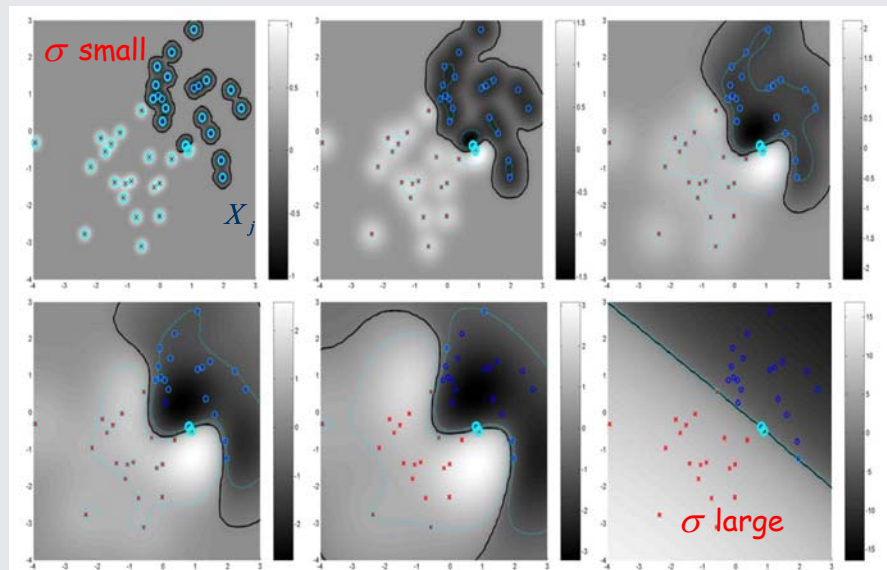
$$\hat{g}(\mathbf{x}) = \sum_{j=1}^{N_{SV}} \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}) + b$$



Gaussian kernel and hyperparameter σ

▲ **Gaussian kernel** $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$

▲ **Hyperparameter σ**



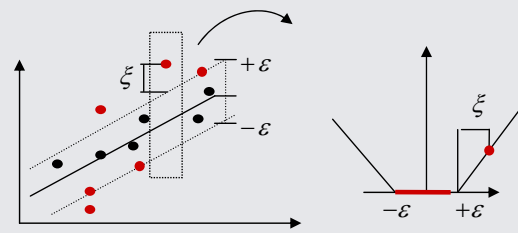
SVM regression (SVR) - Linear case

- ↳ N training data pairs
 $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N) \in D \times \mathbb{R}$

- ↳ We consider here:

$$y_j = f(\mathbf{x}_j) \quad \text{for } j=1, 2, \dots, N$$

ε -tube concept [Vapnik 1995]



Vapnik's ε -insensitive loss function
 $|y - \hat{f}(\mathbf{x})|_{\varepsilon} = \max(0, |y - \hat{f}(\mathbf{x})| - \varepsilon)$

Objective: find a function $f(\mathbf{x})$ that has at most ε deviation from the actually obtained targets y_j for all the training data and, at the same time, is as flat as possible

$$\hat{f}(\mathbf{x}) = \langle \varpi, \mathbf{x} \rangle + b \quad (\text{we suppose here the case of linear } f \text{ functions})$$



SVM regression (SVR) - Corresponding optimization problem

- ↳ **Optimization problem** to be solved:

$$\min_{\varpi, b} \frac{\|\varpi\|^2}{2} \quad \text{subject to} \quad \begin{cases} y_j - \langle \varpi, \mathbf{x}_j \rangle - b \leq \varepsilon \\ \langle \varpi, \mathbf{x}_j \rangle + b - y_j \leq \varepsilon \end{cases} \quad \text{for } j=1, 2, \dots, N$$

Flatness of f ▶ Search for small ϖ

- ↳ Introduction of **slack variables** ξ_i, ξ_i^* to cope with otherwise infeasible constraints of the optimization problem

$$\min_{\varpi, b} \frac{\|\varpi\|^2}{2} + C \sum_{j=1}^N (\xi_j + \xi_j^*) \quad \text{subject to} \quad \begin{cases} y_j - \langle \varpi, \mathbf{x}_j \rangle - b \leq \varepsilon + \xi_j \\ \langle \varpi, \mathbf{x}_j \rangle + b - y_j \leq \varepsilon - \xi_j^* \\ \xi_j, \xi_j^* \geq 0 \end{cases} \quad \text{for } j=1, 2, \dots, N$$

Constant C ▶ Trade-off between the flatness of f and the amount up to which deviations larger than ε are tolerated (**ε -insensitive loss function**)



SVM regression (SVR) - Support Vectors and Kernel Trick

▲ Support vectors (from Lagrangian dual formulation)

$$\bar{\omega} = \sum_{j=1}^{N_{SV}} (\alpha_j - \alpha_j^*) \mathbf{x}_j \quad \text{where } N_{SV} \text{ is the number of support vectors}$$

$$\bar{\omega} = \sum_{j=1}^{N_{SV}} (\alpha_j - \alpha_j^*) \mathbf{x}_j \quad \text{and} \quad \hat{f}(\mathbf{x}) = \langle \bar{\omega}, \mathbf{x} \rangle + b \quad \blacktriangleright \quad \hat{f}(\mathbf{x}) = \sum_{j=1}^{N_{SV}} (\alpha_j - \alpha_j^*) \langle \mathbf{x}_j, \mathbf{x} \rangle + b$$

▲ Kernel trick for nonlinear functions

$$\hat{f}(\mathbf{x}) = \sum_{j=1}^{N_{SV}} (\alpha_j - \alpha_j^*) K(\mathbf{x}_j, \mathbf{x}) + b$$



How to prevent overfitting: The Cross-Validation

▲ Current pitfall

Minimize the **Empirical Error** to improve a model
(**Empirical error** = Error made on the data used to train the algorithm)

Too little data / too precise model \blacktriangleright **Overfitting, bad generalization performance**

▲ Solution : Cross-validation

- ↳ Data divided into k folds (subsets of roughly equal sizes)
- ↳ For each fold k , we use the whole data set except it for training the model
- ↳ Fold k used as test set \blacktriangleright **Error on this fold**
- ↳ **Mean error on each fold** taken as a low biased estimator,
e.g. for model or parameter selection



Advantages and limitations of SVMs

▲ Advantages

- ↳ SVMs based on **Structural Risk Minimization (SRM)** principle:
Minimizes an upper bound on the generalization error as opposed to **ERM**, which minimizes the error on the training data
 - ▶ **Good generalization performance**
- ↳ Global solution of the optimization problem via SQP algorithm
- ↳ Further information comes from the margin (▶ new points to be added there)

▲ Limitations

- ↳ Choice of the kernel (strong assumption) and optimal hyperparameters
- ↳ Speed and size, both for training and testing (for millions of points, issue irrelevant here)



Sensitivity analysis / Reliability analysis

▲ (Global) sensitivity analysis (variance-based sensitivities)

- ↳ **Model:** $Y = f(\mathbf{X})$
- ↳ **Goal:** How the variation (uncertainty) in the output of a model Y can be apportioned to different sources of variation in the input \mathbf{X} of a model (in order to rank the weights of input variables)

$$S_i = \frac{V(E[Y|X_i])}{V(Y)} \quad S_{ij} = \frac{V_{ij}}{V(Y)} = \frac{V(E[Y|X_i, X_j]) - V_i - V_j}{V(Y)} \quad \dots$$

▲ Reliability analysis

- ↳ **Limit-State Function (LSF):** $g(\mathbf{x})$ By convention, $g(\mathbf{x}) \leq 0$ stands for failure
- ↳ **Goal:** Assess $p_f = P(\{g(\mathbf{X}) \leq 0\}) = E_{f_x}[I(\mathbf{X})]$ where $I(\mathbf{x}) = \begin{cases} 1 & \text{if } g(\mathbf{x}) \leq 0 \\ 0 & \text{otherwise} \end{cases}$
and, if possible, sensitivities of p_f w.r.t. input random variables and their distribution parameters (local sensitivities)



Sensitivity analysis - Sobol' indices

▲ Notations

$\mathbf{X} = \langle X_1 \ X_2 \ \dots \ X_n \rangle^T \in \mathbb{R}^n$ Vector of **independent** random variables

$f: \mathbb{R}^d \rightarrow \mathbb{R}$

$\mathbf{X} \rightarrow Y = f(\mathbf{X})$ Model output

▲ Decrease of Y variance for a random variable X_i fixed to x_i

$E[V(Y|X_i = x_i)]$ over space of outcomes x_i

▲ Law of total variance

$$V(Y) = V(E[Y|X_i]) + E[V(Y|X_i)]$$

▲ First order Sobol' sensitivity indices

$$0 \leq S_i = \frac{V(E[Y|X_i])}{V(Y)} \leq 1 \quad \text{Sensitivity of } Y \text{ to the } i^{\text{th}} \text{ random variable}$$



Sensitivity analysis - Sobol' indices

▲ High-Dimensional Model Representation (HDMR) of Y

$$Y = f(\mathbf{X}) = f_0 + \sum_{i=1}^n f_i(X_i) + \sum_{1 \leq i < j \leq n} f_{ij}(X_i, X_j) + \dots + f_{12\dots n}(X_1, X_2, \dots, X_n)$$

▲ Second order Sobol' sensitivity indices (and of superior orders)

Sensitivity measure of Y to interactions between 2 (or more) random variables

$$S_{ij} = \frac{V(f_{ij}(X_i, X_j))}{V(Y)} = \frac{V_{ij}}{V(Y)} = \frac{V(E[Y|X_i, X_j]) - V_i - V_j}{V(Y)} \quad S_{ijk} = \frac{V_{ijk}}{V(Y)} \quad \dots$$

▲ Notes on Sobol' indices

$$\sum_{i=1}^n S_i + \sum_{1 \leq i < j \leq n} S_{ij} + \dots + S_{12\dots n} = 1 \quad (2^n - 1) \text{ Sobol' indices}$$

▲ Total Sobol' indices [Homma & Saltelli 1996]

$$S_{T_i} = \frac{E[V(Y|X_{\sim i})]}{V(Y)}$$



MC estimates of Sobol' indices

▲ Expectation and variance of $Y = f(\mathbf{X})$

$$\tilde{E} = \frac{1}{N} \sum_{k=1}^N f(x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)}) \quad \tilde{V} = \frac{1}{N} \sum_{k=1}^N f^2(x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)}) - \tilde{E}^2$$

$$\blacktriangleright \tilde{E}^2 = \frac{1}{N} \sum_{k=1}^N f(x_1^{(k)(1)}, x_2^{(k)(1)}, \dots, x_n^{(k)(1)}) f(x_1^{(k)(2)}, x_2^{(k)(2)}, \dots, x_n^{(k)(2)}) \quad [\text{Saltelli 2002}]$$

▲ First order Sobol' sensitivity indices

$$\begin{aligned} V_i &= V(\mathbb{E}[Y|X_i]) \\ &= \mathbb{E}[\mathbb{E}^2[Y|X_i]] - \mathbb{E}^2[\mathbb{E}[Y|X_i]] \\ &= U_i - \mathbb{E}^2[Y] \end{aligned}$$

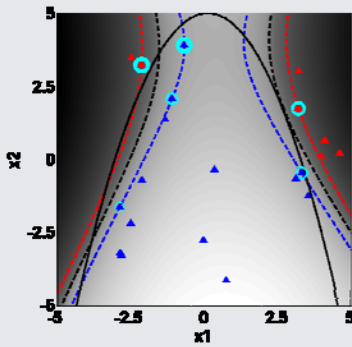
$$\begin{aligned} \tilde{U}_i &= \frac{1}{N} \sum_{k=1}^N f(x_1^{(k)(1)}, x_2^{(k)(1)}, \dots, x_{i-1}^{(k)(1)}, x_i^{(k)(1)}, x_{i+1}^{(k)(1)}, \dots, x_n^{(k)(1)}) \times \dots \\ &\quad \dots f(x_1^{(k)(2)}, x_2^{(k)(2)}, \dots, x_{i-1}^{(k)(2)}, x_i^{(k)(1)}, x_{i+1}^{(k)(2)}, \dots, x_n^{(k)(2)}) \end{aligned}$$



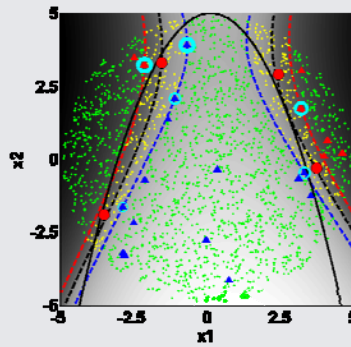
SVMs for reliability analysis



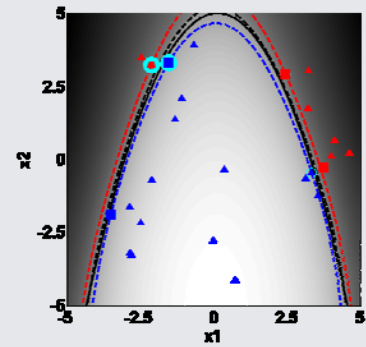
SVM surrogate model as approximation of the limit-state



Initial SVM classifier
learnt from
an initial set of data points



New data points
=
Cluster centers of
"work population" points
lying in the margin



Updated SVM classifier
learnt from
the initial set of data points
+
the extra cluster points

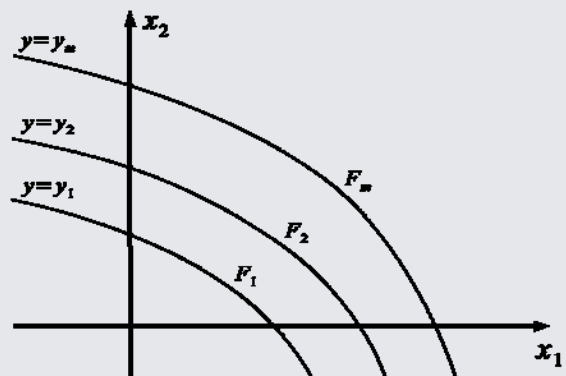
Active learning scheme



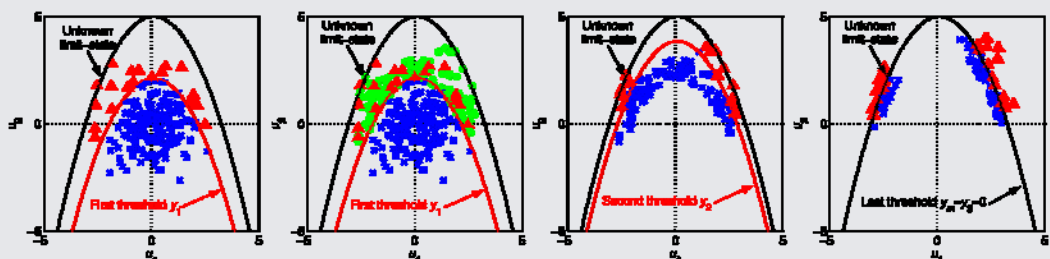
Subset simulation for small failure probabilities

Key idea [Au & Beck 2001]

$$\begin{aligned}
 p_f &= P(F) \\
 &= P(F_m) \\
 &= P(F_m | F_{m-1}) P(F_{m-1}) \\
 &= \dots \\
 &= P(F_1) \prod_{i=2}^m P(F_i | F_{i-1})
 \end{aligned}$$



Clip vidéo



²SMART algorithm

²SMART : Subset simulation by Support vector Margin Algorithm for Reliability esTimation

▲ Objective

Assess small failure probabilities, at an affordable computational cost compared to subset simulation and based on SVM surrogate models

$i = 0, y_0 = \infty$

while $y_i > 0$ [Loop on each intermediate threshold]

$i = i + 1$

I. First set of training points

→ y_i threshold, if $y_i < 0$, set $y_i = 0$

→ Initial training of SVM- y_i classifier

for $k = 1$ to $N_{\text{iter max}}$ [Update of the SVM classifier at each iteration]

II. Coarse to fine adjustment strategy for work populations: (1) Localization L , (2) Stabilization- S (3), Convergence- C

→ New points added to the learning database

→ New training of SVM- y_i classifier

end for k

III. Evaluation of $P(\bar{F}_1)$ or $P(\bar{F}_i | \bar{F}_{i-1})$

Storage of $\approx \alpha N_1$, $\approx \alpha N_2$ and $\approx \alpha N_3$ points for further applications of λr or r_j -mM algorithm if $y_i > 0$

end while

$m = i$

IV. Evaluation of P_f estimate: $P_f = P(\bar{F}_1) \prod_{i=2}^m P(\bar{F}_i | \bar{F}_{i-1})$



Main concepts of the ²SMART algorithm

▲ Active learning scheme

For each subset-like step, training from an initial set of data points plus a few new data points **added iteratively** ► Improvement of the accuracy of the SVM classifier

▲ Two kinds of data points

↳ Data points for **training the SVM classifier** ("work population")

↳ Data points for **assessing probabilities on the SVM classifier** ("work/MC population" for first subset-like step, "work/ r_j -mM population" for next steps)

▲ Coarse to fine adjustment strategy: 3 stages

1) Localization (N_1 points) 2) Stabilization (N_2 points) 3) Accuracy (N_3 points)

▲ Adding new data points at each iteration

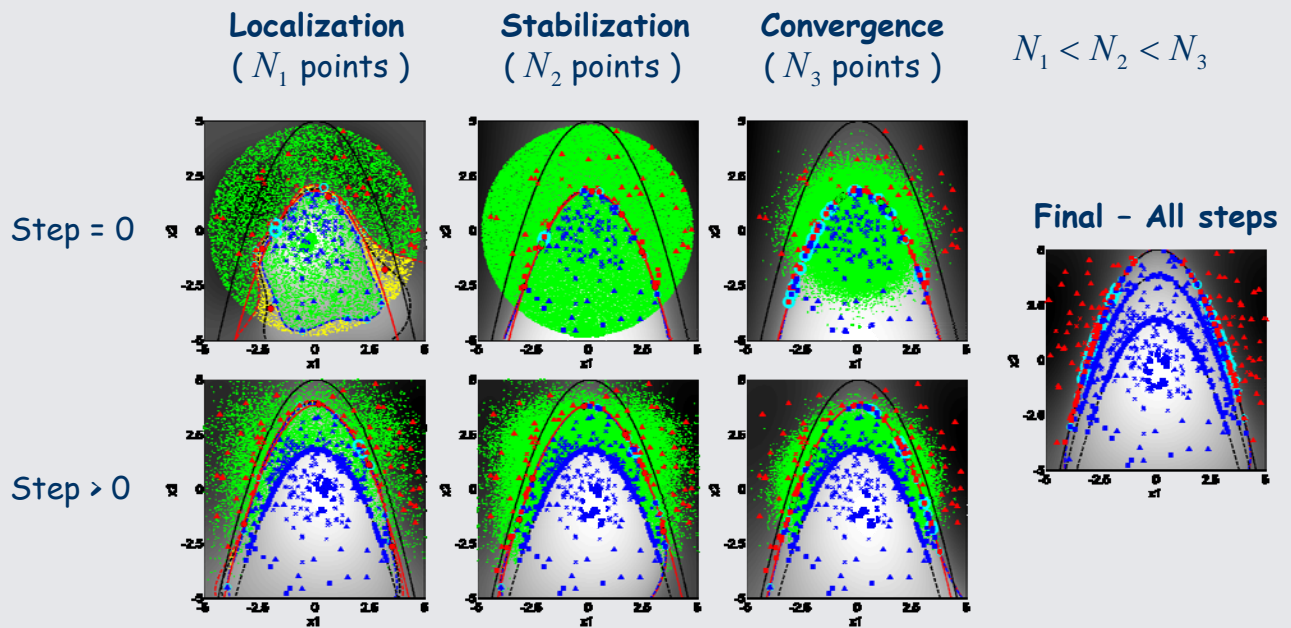
↳ Cluster centers of the "work population" points in the margin

↳ Cluster centers of "work population" points which belong to one class at a given iteration according to a current classifier and to another at the next iteration

↳ "work population" points the closest from the SVM classifier



Building the SVM classifier



Example 1: Series system

▲ Random variables (2)

$$X_1 \sim N(0,1) \quad X_2 \sim N(0,1)$$

▲ Limit-state function

$$g(x_1, x_2) = \min (g_1, g_2, g_3, g_4)$$

$$\text{where: } g_1 = 3 + 0.1(x_1 - x_2)^2 - (x_1 + x_2)/\sqrt{2}$$

$$g_2 = 3 + 0.1(x_1 - x_2)^2 + (x_1 + x_2)/\sqrt{2}$$

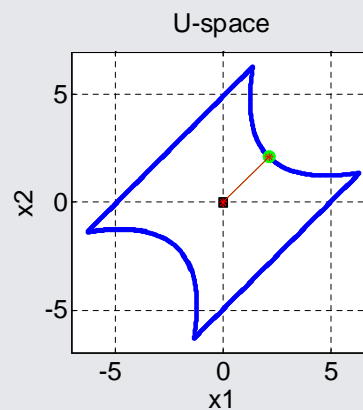
$$g_3 = x_1 - x_2 + 7/\sqrt{2}$$

$$g_4 = x_2 - x_1 + 7/\sqrt{2}$$

▲ Probability of failure (reference)

$$p_f = 2.223 \times 10^{-3} \quad (\text{DirSim, 100 dir})$$

$$p_f = 2.226 \times 10^{-3} \quad (\text{SS, } 3 \times 10^5 \text{ sim/step, 500 runs})$$



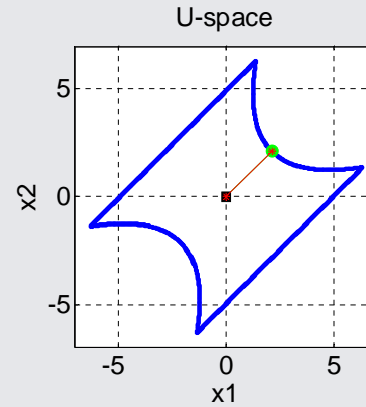
Example 1: Series system

▲ Random variables (2)

$$X_1 \sim N(0,1) \quad X_2 \sim N(0,1)$$

▲ Limit-state function

$$g(x_1, x_2) = \min(g_1, g_2, g_3, g_4)$$



p_f	$2.226 \cdot 10^{-3}$	
SS (500 runs)	1.7% 200000 / step	42% 345 / step
² SMART (50 runs)	1.7% 345 / step	0.99



Example 2: Single DSPT, Highly curved LSF

▲ Random variables (8)

Random variable	m_p	m_s	k_p	k_s	ζ_p	ζ_s	F_s	S_0
Mean	1.5	0.01	1	0.01	0.05	0.02	[11.5 ; 24.5]	100
C.o.v.	0.1	0.1	0.2	0.2	0.4	0.5	0.1	0.1

▲ Limit-state function

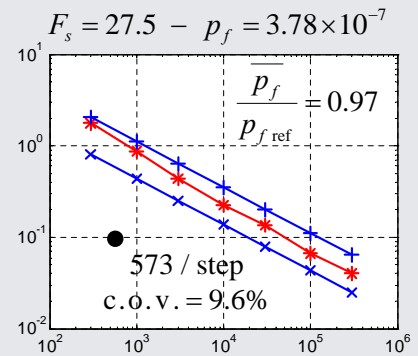
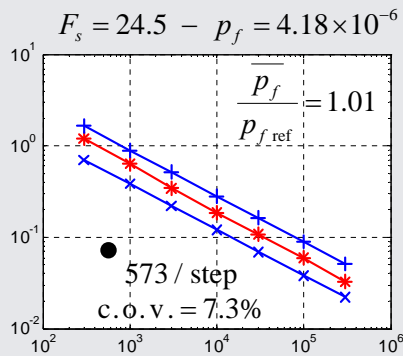
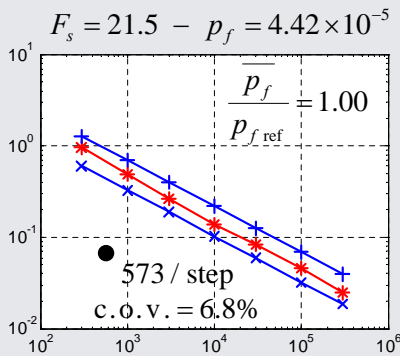
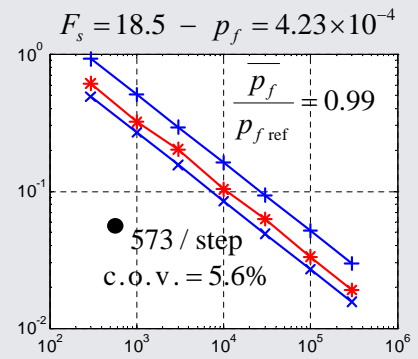
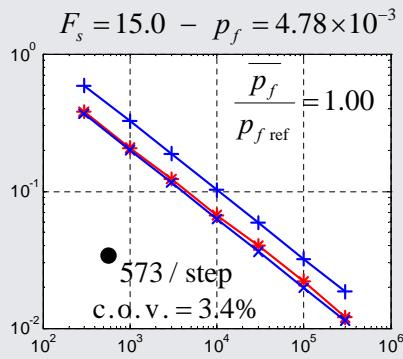
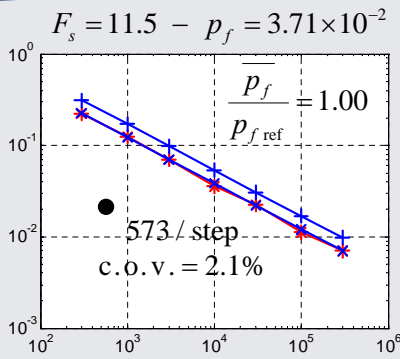
$$g = F_s - 3k_s \sqrt{\frac{\pi S_0}{4\zeta_s \omega_s^3} \frac{\zeta_a \zeta_s}{\zeta_p \zeta_s (4\zeta_a^2 + \theta^2) + \gamma \zeta_a^2} \frac{(\zeta_p \omega_p^3 + \zeta_s \omega_s^3) \omega_p}{4\zeta_a \omega_a^4}}$$

$$\text{where: } \omega_p = \sqrt{\frac{k_p}{m_p}} \quad \omega_s = \sqrt{\frac{k_s}{m_s}} \quad \gamma = \frac{m_s}{m_p}$$

$$\omega_a = \frac{1}{2}(\omega_p + \omega_s) \quad \zeta_a = \frac{1}{2}(\zeta_p + \zeta_s) \quad \theta = \frac{1}{\omega_a}(\omega_p - \omega_s)$$



Example 2: Single DSPT, Highly curved LSF



Example 3: High-dimensional LSF

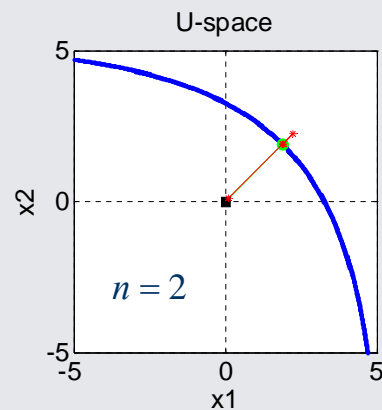
▲ Random variables (40 / 100 / 250)

$$X_i \sim LN(\mu=1, \sigma=0.2) \text{ for } i=1 \dots n$$

▲ Limit-state function

$$g(x_1, \dots, x_n) = (n\mu + a\sigma\sqrt{n}) - \sum_{i=1}^n x_i$$

where: $a = 3$ CLT: $p_f \xrightarrow[n \rightarrow \infty]{} \Phi(-a) = 1.35 \times 10^{-3}$



n	40	100	250
p_f	1.98×10^{-3}	1.73×10^{-3}	1.59×10^{-3}
SS (20 runs)	22% 1243 / step	17% 2012 / step	11% 3569 / step
² SMART (20 runs)	2.8% 1243 / step 0.99	2.2 % 2012 / step 1.01	2.6 % 3569 / step 1.01



SVR for sensitivity analysis

Constructing the SVR-surrogate

SVR-surrogate for f

1. Data points for learning model

- N_{SVR} sampling points $\left(\mathbf{u}^{(j)}, y^{(j)} = \frac{f(\mathbf{u}^{(j)}) - f_{\min}}{f_{\max} - f_{\min}} \right)$
- Deterministic sampling, in standard space

Uniform distribution in an hypersphere "Multivariate normal" distribution

2. Selection of SVR hyperparameters

- $C = 1000000$
- Grid search strategy for (ε, σ) :
 $\sigma = [1; 2; 5; 7; 8; 10; 15; 20; 50; 100]$
 $\log(\varepsilon/2) = [-11; -1; -1]$

3. Criterion for selection

Minimize empirical error + Cross validation

n random variables
 $N = 10000$ samples (QMC)

Sobol' indices
 evaluated from
 the SVR model

First order Sobol' indices
 Sobol' indices of order > 1
 Sobol' total indices

$\times 30$

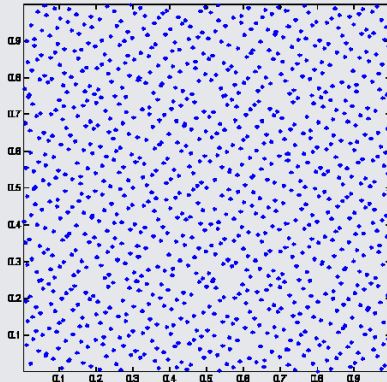
➔ Computational cost: N_{SVR}

1st sampling strategy: Multivariate "Gaussian" distribution

Methodology

- Sample points in the $[0,1]^n$ hypercube based on Sobol' low-discrepancy sequence
- Mapping of the uniformly distributed points to the standard Gaussian space

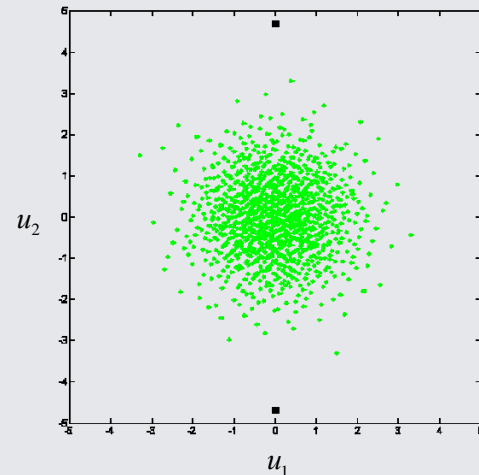
Sobol' sequence $N_{SVR} = 1000$



Mapping



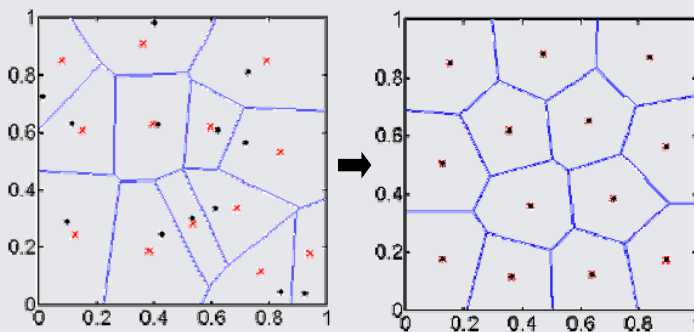
Standard space



2nd sampling strategy: Uniform distribution in hypersphere

Methodology

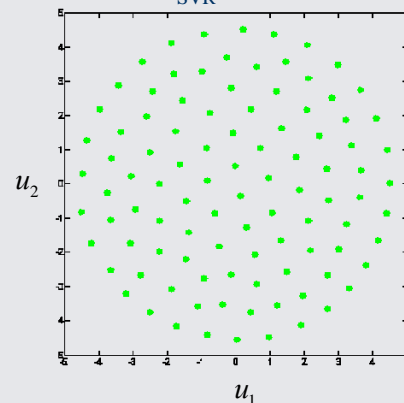
- Select the radius of an hypersphere for learning (maximum radius of $N = 10000$ normally distributed samples)
- Uniform sampling in this hypersphere using **Centroidal Voronoi Tessellation** Iterative process, see e.g. [Hanson 2005]



• Uniform random samples

• CVT centers

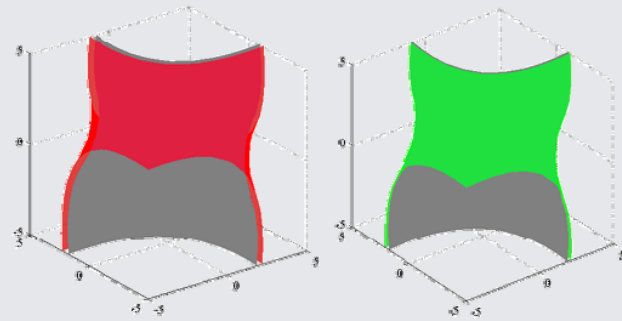
$N_{SVR} = 100$



Model with second order interactions

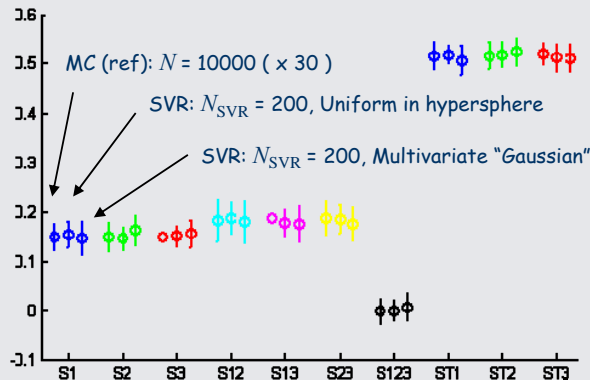
$$X_1 \sim N(0,1) \quad X_2 \sim N(0,1) \quad X_3 \sim N(0,1)$$

$$f(\mathbf{x}) = -0.5(x_1^2 + x_2^2 + x_3^2 \dots \\ \dots - 2x_1x_2 - 2x_1x_3 - 2x_2x_3) \\ - \frac{x_1 + x_2 + x_3}{\sqrt{3}} + 3$$



Multivariate "Gaussian" sampling Uniform sampling in hypersphere

$$N_{SVR} = 200, \sigma = 50, \varepsilon = 4.88 \cdot 10^{-4}, C = 1000000$$



▲ Cost (number of calls to f)

↳ Crude MC ($\times 30$):

$$(2^3 + 1) \times 10000 \times 30 \quad \text{All order indices}$$

$$(3 + 2) \times 10000 \times 30 \quad \text{Total indices only}$$

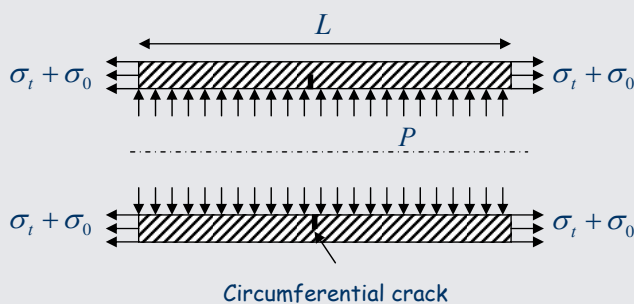
↳ SVR:

$$200 \quad \text{All indices (all orders and total)}$$



Cracked pipe - Problem definition

▲ Structure of interest



Crack length: $a = 15$ mm

Pipe length: $L = 1000$ mm

Internal pressure: $P = 15.5$ MPa

Internal radius: $R_i = 393.5$ mm

Thickness of pipe: $t = 62.5$ mm

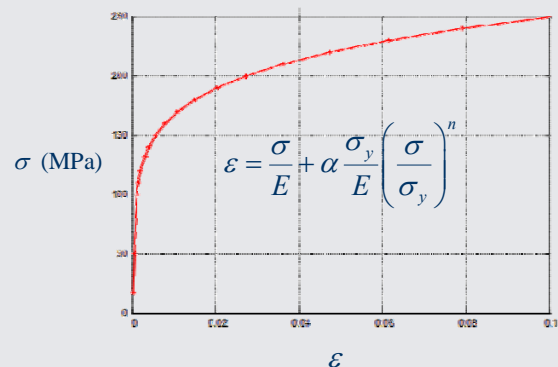
Tensile stress applied: $\sigma_t = 135$ MPa

Tensile stress from "end" effects: $\sigma_0 = P \frac{R_i^2}{(R_i + t)^2 - R_i^2}$

▲ FE code (Code_Aster)

<http://www.code-aster.org>

▲ Nonlinear material behavior (Ramberg-Osgood)



Cracked pipe - Model & Input random variables

▲ **Model** $f(\mathbf{x}) = J(E, \sigma_y, n, \alpha)$ Rice integral $J = \int_{\Gamma} \left(W dy - T \frac{\partial u}{\partial x} ds \right)$

Computational cost (one single run): ≈ 8 min on IFMA 54-cpu cluster

▲ Input (independent) random variables

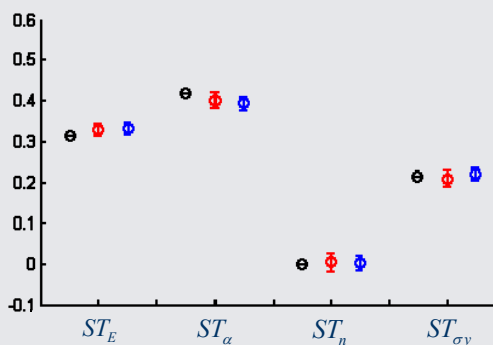
Random variable	Notation	Distribution	Mean	Coefficient of Variation
Young's modulus	E	Lognormal	175500 MPa	0.05
Constant	α	Normal	1.15	0.13
Constant (hardening behavior)	n	Normal	3.5	0.03
Yield strength	σ_y	Lognormal	259.5 MPa	0.038



Cracked pipe - Results

▲ Reference Sobol' indices (CMC)

$N = 5000$, Cost = $(n+2) \times N = 30000$	
ST_E	0.314
ST_α	0.420
ST_n	≈ 0
ST_{σ_y}	0.215



▲ Sobol' indices (SVR)

Uniform distribution in hypersphere of radius R
 $N_{SVR} = 200$, $R = 5.3$, $\sigma = 50$, $\varepsilon = 0.0039$, $C = 1000000$
 $N = 10000$ (x 30), Cost: $N_{SVR} = 500$

	Mean	Stdv	Mean	Stdv
S_E	0.364	0.006	ST_E 0.331	0.009
S_α	0.437	0.006	ST_α 0.402	0.01
S_n	0.018	0.001	ST_n ≈ 0	0.012
S_{σ_y}	0.244	0.006	ST_{σ_y} 0.210	0.01

Multivariate "Gaussian" distribution

$N_{SVR} = 200$, $R = 5.3$, $\sigma = 50$, $\varepsilon = 0.0039$, $C = 1000000$
 $N = 10000$ (x 30), Cost: $N_{SVR} = 500$

	Mean	Stdv	Mean	Stdv
S_E	0.366	0.006	ST_E 0.333	0.007
S_α	0.428	0.007	ST_α 0.394	0.009
S_n	0.016	0.0003	ST_n ≈ 0	0.009
S_{σ_y}	0.254	0.005	ST_{σ_y} 0.220	0.008



Open questions



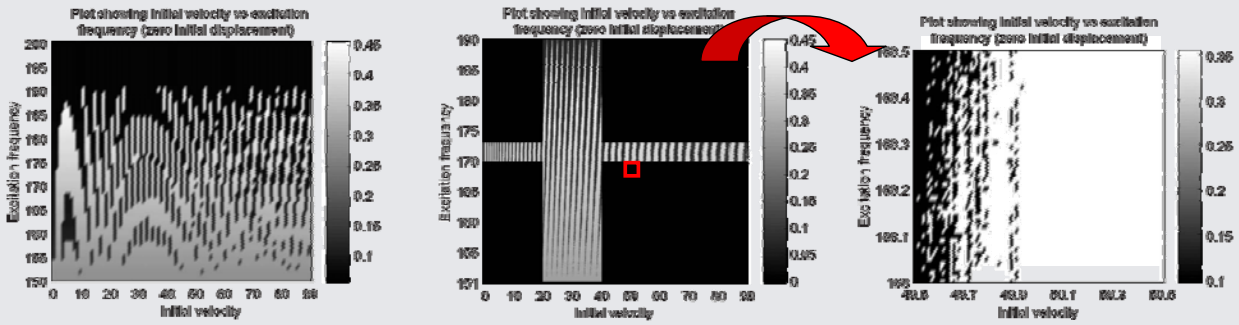
Important and open questions

- ↳ **Optimal choice for a kernel function.** An a priori choice ? Is it possible to find an optimal kernel function based on the information gained from a very few initial data points?
- ↳ **Tuning of kernel (hyper)parameters** (e.g. using cross-validation). Other options?
- ↳ **Bias/error on the quantity(ies) of interest between the surrogate and the true models** (quantity(ies) of interest: sensitivities, failure probability, optimum found in optimization problems, ...). How to estimate the accuracy of the solution?
- ↳ **Curse of dimensionality.** How the selected metamodeling strategies behave with the number of input variables?
- ↳ Need for meaningful **benchmarks**. It would be interesting to compare recent and promising metamodeling methods on a wide class of problems with typical issues (nonlinear / high-dimensional / multi-modal functions, optimization / sensitivity / reliability applications, ...)



An open problem ...

▲ Stochastic dynamics [Worden & al. 2005]

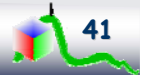


Only one single d.o.f. harmonically forced Duffing oscillator!!!

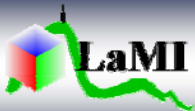
A Fractal aspect ☹️



Probabilistic Approaches in Engineering Mechanics, Aubière, 4 June 2009



41



Laboratoire de Mécanique et Ingénieries
EA 3867 - FR TMS / CNRS 2856

ER MPS

Metamodels in structural reliability and sensitivity analysis

Jean-Marc Bourinet
LaMI / IFMA - Clermont-Ferrand, France

Thanks for your attention
Any questions?



Université Blaise Pascal

