An introduction to optimization under uncertainties

CIMPA, Ulaanbaatar, Mongolia

Didier Lucor

didier.lucor@lisn.upsaclay.fr

Help from Ludovic Coelho (ONERA) & Claire Cannamela (CEA)

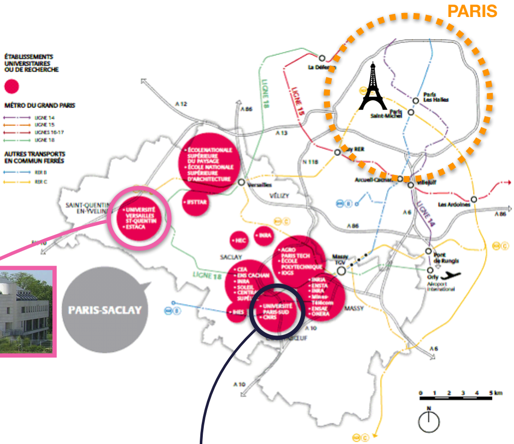
LABORATOIRE INTERDISCIPLINAIRE DES SCIENCES DU NUMÉRIQUE

université PARIS-SACLAY

Course I

Interdisciplinary Laboratory of Numerical Sciences (LISN) @ Paris-Saclay university, south-west of Paris

UNIVERSITÉ DE VERSAILLES
ST-QUENTIN-EN-YVELINES
département de mathématiques



LISN
LABORATOIRE INTERDISCIPLINAIRE
DES SCIENCES DU NUMÉRIQUE

Organization of the Optimization Under Uncertainties (OUU) class

1. Course I : Concepts, formalism and some classes of resolution methods for simple optimization under uncertainty (1.5h, Thursday)

2. Course II : Quantification of uncertainty for more efficient optimization under uncertainty : metamodeling with Gaussian Processes (1.5h, Friday)

3. Course III : Metamodeling-based Reliability-Based Design Optimization (1.5h, Saturday)

Numerical applications are carried out with **UQLab** : a MATLAB®-based open-source general purpose **Uncertainty Quantification framework** developed at ETH Zurich (Switzerland) by *Bruno Sudret*'s team.

Numerical design *"Developing something, or conceiving it, by favoring the numerical simulation over experimental processes (e.g. because it is expensive or out-of-reach)"*

Examples :

- satellites, Mars exploratory robots (Sojourner (1997), Curiosity (2012), Perseverance (2020) et Zhurong (2021) rovers...)
- nuclear power plants (nuclear accident scenario and anticipation of consequences ...)
- cars (design, crash resistance, fuel consumption ...)

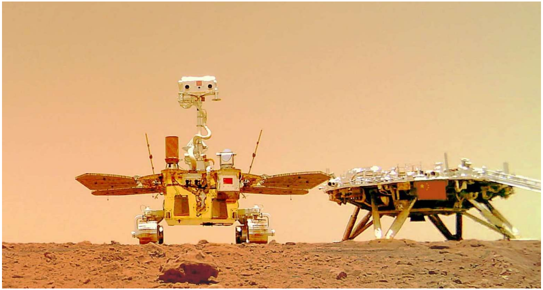
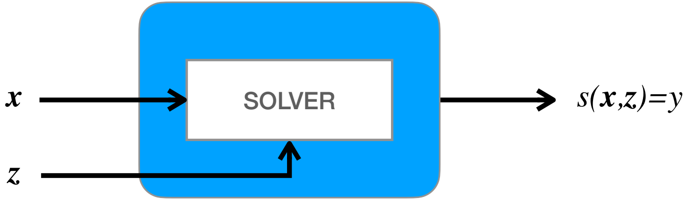


FIGURE: Zhurong rover on Mars [11/06/2021]

Numerical design *"Developing something, or conceiving it, by favoring the numerical simulation over experimental processes (e.g. because it is expensive or out-of-reach)"*

Examples :

- satellites, Mars exploratory robots (Sojourner (1997), Curiosity (2012), Perseverance (2020) et Zhurong (2021) rovers...)
- nuclear power plants (nuclear accident scenario and anticipation of consequences ...)
- cars (design, crash resistance, fuel consumption ...)



Assumption about the model :

1. model/solver/code is a **black-box** and **deterministic** given **inputs** (x, z)
2. a **single** evaluation of this model is **COSTLY !**

Design and optimisation *"Mathematical optimization : selection of a* **best** *element, with regard to some* **criterion**, *from some set of available alternatives"*

$x \leftrightarrow$ **design parameters** (geometry, material, ...), can be **"coupled"**

$z \leftrightarrow$ other (**environmental**) **parameters** : variables which **cannot** be controlled by the designer, (e.g. the loading, turbulence intensity, etc)

Design functions :

$f(\boldsymbol{x}, \boldsymbol{z}) \leftrightarrow$ **cost function** (cost, performance, efficiency, applicability...) which depends on solver output, e.g. at **unconstrained** local **optimum** : $\nabla f = 0$

$g_i(\boldsymbol{x}, \boldsymbol{z}) \leftrightarrow$ **constraints** to be satisfied (e.g. $g_i(\boldsymbol{x}, \boldsymbol{z}) = 0$) or to remain bounded (e.g. $g_i(\boldsymbol{x}, \boldsymbol{z}) \leq 0$), remark : for **constrained** optimization $\nabla f \neq 0$ at optimum

(computer-based) *deterministic* optimization

target-oriented computer-based optimization uses a numerical model mapping $\boldsymbol{x}$ to $f(\boldsymbol{x}, \boldsymbol{z} = \boldsymbol{z}_0)$ in order to find $\boldsymbol{x}^\star$ satisfying :

$$\min_{\boldsymbol{x} \in \boldsymbol{\mathcal{X}}} f(\boldsymbol{x}, \boldsymbol{z} = \boldsymbol{z}_0) \tag{1}$$

$$\text{such that } g(\boldsymbol{x}, \boldsymbol{z} = \boldsymbol{z}_0) \leq 0, \tag{2}$$

here $\boldsymbol{z} = \boldsymbol{z}_0$ corresponds to some (fixed) **nominal** conditions : i.e. $\boldsymbol{z}_0 \equiv \boldsymbol{z}_{\text{nominal}}$, the optimization is said to be **deterministic** : **no uncertainties**

toward (computer-based) optimization under uncertainty (OUU)

solving complex system design problem often faces inherent – physical **stochastic** phenomena, – **lack of knowledge**, – **modeling simplifications**, – variability of final design due to **manufacturing tolerance** etc.

$$\min_{\boldsymbol{x} \in \boldsymbol{\mathcal{X}}} f(\boldsymbol{x}, \boldsymbol{z}) \tag{3}$$

$$\text{such that } g(\boldsymbol{x}, \boldsymbol{z}) \leq 0. \tag{4}$$

... therefore it happens that – highly optimized designs lead to high imperfection sensitivities and tend to loose **robustness**, or – the deterministic optimum is pushed to the boundaries of the feasible design space, which means that there is a **risk of failure** in case of system **imperfections/variations/noise**. In this case the system is less **reliable**.

toward (computer-based) optimization under uncertainty (OUU)

solving complex system design problem often faces inherent – physical **stochastic** phenomena, – **lack of knowledge**, – **modeling simplifications**, – variability of final design due to **manufacturing tolerance** etc.

$$\min_{\boldsymbol{x} \in \boldsymbol{\chi}} f(\boldsymbol{x}, \boldsymbol{z}) \tag{3}$$

$$\text{such that } g(\boldsymbol{x}, \boldsymbol{z}) \leq 0. \tag{4}$$

... therefore it happens that – highly optimized designs lead to high imperfection sensitivities and tend to loose **robustness**, or – the deterministic optimum is pushed to the boundaries of the feasible design space, which means that there is a **risk of failure** in case of system **imperfections/variations/noise**. In this case the system is less **reliable**.

Important :

there can be **uncertainties/noise** in x **or/and** z

Table of contents

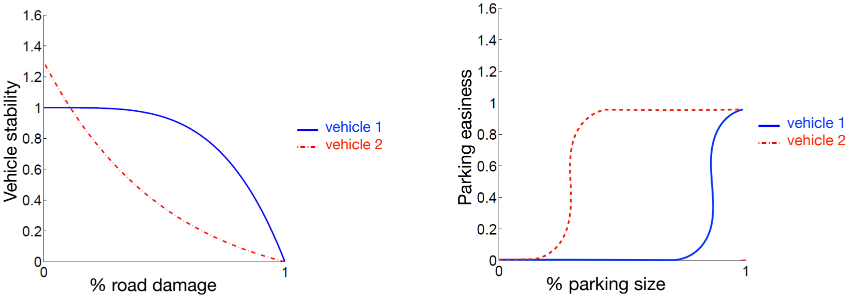
Introduction

concept of Robustness

A system is **robust** if its response remains “under control" (e.g. of satisfactory performance or bounded, etc) under **non-nominal** and even **random** conditions

Which vehicle is **more robust** ? which one is more **efficient/performant** ?... for which **objective**/purpose/task/metric ?

Robustness vs Performance



Which vehicle is **more robust** ? Is **robustness** always important ?

"Best" vehicle ? price, environmental footprint, durability, performance (Formule 1 / going on vacations / car pooling...)

concept of Reliability

A system is **reliable** if it is able to **perform**, **without failing as expected**, over certain time under **specified operating conditions**

"Don't worry about burning the calories — that's already been done!"

Performance : criteria defining when/how product **fails**

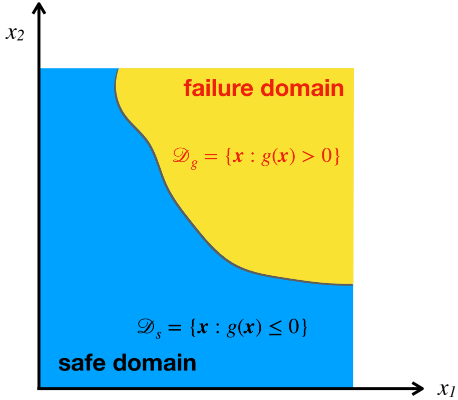
Period of time : **"product life"** varies a lot !

Are **operating conditions** always under control / fixed / stable / **well known** ?

Makes sense to relate reliability to a **probability** of **occurrence of failures**

Reliability and limit-state function

System behavior to optimize is binary in state space : its response may be separable in a **safe** $\mathcal{D}_s$ or a **failure** $\mathcal{D}_g$ region $\leftarrow$ there exists **constraints** on **admissible** solution



Given some parameters x **distribution**, we face a **classification** problem : is the system under **failure** and with **what probability** ?

To recap... 3 types of optimization under uncertainties problem

1. **robust optimization :** a design is **robust** if its performance is not (too) sensitive to inherent **variations/uncertainties** [Beyer and Sendhoff, 2007]

   $\rightarrow$ **uncertainties** on **objective** functions

2. **reliability-based design optimization (RBDO) :** a design is **reliable** if its performance targets are met in the presence of **variations/uncertainties**, minimizing its risk of **failure** [Ditlevsen and Madsen, 1996]

   $\rightarrow$ **uncertainties** on **constraint** functions

3. **... a mix of the previous two**

Some important points raised (or not !) in this course series

Optimization under uncertainties (OUU)

Is OUU only addressing a numerical problem ?

Differences in optimizing approaches to tackle scenario **with or without uncertainty**

What are the different kind of problems and corresponding **mathematical formulations** ?

Main difficulties and tools needed to optimize a system under uncertainties ?

Is the resolution becoming more challenging in case of **large number of parameters** ?

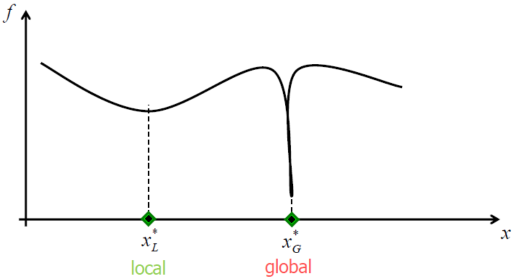
Advantages of addressing uncertainties in a **probabilistic framework** ?

Examples of recent methods : harnessing **metamodeling techniques**

Deal with **optimizations algorithms as black-boxes**...

Design & robustness : illustration on the cost function f

Let us consider a simple one-dimensional **quantity of interest** (**QoI**) represented by the output of the function f. What would be the **best optimal** value T of this function ?

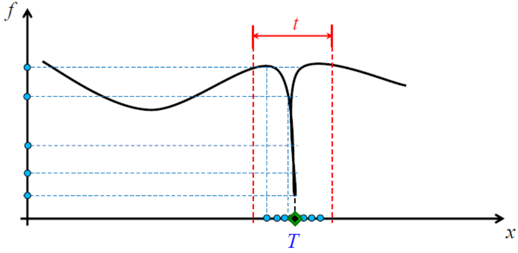


Best optimum (here minimum) ? x_L^* ou x_G^* ?

If **no noise or uncertainties**, then $T \leftarrow x_G^*$ is the best optimum

Design & robustness : illustration on the cost function f

Let us consider a simple one-dimensional **quantity of interest** (**QoI**) represented by the output of the function f. What would be the **best optimal** value T of this function ?

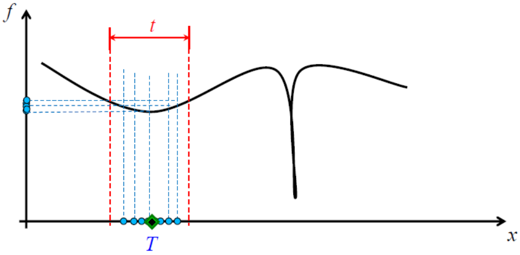


If there exists **some uncertainty related to** x : for instance x has some manufacturing tolerance, so there is some dispersion of its value due to the manufacturing process that is not perfect

Choosing x_G^* will most likely provide a **large performance variation** !

Design & robustness : illustration on the cost function f

Let us consider a simple one-dimensional **quantity of interest** (**QoI**) represented by the output of the function f. What would be the **best optimal** value T of this function ?

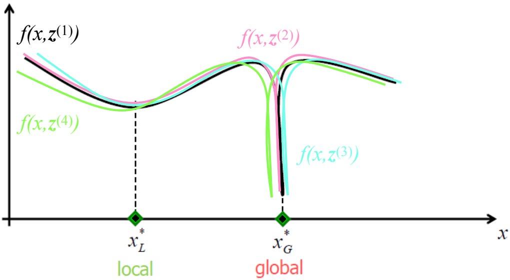


If there exists **some uncertainty related to** x : for instance x has some manufacturing tolerance, so there is some dispersion of its value due to the manufacturing process that is not perfect

Choosing x_L^* will provide a **smaller** performance variation : **more robust !**

Design & robustness : illustration on the cost function f

Let us consider a simple one-dimensional **quantity of interest** (**QoI**) represented by the output of the function f. What would be the **best optimal** value T of this function ?

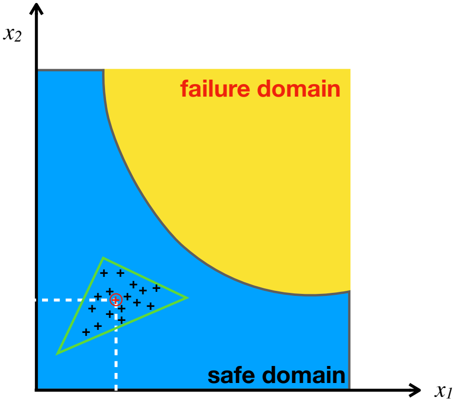


If there exists **some uncertainty related to some environmental parameter** z that **affects the dependence btw** f **and** x...

... choosing **most robust** T optimum is more difficult !

Design & reliability : illustration on the constraint function g

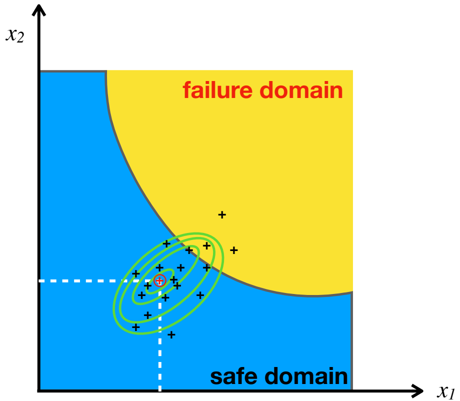
We consider a bivariate QoI depending on design parameters x. What is the impact of the position/**distribution** of x on the **reliable** ($\neq$ admissible) **design** ?



Distribution of x is **bounded** + far from boundary : **no risk of failure** for the system.

Design & reliability : illustration on the constraint function g

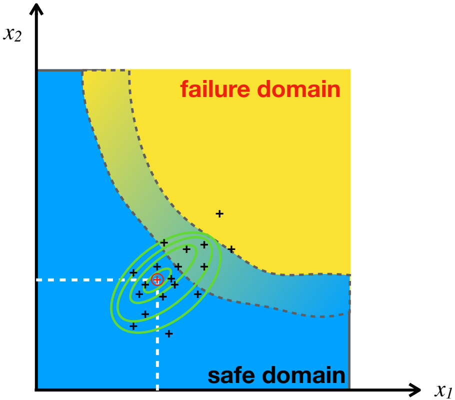
We consider a bivariate QoI depending on design parameters x. What is the impact of the position/**distribution** of x on the **reliable** ($\neq$ admissible) **design** ?



Distribution of x is **unbounded** (e.g. Gaussian) : **probability of failure** depends on **limit-state surface** : **not always** explicitly known

Design & reliability : illustration on the constraint function g

We consider a bivariate QoI depending on parameters $(\boldsymbol{x}, \boldsymbol{z})$. What is the impact of the position/**distribution** of $\boldsymbol{x}$ on the **reliable** ($\neq$ admissible) **design under uncertainty** ?



Distribution of $\boldsymbol{x}$ is **unbounded** (e.g. Gaussian) : **probability of failure** depends on **limit-state surface** : **not always** explicitly known... **may depend on uncertainty !**

Table of contents

Uncertainty sources (1)

They are everywhere ! !

Uncertainties affect simulation models and their working conditions and their effect **must be accounted for** when an optimization process is based on a numerical solver

Uncertainties are **many** and of **different nature** !

- **aleatoric/random** uncertainties (environment, initial/boundary conditions, physical phenomena, etc.)
- **epistemic**/lack of knowledge (model, etc.)

$$s(\boldsymbol{x}, \boldsymbol{z} + \boldsymbol{\delta}; \boldsymbol{q} + \boldsymbol{\beta}) + \boldsymbol{\varepsilon}(\boldsymbol{x}) \equiv y(\boldsymbol{x}, \boldsymbol{U}), \quad \boldsymbol{U} = (\boldsymbol{\delta}, \boldsymbol{\beta}, \boldsymbol{\varepsilon}),$$

Uncertainty sources (1)

They are everywhere ! !

Uncertainties affect simulation models and their working conditions and their effect **must be accounted for** when an optimization process is based on a numerical solver

Uncertainties are **many** and of **different nature** !

- **aleatoric/random** uncertainties (environment, initial/boundary conditions, physical phenomena, etc.)
- **epistemic**/lack of knowledge (model, etc.)

$$s(\boldsymbol{x}, \boldsymbol{z} + \boldsymbol{\delta}; \boldsymbol{q} + \boldsymbol{\beta}) + \boldsymbol{\varepsilon}(\boldsymbol{x}) \equiv y(\boldsymbol{x}, \boldsymbol{U}), \quad \boldsymbol{U} = (\boldsymbol{\delta}, \boldsymbol{\beta}, \boldsymbol{\varepsilon}),$$

- $\boldsymbol{s}$: represents the **numerical solver**
- ✓ $\boldsymbol{x}$: **design** parameters
- $\boldsymbol{z}$: environmental parameters, $\boldsymbol{q}$: solver internal parameters, $\boldsymbol{\varepsilon}$: outputs uncertainty (model discrepancy)
- ✓ All uncertainties sources may be lumped into the **random vector** $\boldsymbol{U}$, but in this class $\boldsymbol{U}$ will mainly account for **environmental uncertainties**

Uncertainty sources (2)

They are everywhere ! !

Uncertainties affect simulation models and their working conditions and their effect **must be accounted for** when an optimization process is based on a numerical solver

Uncertainties are **many** and of **different nature** !

- **aleatoric/random** uncertainties (environment, initial/boundary conditions, physical phenomena, etc.)
- **design variables are sometimes noisy** in the sense that, even if we optimize their **nominal**/target values x, their true values are *in fine* random due to **manufacturing tolerance**

$$s(\boldsymbol{x}+\boldsymbol{\gamma},\boldsymbol{z}+\boldsymbol{\delta};q+\beta)+\varepsilon(x)\equiv y(\boldsymbol{D}(\boldsymbol{x}),\boldsymbol{U}),\quad \boldsymbol{U}\equiv(\boldsymbol{\delta},\beta,\varepsilon),$$

They are everywhere ! !

Uncertainties affect simulation models and their working conditions and their effect **must be accounted for** when an optimization process is based on a numerical solver

Uncertainties are **many** and of **different nature** !

aleatoric/random uncertainties (environment, initial/boundary conditions, physical phenomena, etc.)

design variables are sometimes noisy in the sense that, even if we optimize their **nominal**/target values x, their true values are *in fine* random due to **manufacturing tolerance**

$$s(\boldsymbol{x}+\boldsymbol{\gamma}, \boldsymbol{z}+\boldsymbol{\delta}; q+\beta) + \varepsilon(\boldsymbol{x}) \equiv y(\boldsymbol{D}(\boldsymbol{x}), \boldsymbol{U}), \quad \boldsymbol{U} \equiv (\boldsymbol{\delta}, \beta, \varepsilon),$$

- ✓ $\boldsymbol{D}(\boldsymbol{x})$: **noisy** design parameters with **known** random **"dispersion"** (e.g. probability distribution function), depend on **hyperparameters** $\boldsymbol{x}$

  $\boldsymbol{x}$: **nominal** design parameters **to be optimized**, e.g. **mean values**

  $\boldsymbol{z}$: environmental parameters
- ✓ All remaining uncertainties sources may be lumped into the **random vector** $\boldsymbol{U}$

Probabilistic formulation

Standard model equipped with a **probability space** $(\Omega, A, \mathbb{P})$:

$$s(\boldsymbol{D}(\boldsymbol{x}), \boldsymbol{z} + \boldsymbol{\delta}) = y(\boldsymbol{D}(\boldsymbol{x}), \boldsymbol{U}) \text{ with } \quad \boldsymbol{U} \sim f_{\boldsymbol{U}} \text{ and } \boldsymbol{D}|x \sim f_{\boldsymbol{D}|\boldsymbol{x}},$$

where y is the solution of the numerical solver s, and the distributions are **known**.

Double $(\boldsymbol{D}(\boldsymbol{x}), \boldsymbol{U})$ **parameterization** ("augmented space" or "hybrid space", cf. [Beyer and Sendhoff, 2007], [Pujol et al., 2009]) :

- $\boldsymbol{x} \in \boldsymbol{\chi} \subset \mathbb{R}^d$: vector of (deterministic) **design** parameters (+ soft constraints)
- $\boldsymbol{U} : \Omega \mapsto \mathcal{A} \subset \mathbb{R}^n$ vector of **random variables** of known joint distribution

Generic optimization setup

$$
\begin{aligned}
\boldsymbol{x}^{\star} &= \arg\min_{\boldsymbol{x}\in\boldsymbol{\mathcal{X}}} f(\boldsymbol{x}), \\
\text{such that } g_i(\boldsymbol{x}) &\leq 0 \text{ for } i \in \{1, \ldots, m\} \\
\boldsymbol{x}_{\text{min}} \leq \boldsymbol{x} &\leq \boldsymbol{x}_{\text{max}}
\end{aligned} \tag{5}
$$

Generic optimization setup

$$\boldsymbol{x}^\star = \arg\min_{\boldsymbol{x}\in\boldsymbol{\mathcal{X}}} f(\boldsymbol{x}, \boldsymbol{U}), \tag{6}$$

$$\text{such that } g_i(\boldsymbol{x}, \boldsymbol{U}) \leq 0 \text{ for } i \in \{1, \dots, m\} \tag{7}$$

$$h_j(\boldsymbol{x}) \leq 0, \quad \text{for } j \in \{m+1, \dots, M\} \tag{8}$$

Generic optimization setup

$$\boldsymbol{x}^{\star} = \arg\min_{\boldsymbol{x} \in \boldsymbol{\mathcal{X}}} f(\boldsymbol{x}, \boldsymbol{U}), \tag{6}$$

$$\text{such that } g_i(\boldsymbol{x}, \boldsymbol{U}) \leq 0 \text{ for } i \in \{1, \dots, m\} \tag{7}$$

$$h_j(\boldsymbol{x}) \leq 0, \quad \text{for } j \in \{m+1, \dots, M\} \tag{8}$$

$f(y(\boldsymbol{x}, \boldsymbol{U})) \equiv f(\boldsymbol{x}, \boldsymbol{U}) : \mathbb{R}^d \times \mathbb{R}^n \to \mathbb{R}$: **function to be optimized** – sometimes called **cost** or **objective** function – under various constraints :

$g_i(y(\boldsymbol{x}, \boldsymbol{U})) \equiv g_i(\boldsymbol{x}, \boldsymbol{U}) : \mathbb{R}^d \times \mathbb{R}^n \to \mathbb{R} : i = 1 \dots m$ (hard) **constraints** : limit-state functions

$h_j(\boldsymbol{x}) \leq 0, j = 1 \dots s$ (soft) **constraints** : fcts bounding the design space

Generic optimization setup

$$\boldsymbol{x}^\star = \arg\min_{\boldsymbol{x}\in\boldsymbol{\mathcal{X}}} f(\boldsymbol{x},\boldsymbol{U}), \tag{6}$$

$$\text{such that } g_i(\boldsymbol{x},\boldsymbol{U}) \le 0 \text{ for } i \in \{1,\ldots,m\} \tag{7}$$

$$h_j(\boldsymbol{x}) \le 0, \quad \text{for } j \in \{m+1,\ldots,M\} \tag{8}$$

$f(y(\boldsymbol{x},\boldsymbol{U})) \equiv f(\boldsymbol{x},\boldsymbol{U}) : \mathbb{R}^d \times \mathbb{R}^n \to \mathbb{R}$: **function to be optimized** – sometimes called **cost** or **objective** function – under various constraints :

$g_i(y(\boldsymbol{x},\boldsymbol{U})) \equiv g_i(\boldsymbol{x},\boldsymbol{U}) : \mathbb{R}^d \times \mathbb{R}^n \to \mathbb{R} : i = 1\ldots m$ (hard) **constraints** : limit-state functions

$h_j(\boldsymbol{x}) \le 0, j = 1\ldots s$ (soft) **constraints** : fcts bounding the design space

$\boldsymbol{U} : \Omega \mapsto \mathcal{A} \in \mathbb{R}^d$ vector of random variables... written like that : **problem is not well-posed**

$\Rightarrow y(\boldsymbol{x},\boldsymbol{U})$, $f(\boldsymbol{x},\boldsymbol{U})$, $g_i(\boldsymbol{x},\boldsymbol{U})$ are **dependent random quantities** **!**

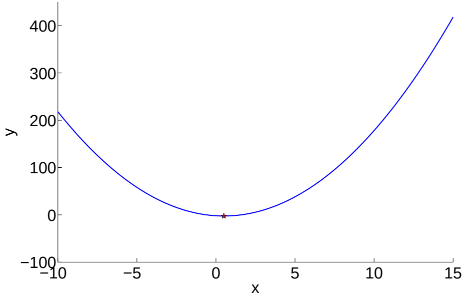
Effect of uncertainties on ROBUST optimization

To begin we consider a simple **unconstrained** optimization problem

1D nonlinear function with Gaussian noises

Example : $f(x; \boldsymbol{z}) = y(x; \boldsymbol{z}) = z_1 x^2 + z_2 x + z_3$, $z_1 \sim \mathcal{N}(2, 1)$, $z_2 \sim \mathcal{N}(-2, 1)$, $z_3 \sim \mathcal{N}(-2, 1)$, $\boldsymbol{z} \equiv (z_1, z_2, z_3)$ independently distributed, $x \in [-10, 15]$

$$x^\star = \arg \min_{x \in [-10,15]} y(x; \boldsymbol{z} = \boldsymbol{z}(\omega))$$



Deterministic optimization

Effect of uncertainties on ROBUST optimization

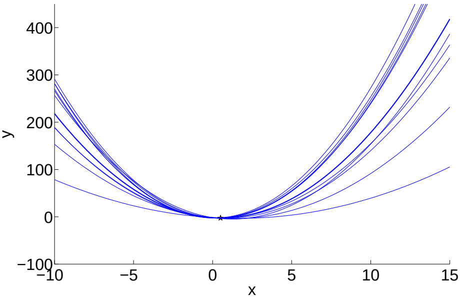
To begin we consider a simple **unconstrained** optimization problem

1D nonlinear function with Gaussian noises

Example : $f(x; \boldsymbol{z}) = y(x; \boldsymbol{z}) = z_1 x^2 + z_2 x + z_3$, $z_1 \sim \mathcal{N}(2, 1)$, $z_2 \sim \mathcal{N}(-2, 1)$, $z_3 \sim \mathcal{N}(-2, 1)$, $\boldsymbol{z} \equiv (z_1, z_2, z_3)$ independently distributed, $x \in [-10, 15]$

$$x^\star = \arg \min_{x \in [-10,15]} y(x; \boldsymbol{z} = \boldsymbol{z}(\omega))$$

With uncertainty, few samples of $y(x; \boldsymbol{z})$



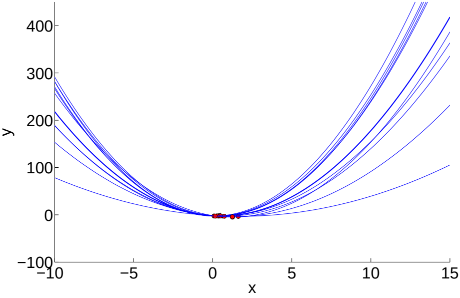
Effect of uncertainties on ROBUST optimization

To begin we consider a simple **unconstrained** optimization problem

1D nonlinear function with Gaussian noises

Example : $f(x;\boldsymbol{z}) = y(x;\boldsymbol{z}) = z_1 x^2 + z_2 x + z_3$, $z_1 \sim \mathcal{N}(2,1)$, $z_2 \sim \mathcal{N}(-2,1)$, $z_3 \sim \mathcal{N}(-2,1)$, $\boldsymbol{z} \equiv (z_1, z_2, z_3)$ independently distributed, $x \in [-10, 15]$

$$x^\star = \arg \min_{x\in[-10,15]} y(x;\boldsymbol{z} = \boldsymbol{z}(\omega))$$



No unique optimum

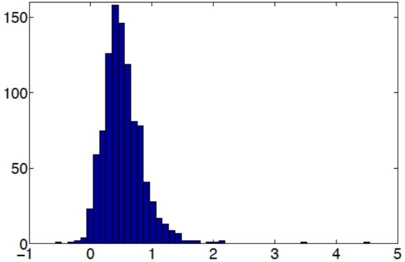
Effect of uncertainties on ROBUST optimization

To begin we consider a simple **unconstrained** optimization problem

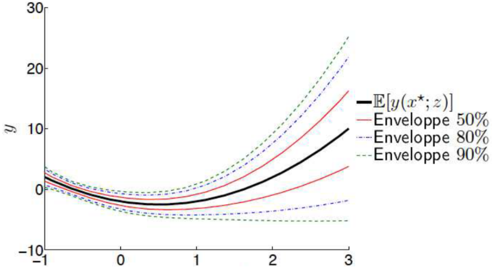
1D nonlinear function with Gaussian noises

Example : $f(x;\boldsymbol{z}) = y(x;\boldsymbol{z}) = z_1 x^2 + z_2 x + z_3$, $z_1 \sim \mathcal{N}(2,1)$, $z_2 \sim \mathcal{N}(-2,1)$, $z_3 \sim \mathcal{N}(-2,1)$, $\boldsymbol{z} \equiv (z_1, z_2, z_3)$ independently distributed, $x \in [-10, 15]$

$$x^\star = \arg \min_{x\in[-10,15]} y(x;\boldsymbol{z} = \boldsymbol{z}(\omega))$$

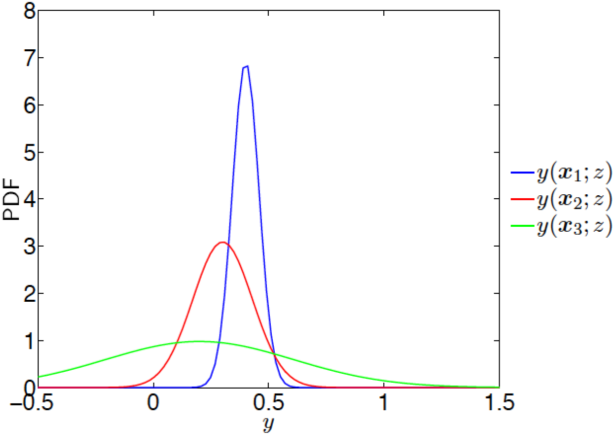


(a) $x^\star$ histogram : the optimum follows a random distribution (not normal !) due to z noise



(b) Variability of y given $x^\star$ (horizontal axis)

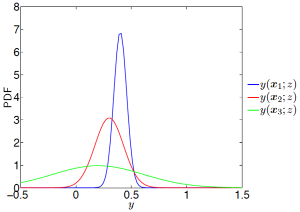
How do we rank the various optimum ?



⇒ How to choose among $x_1^\star$, $x_2^\star$ et $x_3^\star$?

⇒ what is the meaning of "$x_i^\star$ is better than $x_j^\star$" ?

Idea : "remove" the uncertainty (mathematically) with statistical (risk) measures
this way the function does not explicitly depends on the uncertainty anymore
(marginalized, implicit dependance through $f_{\boldsymbol{z}}$)



$$\boldsymbol{x}^\star = \arg\min_{\boldsymbol{x}\in\boldsymbol{\mathcal{X}}} \mathbb{L}\big(y(x, \boldsymbol{z})\big),$$

where $\boldsymbol{z}$ is the random quantity.

What choices of the measure $\mathbb{L}$ can we choose to express that "$\boldsymbol{x}^\star$ is better (in some sense) than $\boldsymbol{x}'^\star$"?

Making use of statistical measures

Different statistical (risk) measures for the objective function

"worst-case" approach : $\max_{\boldsymbol{z}}[y(x;\boldsymbol{z})] \leq \max_{\boldsymbol{z}}[y(x';\boldsymbol{z})]$.

mean approach : $\mathbb{E}[y(x;\boldsymbol{z})] \leq \mathbb{E}[y(x';\boldsymbol{z})]$, with $\mathbb{E}[y(\cdot;\boldsymbol{z})] = \int_{\mathcal{A}} y(\cdot,\boldsymbol{z}) f_{\boldsymbol{Z}}(\boldsymbol{z}) d\boldsymbol{z}$

variance approach : $\mathbb{V}[y(x;\boldsymbol{z})] \leq \mathbb{V}[y(x';\boldsymbol{z})]$, with $\mathbb{V}[y(\cdot;\boldsymbol{z})] = \mathbb{E}[(y(\cdot;\boldsymbol{z}) - \mathbb{E}[y(\cdot;\boldsymbol{z})])^2]$

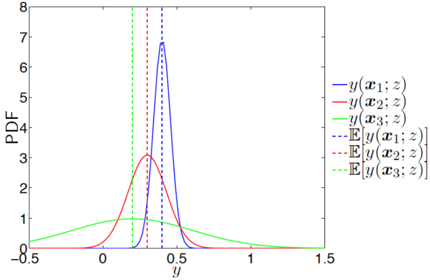
quantile approach : $\mathbb{Q}_\alpha[y(x;\boldsymbol{z})] \leq \mathbb{Q}_\alpha[y(x';\boldsymbol{z})]$, $\mathbb{P}(y(x;\boldsymbol{z}) \leq \mathbb{Q}_\alpha[y(x;\boldsymbol{z})]) = \alpha$, with $\mathbb{Q}_\alpha[y(\cdot;\boldsymbol{z})]) = \inf\{q \in \mathbb{R} : \mathbb{P}\left[y(\cdot;\boldsymbol{z}) \leq q\right] \geq \alpha\}$

conditional value-at-risk approach :
$\mathbb{E}[y(x;\boldsymbol{z}) \mid y(x;\boldsymbol{z}) \geq \mathbb{Q}_\alpha[y(x;\boldsymbol{z})]] \leq \mathbb{E}[y(x';\boldsymbol{z}) \mid y(x';\boldsymbol{z}) \geq \mathbb{Q}_\alpha[y(x';\boldsymbol{z})]]$

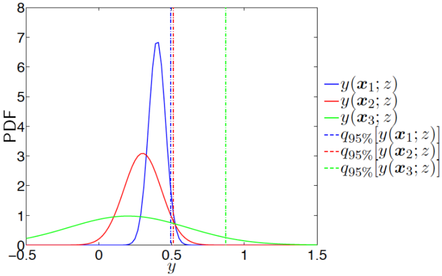
a **multi-objective criteria** approach $(\mathbb{E}[y(x;\boldsymbol{z})], \mathbb{V}[y(x;\boldsymbol{z})])$

... **108(!) different statistical measures** in [Göhler et al., Journal of Mechanical Design, 2016], see also [Yao et al. 2011 ; Lelievre et al. 2016]

The choice of the measure completely changes the ordering of the design !



(c) Mean : $\boldsymbol{x}_3 > \boldsymbol{x}_2 > \boldsymbol{x}_1$

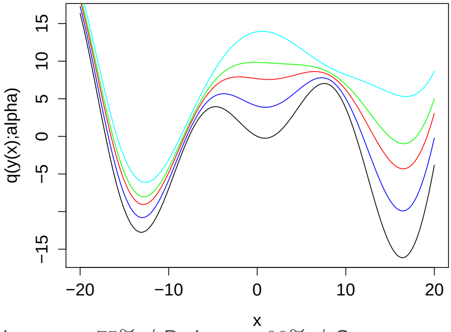


(d) Quantile 95% : $\boldsymbol{x}_1 > \boldsymbol{x}_2 > \boldsymbol{x}_3$

The optimum varies depending on the level of uncertainty

Example 1 : $y(x;z) = y(x) + z(x)$, chosen $\mathbb{L} \leftrightarrow \mathbb{Q}_\alpha$, type of uncertainty $z \leftrightarrow$ model error

$$x^\star = \arg \min_{x \in [-20,20]} \mathbb{Q}_\alpha \left[y(x;z) \right],$$



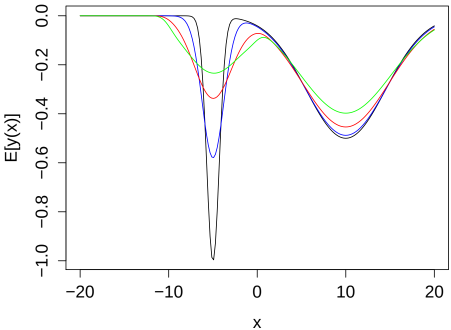
Black : $\alpha = 50\%$ / Blue : $\alpha = 75\%$ / Red : $\alpha = 90\%$ / Green : $\alpha = 95\%$ / Cyan : $\alpha = 99\%$.

Impact of uncertainty level

The optimum varies depending on the level of uncertainty

Example 2 : $y(x,z) = y(x+z)$, chosen $\mathbb{L} \leftrightarrow \mathbb{E}$, $z \leftrightarrow$ manufacturing tolerance

$$x^\star = \arg \min_{x \in [-20,20]} \mathbb{E}\left[y(x+z)\right], \quad z \sim \mathcal{N}(0,\sigma^2)$$

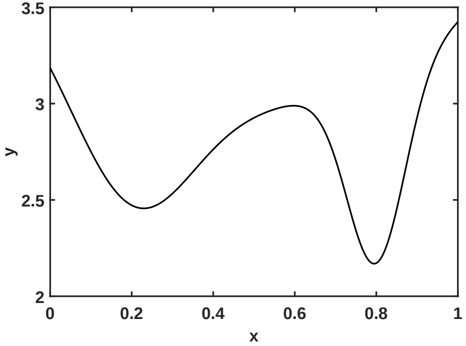


Black : $\sigma = 0$ / Blue : $\sigma = 1$ / Red : $\sigma = 2$ / Green : $\sigma = 3$.

The optimum varies a lot depending on the measure

Example 3 : $y(x,z) = y(x+z)$, chosen $\mathbb{L} \leftrightarrow ???$, $z \leftrightarrow$ manufacturing tolerance

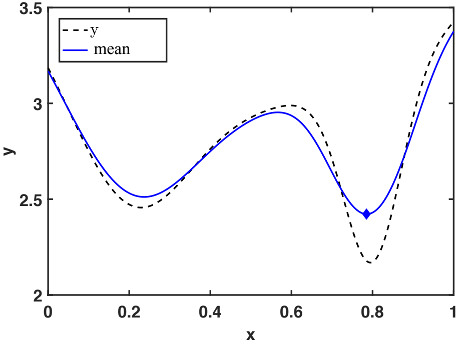
$$x^\star = \arg \min_{x \in [-20,20]} \mathbb{L}\left[y(x+z)\right], \quad z \sim \mathcal{U}(-0.1, 0.1)$$



The optimum varies a lot depending on the measure

Example 3 : $y(x,z) = y(x+z)$, chosen $\mathbb{L} \leftrightarrow$???, $z \leftrightarrow$ manufacturing tolerance

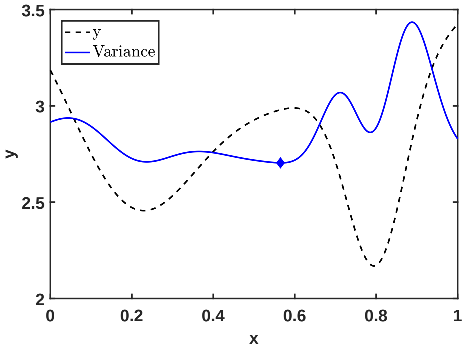
$$x^\star = \arg \min_{x \in [-20,20]} \mathbb{L}\left[y(x+z)\right], \quad z \sim \mathcal{U}(-0.1, 0.1)$$



The optimum varies a lot depending on the measure

Example 3 : $y(x,z) = y(x+z)$, chosen $\mathbb{L} \leftrightarrow$???, $z \leftrightarrow$ manufacturing tolerance

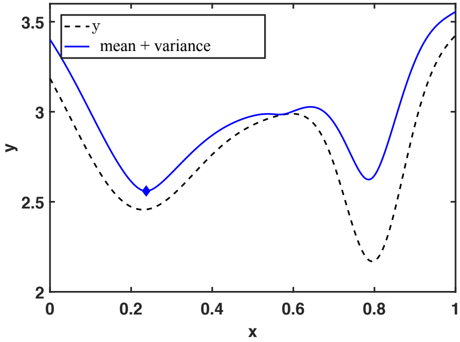
$$x^\star = \arg \min_{x \in [-20,20]} \mathbb{L}\left[y(x+z)\right], \quad z \sim \mathcal{U}(-0.1, 0.1)$$



The optimum varies a lot depending on the measure

Example 3 : $y(x,z) = y(x+z)$, chosen $\mathbb{L} \leftrightarrow$???, $z \leftrightarrow$ manufacturing tolerance

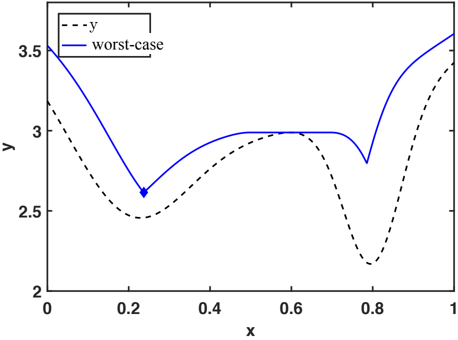
$$x^\star = \arg \min_{x \in [-20,20]} \mathbb{L}\left[y(x+z)\right], \quad z \sim \mathcal{U}(-0.1, 0.1)$$



The optimum varies a lot depending on the measure

Example 3 : $y(x,z) = y(x+z)$, chosen $\mathbb{L} \leftrightarrow$???, $z \leftrightarrow$ manufacturing tolerance

$$x^{\star} = \arg \min_{x \in [-20,20]} \mathbb{L}\left[y(x+z)\right], \quad z \sim \mathcal{U}(-0.1, 0.1)$$

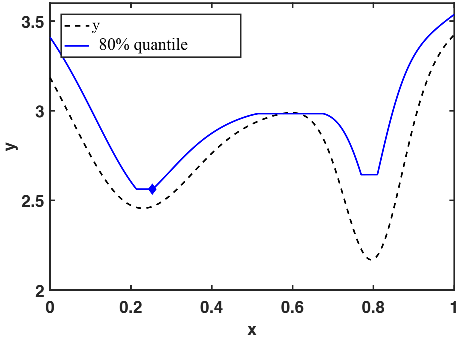


Impact of statistical measures choice

The optimum varies a lot depending on the measure

Example 3 : $y(x,z) = y(x+z)$, chosen $\mathbb{L} \leftrightarrow ???$, $z \leftrightarrow$ manufacturing tolerance

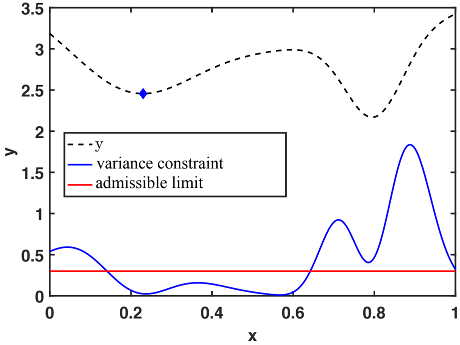
$$x^\star = \arg\min_{x\in[-20,20]} \mathbb{L}\left[y(x+z)\right], \quad z \sim \mathcal{U}(-0.1, 0.1)$$



The optimum varies a lot depending on the measure

Example 3 : $y(x,z) = y(x+z)$, chosen $\mathbb{L} \leftrightarrow$???, $z \leftrightarrow$ manufacturing tolerance

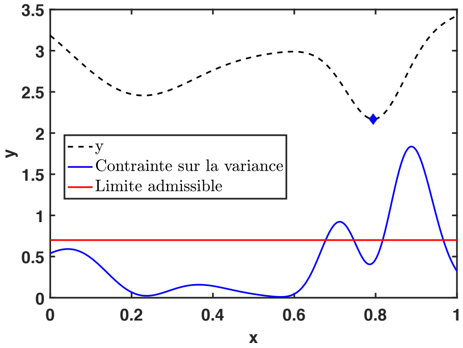
$$x^\star = \arg \min_{x\in[-20,20]} \mathbb{L}\left[y(x+z)\right], \quad z \sim \mathcal{U}(-0.1, 0.1)$$



The optimum varies a lot depending on the measure

Example 3 : $y(x,z) = y(x+z)$, chosen $\mathbb{L} \leftrightarrow$???, $z \leftrightarrow$ manufacturing tolerance

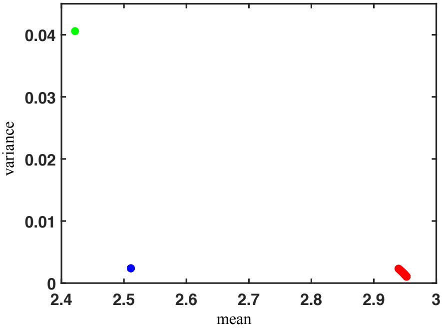
$$x^\star = \arg \min_{x \in [-20,20]} \mathbb{L}\left[y(x+z)\right], \quad z \sim \mathcal{U}(-0.1, 0.1)$$

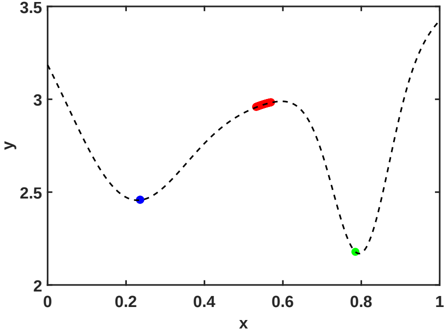


The optimum varies a lot depending on the measure

Example 3 : $y(x,z) = y(x+z)$, chosen $\mathbb{L} \leftrightarrow$???, $z \leftrightarrow$ manufacturing tolerance

$$x^\star = \arg \min_{x \in [-20,20]} \mathbb{L}\left[y(x+z)\right], \quad z \sim \mathcal{U}(-0.1, 0.1)$$





What about more generic OUU problem formulation

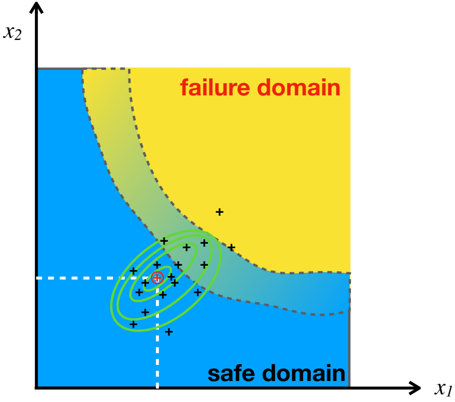
This time we start from a **constrained** deterministic optimization problem and the OUU formulation becomes :

$$\begin{aligned} \boldsymbol{x}^\star = \arg\min_{\boldsymbol{x}\in\boldsymbol{\mathcal{X}}} \mathbb{L}\left[f(\boldsymbol{x},\boldsymbol{U})\right], \\ \text{such that } \mathbb{K}_i\left[g_i(\boldsymbol{x},\boldsymbol{U})\right] \leq 0 \text{ for } i \in \{1,\dots,m\} \\ \boldsymbol{x}_{\mathsf{min}} \leq \boldsymbol{x} \leq \boldsymbol{x}_{\mathsf{max}} \end{aligned} \tag{9}$$

Again, **statistical measures** must be introduced in $\mathbb{K}$, to deal with the **constraints** this time.

Design & reliability : illustration on the constraint function

We consider a bivariate QoI depending on parameters $(x, \boldsymbol{U})$. What is the impact of the position/**distribution** of x on the **reliable** ($\neq$ admissible) **design under uncertainty** ?



Distribution of x is **unbounded** (e.g. Gaussian) : **probability of failure** depends on **limit-state surface** : **not always** explicitly known... **may depend on uncertainty !**

RBDO problems formulation (1)

$$
\begin{aligned}
\boldsymbol{x}^\star = \arg\min_{\boldsymbol{x}\in\boldsymbol{\mathcal{X}}} f(\boldsymbol{x},\overline{\boldsymbol{U}}) \\
\text{such that } \mathbb{K}_i\left[g_i(\boldsymbol{D}(\boldsymbol{x}),\overline{\boldsymbol{U}})\right] \leq 0 \text{ for } i \in \{1,\ldots,m\} \\
\boldsymbol{x}_{\text{min}} \leq \boldsymbol{x} \leq \boldsymbol{x}_{\text{max}}
\end{aligned}
$$

with

$$
\begin{aligned}
\mathbb{K}\left[g_i(\boldsymbol{D}(\boldsymbol{x}),\overline{\boldsymbol{U}})\right] &\equiv \mathbb{P}\left[g_i(\boldsymbol{D}(\boldsymbol{x}),\overline{\boldsymbol{U}}) > 0\right] - P_f, \\
&= \int_{g_i(\boldsymbol{D}(\boldsymbol{x}),\overline{\boldsymbol{U}})>0} f_{\boldsymbol{D}|\boldsymbol{x}}(\boldsymbol{D}|\boldsymbol{x})d\boldsymbol{D}|\boldsymbol{x} - P_f \\
&= \int_{\mathbb{R}^d} \mathbb{1}_{g_i(\boldsymbol{D}(\boldsymbol{x}),\overline{\boldsymbol{U}})>0} f_{\boldsymbol{D}|\boldsymbol{x}}(\boldsymbol{D}|\boldsymbol{x})d\boldsymbol{D}|\boldsymbol{x} - P_f,
\end{aligned} \tag{10}
$$

so as to denote it will respect a **given tolerance** on a **probability of failure (PoF)** P_f.

RBDO problems formulation (2)

$$
\begin{aligned}
\boldsymbol{x}^\star &= \arg\min_{\boldsymbol{x}\in\boldsymbol{\mathcal{X}}} f(\boldsymbol{x}, \overline{\boldsymbol{U}}) \\
\text{such that } \mathbb{K}_i\left[g_i(\boldsymbol{x}, \boldsymbol{U})\right] &\leq 0 \text{ for } i \in \{1, \ldots, m\} \\
\boldsymbol{x}_{\text{min}} &\leq \boldsymbol{x} \leq \boldsymbol{x}_{\text{max}}
\end{aligned}
$$

with

$$
\begin{aligned}
\mathbb{K}\left[g_i(\boldsymbol{x}, \boldsymbol{U})\right] &\equiv \mathbb{P}\left[g_i(\boldsymbol{x}, \boldsymbol{U}) > 0\right] - P_f, \\
&= \int_{g_i(\boldsymbol{x}, \boldsymbol{U})>0} f_{\boldsymbol{U}}(\boldsymbol{U}) d\boldsymbol{U} - P_f = \int_{\mathbb{R}^n} \mathbb{1}_{g_i(\boldsymbol{x}, \boldsymbol{U})>0} f_{\boldsymbol{U}}(\boldsymbol{U}) d\boldsymbol{U} - P_f,
\end{aligned} \tag{11}
$$

so as to denote it will respect a **given tolerance** on a **probability of failure** (**PoF**) P_f.

Remark : **reliability** problem was already involving probabilities... additional uncertainties make the formulation $\rightarrow$ not as straightforward as in the robust case !

Solution of RBDO problem relies on **estimation of PoF** for different values of design parameters $\rightarrow$ **cost**$^{++}$

Relying on computation of probability or quantile ?

Equivalence between RBDO and quantile-based formulation

Standard reliable formulation via PoF defined for particular constraint g_i :

$$\mathbb{K}\left[g_i(\boldsymbol{x},\boldsymbol{U})\right] \equiv \mathbb{P}\left[g_i(\boldsymbol{x},\boldsymbol{U}) > 0\right] \leq \overline{P_f^{g_i}} \leftrightarrow P_f(\ll 1)$$

for a given design : $P_f^{g_i}(\boldsymbol{x}) = \mathbb{P}\left[g_i(\boldsymbol{U}|\boldsymbol{x}) > 0\right] = \int_{g_i(\boldsymbol{U}|\boldsymbol{x})>0} f_{\boldsymbol{U}|\boldsymbol{X}}(\boldsymbol{U}|\boldsymbol{x}) d\boldsymbol{U}$

Relying on computation of probability or quantile ?

Equivalence between RBDO and quantile-based formulation

Standard reliable formulation via PoF defined for particular constraint g_i :

$$\mathbb{K}\left[g_i(\boldsymbol{x},\boldsymbol{U})\right] \equiv \mathbb{P}\left[g_i(\boldsymbol{x},\boldsymbol{U}) > 0\right] \leq \overline{P_f^{g_i}} \leftrightarrow P_f(\ll 1)$$

$$\text{for a given design : } P_f^{g_i}(\boldsymbol{x}) = \mathbb{P}\left[g_i(\boldsymbol{U}|\boldsymbol{x}) > 0\right] = \int_{g_i(\boldsymbol{U}|\boldsymbol{x})>0} f_{\boldsymbol{U}|\boldsymbol{X}}(\boldsymbol{U}|\boldsymbol{x}) d\boldsymbol{U}$$

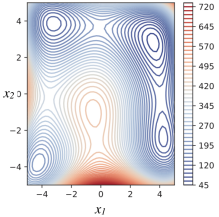
Equivalence between RBDO and quantile-based formulation

$$\mathbb{P}\left[g_i(\boldsymbol{x},\boldsymbol{U}) > 0\right] \leq P_f \quad \Longleftrightarrow \quad \mathbb{Q}_\alpha\left[\boldsymbol{x}; g_i(\boldsymbol{x},\boldsymbol{U})\right] \leq 0 \text{ (put otherwise)}$$

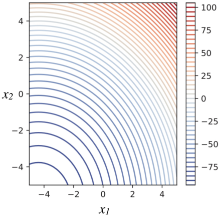
- quantile approach : easier coupling with already existing deterministic design process
- outer loop explores the design space while inner loop simply computes constraints quantiles

2D nonlinear objective function under constraints [Himmelblau 1972, Balesdent et al. 2020] **and subject to Gaussian noise**

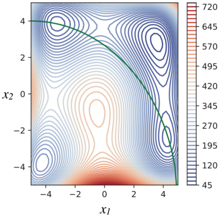
$$\begin{aligned} \text{min} \quad & (x_1^2 + x_2 - 15 - 0.5U^2)^2 + (x_1 + x_2^2 - 11 + U)^2 + 10(5 - x_1) + 10(5 - x_2), \\ \text{such that :} \quad & \left((x_1 + 5 - 0.1(U - 2.5)^2)^2 + (x_2 + 6 - 2U)^2 - 100\right) \leq 0, \\ \text{with} \quad & U \sim \mathcal{N}(2, 1.5) \ \text{ and } \ \boldsymbol{x} \in [-5, 5] \end{aligned}$$



objective function $f(\cdot)$



constraint $g(\cdot)$



Det. obj. fct + limite state $g(\boldsymbol{x}) = 0$

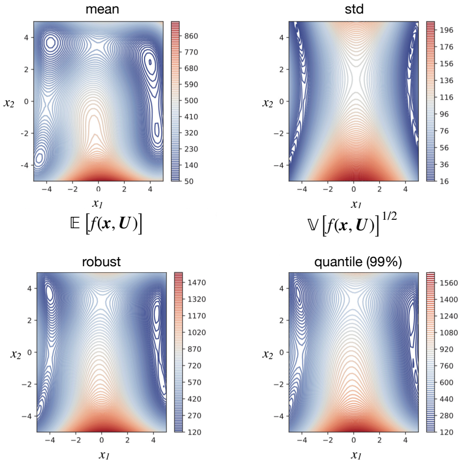
2D nonlinear objective function under constraints [Himmelblau 1972, Balesdent et al. 2020] **and subject to Gaussian noise**

Different measures of uncertainty applied to the **objective function**.

E.g. a **robust** measure with a multi-objective optimization $\rightarrow$ as a **single-objective** one [Papadrakakis 2005] :

$$\mathbb{L} \equiv \mathbb{E}\left[f(\boldsymbol{x},\boldsymbol{U})\right] + k\mathbb{V}\left[f(\boldsymbol{x},\boldsymbol{U})\right]^{1/2},$$

the larger $k > 0$, the more **conservative** the design.



$\mathbb{E}\left[f(\boldsymbol{x},\boldsymbol{U})\right]$ $\mathbb{V}\left[f(\boldsymbol{x},\boldsymbol{U})\right]^{1/2}$

$\mathbb{E}\left[f(\boldsymbol{x},\boldsymbol{U})\right] + 3\mathbb{V}\left[f(\boldsymbol{x},\boldsymbol{U})\right]^{1/2}$ $\mathbb{Q}_{99\%}\left[f(\boldsymbol{x},\boldsymbol{U})\right]$

More generic OUU problem example

2D nonlinear objective function under constraints [Himmelblau 1972, Balesdent et al. 2020] **and subject to Gaussian noise**

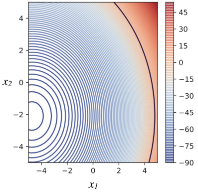
Different measures of uncertainty applied to the **constraint function** (and corresponding limit state).

similar compound **robust** measure can also be applied on the constraint $g_i(\cdot)$:

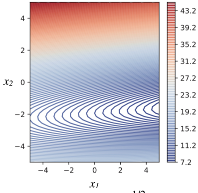
$$\mathbb{K} \equiv \mathbb{E}\left[g_i(\boldsymbol{x},\boldsymbol{U})\right] + k_i \mathbb{V}\left[g_i(\boldsymbol{x},\boldsymbol{U})\right]^{1/2},$$

the magnitude of $k_i > 0$ is more difficult to interpret depending on the localization of the high variance region.

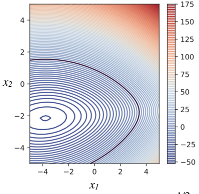
Or we may rely on the **reliability measure** : $\mathbb{P}\left[g(\boldsymbol{x},\boldsymbol{U}) > 0\right] \leq P_f$



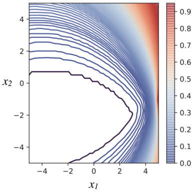
$\mathbb{E}\left[g(\boldsymbol{x},\boldsymbol{U})\right]$



$\mathbb{V}\left[g(\boldsymbol{x},\boldsymbol{U})\right]^{1/2}$



$\mathbb{E}\left[g(\boldsymbol{x},\boldsymbol{U})\right] + 3\mathbb{V}\left[g(\boldsymbol{x},\boldsymbol{U})\right]^{1/2}$



$\mathbb{P}\left[g(\boldsymbol{x},\boldsymbol{U}) \leq 0\right] \leq 0.01$

To recap...

1. Depending on the different measures of uncertainty the designer has to choose **most representative uncertainty measures** $(\mathbb{L}, \mathbb{K})$ for his design under uncertainties problem.
   The choice of a given uncertainty measure for the objective function and constraints leads to a **particular formulation** of the design problem.

   This could be (roughly) : 1. a **ROBUST**, 2. a **reliability-based RBDO** or 3. a **robust & reliability-based** formulation

ROBUSTNESS

RELIABILITY

	No objective function	Objective function with X, P deterministic	Objective function with X, P uncertain
No constraint function	X	determinitic optimization without constraints	Optimization of the robutness
Constraint function with X, P deterministic	Sizing / dimensionning	determinitic optimization under constraints	Optimization of the robutness under determinitic constraint
Constraint function with X, P uncertain	Reliability	Reliability-based design Optimization (RBDO)	Optimization of the robutness under uncertain constraint

FIGURE: [Lelievre et al. 2016]

OUU roadmap

To recap...

1. Depending on the different measures of uncertainty the designer has to choose **most representative uncertainty measures** $(\mathbb{L}, \mathbb{K})$ for his design under uncertainties problem.
   The choice of a given uncertainty measure for the objective function and constraints leads to a **particular formulation** of the design problem.

   This could be (roughly) : 1. a **ROBUST**, 2. a **reliability-based RBDO** or 3. a **robust & reliability-based** formulation

ROBUSTNESS

RELIABILITY

	No objective function	Objective function with X, P deterministic	Objective function with X, P uncertain
No constraint function	X	determinitic optimization without constraints	Optimization of the robustness
Constraint function with X, P deterministic	Sizing / dimensionning	determinitic optimization under constraints	Optimization of the robustness under determinitic constraint
Constraint function with X, P uncertain	Reliability	Reliability-based design Optimization (RBDO)	Optimization of the robustness under uncertain constraint

FIGURE: [Lelievre et al. 2016]

To recap...

1. Depending on the different measures of uncertainty the designer has to choose **most representative uncertainty measures** $(\mathbb{L}, \mathbb{K})$ for his design under uncertainties problem.
   The choice of a given uncertainty measure for the objective function and constraints leads to a **particular formulation** of the design problem.

   This could be (roughly) : 1. a **ROBUST**, 2. a **reliability-based RBDO** or 3. a **robust & reliability-based** formulation

2. Once he holds the right formulation with respect to the system specifications and optimization goal then he has to select :

   1. the right algorithm to *efficiently* **evaluate** the statistical **measures** $(\mathbb{L}, \mathbb{K})$ and 2. the right **optimization algorithm** to use in conjunction with the one of 1.

Table of contents

The “double loop” issue

In the context of expensive deterministic solver for a ROBUST optimization

$$\boldsymbol{x}^\star = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \mathbb{L}(f(\boldsymbol{x};\boldsymbol{U})),$$

searching for x and estimating the risk measure : double loop

$\mathbb{L}^\star = +\infty,\ n = 1$

While $n \leq N$, **(loop 1 : optimization loop)**

$\rightarrow$ propose new $\boldsymbol{x} \in \mathcal{X}$,

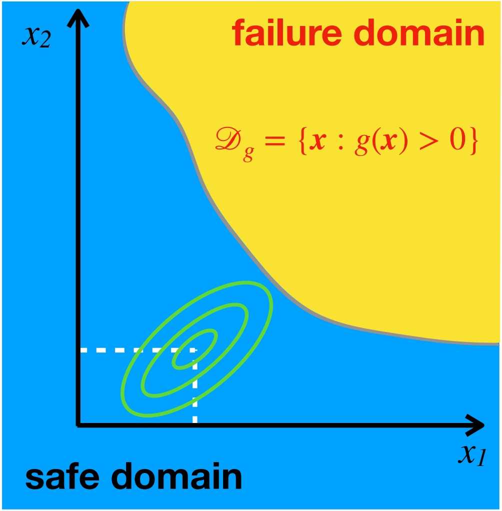
$\rightarrow$ “evaluate” $\mathbb{L}(f(\boldsymbol{x},\boldsymbol{U}))$ by repeating solver calls **(loop 2 : risk estimation loop)**

if $\mathbb{L}(f(\boldsymbol{x},\boldsymbol{U})) \leq \mathbb{L}^\star$, $\boldsymbol{x}^\star = \boldsymbol{x}$, $\mathbb{L}^\star = \mathbb{L}(f(\boldsymbol{x},\boldsymbol{U}))$ e.g. Monte-Carlo with m samples

end **if**

end **While**

High (multiplicative) **cost** : $\mathcal{O}(n \times m)$. **very time consuming !**


failure domain
$\mathscr{D}_g = \{\boldsymbol{x} : g(\boldsymbol{x}) > 0\}$
x_2
x_1
safe domain

Numerical approaches for failure probability estimations

Computational **cost very high** due to estimation of **failure probability** (small numbers $\propto$ tail of distribution) : need for **acceleration of one** of the 2 loops **or both**.

Existing useful approaches :

- **Simulation methods** : **Monte-Carlo** simulation (**MCS**) (robust but slow), numerical integration, importance sampling, subset simulation, stratified sampling
- **Approximation methods** :
  - First-Order Reliability method (**FORM**) [Ditlevsen & Madsen, 1996] it is based on the combination of an iterative **gradient-based** search of the so-called design point and a **local linear approximation of the limit-state function** in a suitably **transformed probabilistic space**
  - Second-Order Reliability method (SORM) it is a second-order refinement of FORM, the computational costs associated to this refinement increase rapidly with the number of input random variables
- **Metamodel approaches** (next courses)

Quick recall on Monte-Carlo approach for uncertainty propagation

relies on a **sample** of **iid** random variables. Vector $\boldsymbol{X}$ is randomly and independently sampled according to its pdf $f_{\boldsymbol{X}}$: $(\boldsymbol{X})_{i=1,\ldots,N} = (X_1^{(i)}, \ldots, X_n^{(i)})$

corresponding realizations are computed from the model g : $(g(\boldsymbol{X}))_{i=1,\ldots,N} = (g(X_1^{(i)}), \ldots, g(X_n^{(i)}))$: vector of realizations of a random variable of **unknown distribution**

goal : estimate some following quantity, where $f_{\boldsymbol{X}}$ is the pdf of $\boldsymbol{x}$:

$$I = \mathbb{E}[\psi(g(\boldsymbol{X}))] = \int_{\mathcal{A}} \psi(g(\boldsymbol{x})) f_{\boldsymbol{X}}(\boldsymbol{x}) d\boldsymbol{x},$$

thanks to the **MC estimator** of I :

$$I \approx \hat{I}_n = \frac{1}{N} \sum_{i=1}^{N} \psi(g(\boldsymbol{X}^{(i)})),$$

with the function ψ defining the desired QoI (mean, variance, **probability of failure**, etc.).

“Hit or miss” crude Monte-Carlo estimator of PoF

we wish to estimate :

$$P \equiv \mathbb{P}\left[g(\boldsymbol{X}) > 0\right] = \int_{\mathbb{R}^n} \mathbb{1}_{g(\boldsymbol{X})>0} f_{\boldsymbol{X}}(\boldsymbol{x}) d\boldsymbol{x},$$

with an **MC estimator** of P : $\hat{P}_N = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{g(\boldsymbol{X}^{(i)})>0}$.

Properties of this estimator :

- **no biais**
- **almost sure convergence** $\hat{P}_N \xrightarrow{N\to\infty} P$ a.s.
- variance : $\mathbb{V}(\hat{P}_N) = P(1-P)/N$ estimated by : $\hat{\sigma}^2_{\hat{P}_N} = \hat{P}_N(1-\hat{P}_N)/N$
- Central Limit theorem (CLT) allows to build **confidence intervals** around the prediction

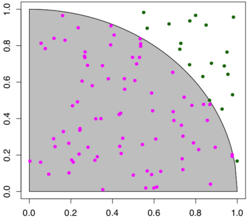
Precision of the MC estimator

coefficient of variation (error dispersion) : $\delta = \sqrt{\mathbb{V}(\hat{P}_N)}/\mathbb{E}(\hat{P}_N) = \sqrt{(1-P)/nP}$ is a nice criteria to monitor the convergence

For small values of P, the number of simulations required to reach precision δ is : $N_\delta \simeq \frac{1}{P\delta^2}$.

Quick example : estimating the number π with a **MC method** drawing **uniformly distributed points** over **a quarter of a disk** centered at the origin and of radius unity.

come up with an estimation of the sample size necessary to achieve a given accuracy η with a certain level of confidence α



Numerical approaches for failure probability estimations

Computational **cost very high** due to estimation of **failure probability** (small numbers $\propto$ tail of distribution) : need for **acceleration of one** of the 2 loops **or both**.

Existing useful approaches :

- **Simulation methods** : **Monte-Carlo** simulation (**MCS**) (robust but slow), numerical integration, importance sampling, subset simulation, stratified sampling
- **Approximation methods** :
  - First-Order Reliability method (**FORM**) [Ditlevsen & Madsen, 1996] it is based on the combination of an iterative **gradient-based** search of the so-called design point and a **local linear approximation of the limit-state function** in a suitably **transformed probabilistic space**
  - Second-Order Reliability method (SORM) [Breitung, 1989] it is a second-order refinement of FORM, the computational costs associated to this refinement increase rapidly with the number of input random variables
- **Metamodel approaches** (next courses)

Originally FORM is an approximation method for failure probability estimations

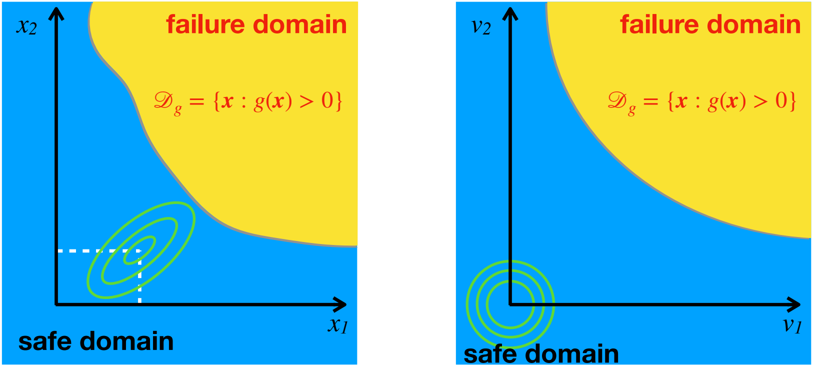
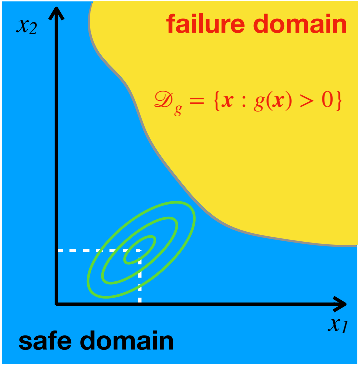


FIGURE: Physical space (left) and normalized space (right)

Originally FORM is an approximation method for failure probability estimations



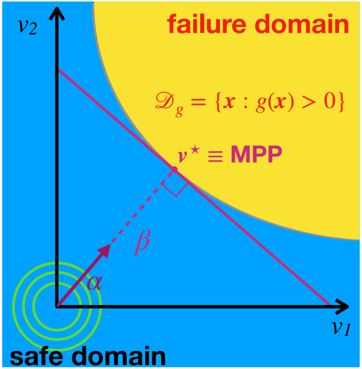


FIGURE: Physical space (left) and standard normalized space (right)

FORM approximation method for failure probability estimations

aims at the approximation of the integral involved in the failure probability, for a given design : $P_{f_i} \equiv \mathbb{P}\left[g_i(\boldsymbol{x}, \overline{\boldsymbol{U}}) > 0\right]$

4 main steps :

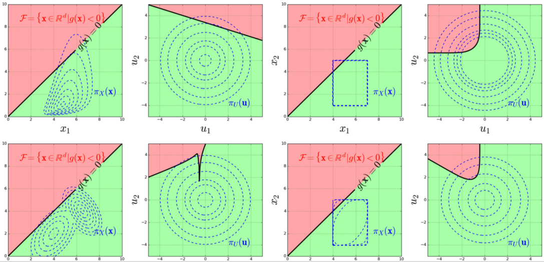
1. **isoprobabilistic transform** of the input random vector into a **standard normal** vector $\boldsymbol{v} \sim f_{\boldsymbol{V}} \equiv \mathcal{N}(\mathbb{0}, \mathbb{1})$
2. A search for the **most probable point of failure** (**MPP**), closest to the origin in this new standard normal space
3. An **approximation of the limit-state surface** at the MPP
4. **Computation of the PoF** from this approximated surface

FORM approximation method for failure probability estimations

aims at the approximation of the integral involved in the failure probability, for a given design : $P_{f_i} \equiv \mathbb{P}\left[g_i(\boldsymbol{x},\cdot) > 0\right]$

1. **isoprobabilistic transform** e.g. Rosenblatt (ie. joint probability $f_.$ is known) or Nataf (marginals f_{U_i} and correlation are known) [Lemaire et al., 2005] such that $g_i(\boldsymbol{x},\cdot) = g_i(\tau^{-1}(\cdot,\boldsymbol{v}),\cdot) = g_i^{\boldsymbol{v}}(\boldsymbol{v},\cdot)$ of the input random vector $\boldsymbol{x} \sim f_{\boldsymbol{X}}(\boldsymbol{x})$ into a **standard normal** vector $\boldsymbol{v} \sim f_{\boldsymbol{V}} \equiv \mathcal{N}(\mathbb{0},\mathbb{1})$. PoF takes a simpler form :

$$\int_{g_i(\boldsymbol{x},\cdot)>0} f_{\boldsymbol{X}}(\boldsymbol{x})d\boldsymbol{x} = \int_{\boldsymbol{v}\in\mathbb{R}^n:\, g_i^{\boldsymbol{v}}(\boldsymbol{v},\cdot)>0} f_{\boldsymbol{V}}(\boldsymbol{v})d\boldsymbol{v}$$



FORM approximation method for failure probability estimations

aims at the approximation of the integral involved in the failure probability, for a given design : $P_{f_i} \equiv \mathbb{P}\left[g_i(\boldsymbol{x}, \cdot) > 0\right]$

2. A search for the **most probable point of failure (MPP)** in the standard normal space (SNS), known as the **design point** $\boldsymbol{v}^\star = \arg\min_{\boldsymbol{v}}\{\|\boldsymbol{v}\|, g_i^{\boldsymbol{v}}(\boldsymbol{v}) = 0\}$ (nonlinear optimization problem under equality constraint) and $\|\boldsymbol{v}^\star\| \equiv \beta$ [Hasofer-Lind reliability index, 1974].
   The MPP is the point of the limit-state the closest to the origin (new pdf is spherically symmetric). Calling $g_i^{\boldsymbol{v}}$ is costly (nbr of calls must remain reasonable)

FORM approximation method for failure probability estimations

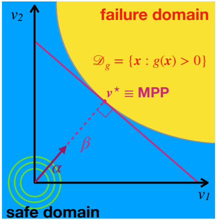
aims at the approximation of the integral involved in the failure probability, for a given design : $P_{f_i} \equiv \mathbb{P}\left[g_i(\boldsymbol{x},\cdot) > 0\right]$

2 A search for the **most probable point of failure (MPP)** in the standard normal space (SNS), known as the **design point** $\boldsymbol{v}^\star = \arg\min_{\boldsymbol{v}}\{\|\boldsymbol{v}\|, g_i^{\boldsymbol{v}}(\boldsymbol{v}) = 0\}$ (nonlinear optimization problem under equality constraint) and $\|\boldsymbol{v}^\star\| \equiv \beta$ [Hasofer-Lind reliability index, 1974].
The MPP is the point of the limit-state the closest to the origin (new pdf is spherically symmetric). Calling $g_i^{\boldsymbol{v}}$ is costly (nbr of calls must remain reasonable)

3 (once previous problem solved : next slide) a **linearization** (hypertangent plane) of the **limit-state surface at $\boldsymbol{v}^\star$** provides an **approximation** of the limit-state surface and is provided by a **first-order Taylor expansion** :

$$g_i^{\boldsymbol{v}}(\boldsymbol{v}) \approx \nabla g_i^{\boldsymbol{v}}(\boldsymbol{v})^T|_{\boldsymbol{v}=\boldsymbol{v}^\star}(\boldsymbol{v}-\boldsymbol{v}^\star),$$

so that **reliability index** $\beta = \boldsymbol{\alpha}^\star \cdot \boldsymbol{v}^\star$, where $\boldsymbol{\alpha}$ is the unit normal vector to the tangent plane at the design point.



FORM approximation method for failure probability estimations

aims at the approximation of the integral involved in the failure probability, for a given design : $P_{f_i} \equiv \mathbb{P}\left[g_i(\boldsymbol{x}, \cdot) > 0\right]$

3 a **linearization** of the **limit-state surface at** $v^\star$ provides an **approximation** of the limit-state surface :

$$g_i^{\boldsymbol{v}}(\boldsymbol{v}) \approx \nabla^T g_i^{\boldsymbol{v}}(\boldsymbol{v})|_{\boldsymbol{v}=\boldsymbol{v}^\star}(\boldsymbol{v} - \boldsymbol{v}^\star),$$

with normalized version :

$$\hat{g}_i^{\boldsymbol{v}}(\boldsymbol{v}) = \frac{\nabla^T g_i^{\boldsymbol{v}}(\boldsymbol{v}^\star)}{\|\nabla^T g_i^{\boldsymbol{v}}(\boldsymbol{v}^\star)\|}(\boldsymbol{v} - \boldsymbol{v}^\star) = \beta - \boldsymbol{\alpha}^T \cdot \boldsymbol{v},$$

so that **reliability index** $\beta = \boldsymbol{\alpha}^T \cdot \boldsymbol{v}^\star$, where $\boldsymbol{\alpha}$ is the unit normal vector to the tangent plane at the design point.

FORM approximation method for failure probability estimations

aims at the approximation of the integral involved in the failure probability, for a given design : $P_{f_i} \equiv \mathbb{P}\left[g_i(\boldsymbol{x}, \cdot) > 0\right]$

3 a **linearization** of the **limit-state surface at $\boldsymbol{v}^\star$** provides an **approximation** of the limit-state surface :
$$g_i^{\boldsymbol{v}}(\boldsymbol{v}) \approx \nabla^T g_i^{\boldsymbol{v}}(\boldsymbol{v})|_{v=\boldsymbol{v}^\star}(\boldsymbol{v}-\boldsymbol{v}^\star),$$
with normalized version :
$$\hat{g}_i^{\boldsymbol{v}}(\boldsymbol{v}) = \frac{\nabla^T g_i^{\boldsymbol{v}}(\boldsymbol{v}^\star)}{\|\nabla^T g_i^{\boldsymbol{v}}(\boldsymbol{v}^\star)\|}(\boldsymbol{v}-\boldsymbol{v}^\star) = \beta - \boldsymbol{\alpha}^T \cdot \boldsymbol{v},$$

so that **reliability index** $\beta = \boldsymbol{\alpha}^T \cdot \boldsymbol{v}^\star$, where $\boldsymbol{\alpha}$ is the unit normal vector to the tangent plane at the design point.

4 **analytical** computation of the resulting approximation of $P_{f_i} \approx \mathbb{P}(\hat{g}_i^{\boldsymbol{v}}(\boldsymbol{v}) > 0) = \mathbb{P}(\beta - \boldsymbol{\alpha}^T \cdot \boldsymbol{v} > 0) = 1 - \mathbb{P}(\beta \leq \boldsymbol{\alpha}^T \cdot \boldsymbol{v})$. Because is a standard normal RV $P_{f_i} \approx 1 - \Phi\left(-\beta_i\right)$, Φ : is the standard normal CDF.

Once β_i is obtained, the estimation of the PoF is direct (exact when limit-state is linear).

Hasofer-Lind-Rackwitz-Fiessler (HL-RF) algorithm

The rationale behind the **HL-RF algorithm** is to **iteratively** solve a **linearized** problem around the current point. Normally, the algorithm is started with $\boldsymbol{v}_0 = 0$

quadratic functional under **nonlinear equality** constraints $\Rightarrow$ Lagragian multiplier approach : $\mathcal{L}(\boldsymbol{v}, \lambda) = \frac{1}{2}\|\boldsymbol{v}\|^2 + \lambda\, g_i^{\boldsymbol{v}}(\boldsymbol{v})$, which gives :

Hasofer-Lind-Rackwitz-Fiessler (HL-RF) algorithm

The rationale behind the **HL-RF algorithm** is to **iteratively** solve a **linearized** problem around the current point. Normally, the algorithm is started with $\boldsymbol{v}_0 = 0$

quadratic functional under **nonlinear equality** constraints $\Rightarrow$ Lagragian multiplier approach : $\mathcal{L}(\boldsymbol{v}, \lambda) = \frac{1}{2}\|\boldsymbol{v}\|^2 + \lambda\, g_i^{\boldsymbol{v}}(\boldsymbol{v})$, which gives :

1. $\nabla_{\boldsymbol{v}} \mathcal{L}(\boldsymbol{v}^\star, \lambda^\star) = 0$ and 2. $\frac{\partial \mathcal{L}}{\partial \lambda}(\boldsymbol{v}^\star, \lambda^\star) = 0$, becoming :

Hasofer-Lind-Rackwitz-Fiessler (HL-RF) algorithm

The rationale behind the **HL-RF algorithm** is to **iteratively** solve a **linearized** problem around the current point. Normally, the algorithm is started with $\boldsymbol{v}_0 = 0$

quadratic functional under **nonlinear equality** constraints $\Rightarrow$ Lagragian multiplier approach : $\mathcal{L}(\boldsymbol{v}, \lambda) = \frac{1}{2}\|\boldsymbol{v}\|^2 + \lambda\, g_i^{\boldsymbol{v}}(\boldsymbol{v})$, which gives :

1. $\nabla_{\boldsymbol{v}}\mathcal{L}(\boldsymbol{v}^\star, \lambda^\star) = 0$ and 2. $\frac{\partial \mathcal{L}}{\partial \lambda}(\boldsymbol{v}^\star, \lambda^\star) = 0$, becoming :

1. $g_i^{\boldsymbol{v}}(\boldsymbol{v}^\star) = 0$ and 2. $\boldsymbol{v}^\star + \lambda \nabla g_i^{\boldsymbol{v}}((\boldsymbol{v}^\star) = 0$.

Hasofer-Lind-Rackwitz-Fiessler (HL-RF) algorithm

The rationale behind the **HL-RF algorithm** is to **iteratively** solve a **linearized** problem around the current point. Normally, the algorithm is started with $\boldsymbol{v}_0 = 0$

quadratic functional under **nonlinear equality** constraints $\Rightarrow$ Lagragian multiplier approach : $\mathcal{L}(\boldsymbol{v}, \lambda) = \frac{1}{2}\|\boldsymbol{v}\|^2 + \lambda\, g_i^{\boldsymbol{v}}(\boldsymbol{v})$, which gives :

1. $\nabla_{\boldsymbol{v}}\mathcal{L}(\boldsymbol{v}^\star, \lambda^\star) = 0$ and 2. $\frac{\partial \mathcal{L}}{\partial \lambda}(\boldsymbol{v}^\star, \lambda^\star) = 0$, becoming :

1. $g_i^{\boldsymbol{v}}(\boldsymbol{v}^\star) = 0$ and 2. $\boldsymbol{v}^\star + \lambda \nabla g_i^{\boldsymbol{v}}((\boldsymbol{v}^\star) = 0$.

At each iteration, limit-state approximation :

$g_i^{\boldsymbol{v}}(\boldsymbol{v}) \approx g_i^{\boldsymbol{v}}(\boldsymbol{v}_k) + \nabla g^{\boldsymbol{v}}_{i_{|\boldsymbol{v}_k}} \cdot (\boldsymbol{v} - \boldsymbol{v}_k)$, so the two equations become :

Hasofer-Lind-Rackwitz-Fiessler (HL-RF) algorithm

The rationale behind the **HL-RF algorithm** is to **iteratively** solve a **linearized** problem around the current point. Normally, the algorithm is started with $\boldsymbol{v}_0 = 0$

quadratic functional under **nonlinear equality** constraints $\Rightarrow$ Lagragian multiplier approach : $\mathcal{L}(\boldsymbol{v}, \lambda) = \frac{1}{2}\|\boldsymbol{v}\|^2 + \lambda\, g_i^{\boldsymbol{v}}(\boldsymbol{v})$, which gives :

1. $\nabla_{\boldsymbol{v}}\mathcal{L}(\boldsymbol{v}^\star, \lambda^\star) = 0$ and 2. $\frac{\partial \mathcal{L}}{\partial \lambda}(\boldsymbol{v}^\star, \lambda^\star) = 0$, becoming :

1. $g_i^{\boldsymbol{v}}(\boldsymbol{v}^\star) = 0$ and 2. $\boldsymbol{v}^\star + \lambda \nabla g_i^{\boldsymbol{v}}((\boldsymbol{v}^\star) = 0$.

At each iteration, limit-state approximation :

$g_i^{\boldsymbol{v}}(\boldsymbol{v}) \approx g_i^{\boldsymbol{v}}(\boldsymbol{v}_k) + \nabla g^{\boldsymbol{v}}_{i_{|\boldsymbol{v}_k}} \cdot (\boldsymbol{v} - \boldsymbol{v}_k)$, so the two equations become :

1. $\nabla g^{\boldsymbol{v}}_{i_{|\boldsymbol{v}_k}} \cdot (\boldsymbol{v}_{k+1} - \boldsymbol{v}_k)) + g_i^{\boldsymbol{v}}(\boldsymbol{v}_k) = 0$, and 2. $\boldsymbol{v}_{k+1} = \lambda \nabla g^{\boldsymbol{v}}_{i_{|\boldsymbol{v}_k}}$, ultimately giving :

Hasofer-Lind-Rackwitz-Fiessler (HL-RF) algorithm

The rationale behind the **HL-RF algorithm** is to **iteratively** solve a **linearized** problem around the current point. Normally, the algorithm is started with $\boldsymbol{v}_0 = 0$

quadratic functional under **nonlinear equality** constraints $\Rightarrow$ Lagragian multiplier approach : $\mathcal{L}(\boldsymbol{v}, \lambda) = \frac{1}{2}\|\boldsymbol{v}\|^2 + \lambda\, g_i^{\boldsymbol{v}}(\boldsymbol{v})$, which gives :

1. $\nabla_{\boldsymbol{v}}\mathcal{L}(\boldsymbol{v}^\star, \lambda^\star) = 0$ and 2. $\frac{\partial \mathcal{L}}{\partial \lambda}(\boldsymbol{v}^\star, \lambda^\star) = 0$, becoming :
1. $g_i^{\boldsymbol{v}}(\boldsymbol{v}^\star) = 0$ and 2. $\boldsymbol{v}^\star + \lambda \nabla g_i^{\boldsymbol{v}}((\boldsymbol{v}^\star) = 0$.

At each iteration, limit-state approximation :
$g_i^{\boldsymbol{v}}(\boldsymbol{v}) \approx g_i^{\boldsymbol{v}}(\boldsymbol{v}_k) + \nabla g^{\boldsymbol{v}}_{i_{|\boldsymbol{v}_k}} \cdot (\boldsymbol{v} - \boldsymbol{v}_k)$, so the two equations become :

1. $\nabla g^{\boldsymbol{v}}_{i_{|\boldsymbol{v}_k}} \cdot (\boldsymbol{v}_{k+1} - \boldsymbol{v}_k)) + g_i^{\boldsymbol{v}}(\boldsymbol{v}_k) = 0$, and 2. $\boldsymbol{v}_{k+1} = \lambda \nabla g^{\boldsymbol{v}}_{i_{|\boldsymbol{v}_k}}$, ultimately giving :

$\boldsymbol{v}_{k+1} = \left(\alpha_k \cdot \boldsymbol{v}_k + \frac{g_i^{\boldsymbol{v}}(\boldsymbol{v}_k)}{\nabla g^{\boldsymbol{v}}_{i_{|\boldsymbol{v}_k}}} \right) \alpha_k = \beta_k \alpha_k$, and ... at convergence :

Hasofer-Lind-Rackwitz-Fiessler (HL-RF) algorithm

The rationale behind the **HL-RF algorithm** is to **iteratively** solve a **linearized** problem around the current point. Normally, the algorithm is started with $\boldsymbol{v}_0 = 0$

quadratic functional under **nonlinear equality** constraints $\Rightarrow$ Lagragian multiplier approach : $\mathcal{L}(\boldsymbol{v}, \lambda) = \frac{1}{2}\|\boldsymbol{v}\|^2 + \lambda\, g_i^{\boldsymbol{v}}(\boldsymbol{v})$, which gives :

1. $\nabla_{\boldsymbol{v}}\mathcal{L}(\boldsymbol{v}^\star, \lambda^\star) = 0$ and 2. $\frac{\partial \mathcal{L}}{\partial \lambda}(\boldsymbol{v}^\star, \lambda^\star) = 0$, becoming :
1. $g_i^{\boldsymbol{v}}(\boldsymbol{v}^\star) = 0$ and 2. $\boldsymbol{v}^\star + \lambda \nabla g_i^{\boldsymbol{v}}((\boldsymbol{v}^\star) = 0$.

At each iteration, limit-state approximation :
$g_i^{\boldsymbol{v}}(\boldsymbol{v}) \approx g_i^{\boldsymbol{v}}(\boldsymbol{v}_k) + \nabla g^{\boldsymbol{v}}_{i_{|\boldsymbol{v}_k}} \cdot (\boldsymbol{v} - \boldsymbol{v}_k)$, so the two equations become :

1. $\nabla g^{\boldsymbol{v}}_{i_{|\boldsymbol{v}_k}} \cdot (\boldsymbol{v}_{k+1} - \boldsymbol{v}_k)) + g_i^{\boldsymbol{v}}(\boldsymbol{v}_k) = 0$, and 2. $\boldsymbol{v}_{k+1} = \lambda \nabla g^{\boldsymbol{v}}_{i_{|\boldsymbol{v}_k}}$, ultimately giving :

$\boldsymbol{v}_{k+1} = \left(\alpha_k \cdot \boldsymbol{v}_k + \frac{g_i^{\boldsymbol{v}}(\boldsymbol{v}_k)}{\nabla g^{\boldsymbol{v}}_{i_{|\boldsymbol{v}_k}}} \right) \alpha_k = \beta_k \alpha_k$, and ... at convergence :

$\beta = \alpha^\star \cdot \boldsymbol{v}^\star$, when $g_i^{\boldsymbol{v}}(\boldsymbol{v}_k = \boldsymbol{v}^\star) \approx 0$.

Advantages and disadvantages of approximation methods for PoF estimation

(+) **easy to implement** (for reliability study alone)

(+) **inexpensive** in terms of number of simulations.
The integration problem is replaced by a minimization problem, often requiring much less evaluations.

(-) no real guarantee on the result, an no error bars for the probability of failure

(-) require intermediate steps which may be **hard to integrate in a general-purpose optimization algorithm**

Focus on RBDO problems resolution

We went briefly over techniques handling the "uncertainty part" (inner loop) but it obviously has to be linked to the "optimization part" (outer loop).

For solving a full RBDO problem different approaches are classified into three groups, – **two-level**, – **mono-level** [Liang et al., 2004] and – **decoupled** [Du and Chen, 2004] approaches [Chateauneuf and Aoues, 2008]

The **two-level approaches** either rely on :

simulations to sample the joint distribution of the random variables at play (MCS, Importance Sampling, Subset simulation, etc) or

✓ **approximation methods** such as FORM & SORM with notably :

- **Reliability Index Approach (RIA with a FORM analysis)** [Rackwitz and Fiessler, 1978]
- **Performance Measure Approach (PMA with an inverse FORM analysis)** [Youn et al. 2005, Cho and Lee 2011]
- **Quantile Estimate Approach** [Moustapha et al. 2016]

Those techniques can be coupled to **general-purpose global optimization algorithms**, such as **SQP, GA, constrained (1+1)-CMA-ES**, (eventually with hybrid formulation involving additional gradient-based minimization)

Two-level RBDO solvers relying on two different FORM algorithms

essentially

for a given design $\boldsymbol{x}$,

Reliability Index Approach (**RIA**) :

$$\boldsymbol{v}^{\star}_{\mathsf{RIA}_i} = \arg\min_{\boldsymbol{v}\in\boldsymbol{\mathcal{V}}} \|\boldsymbol{v}\| - \beta_i, \tag{12}$$

$$\text{such that } g_i^{\boldsymbol{v}}(\boldsymbol{x},\boldsymbol{v}) = 0 \tag{13}$$

Performance Measure Approach (**PMA**) :

$$\boldsymbol{v}^{\star}_{\mathsf{PMA}_i} = \arg\min_{\boldsymbol{v}\in\boldsymbol{\mathcal{V}}} g_i^{\boldsymbol{v}}(\boldsymbol{x},\boldsymbol{v}), \tag{14}$$

$$\text{such that } \|\boldsymbol{v}\| = \beta_i^{\mathsf{target}} \tag{15}$$

Integrating **uncertainties** into the design process **greatly** modifies the optimization problem to be solved.

The **statistical measures of risk** must be carefully **adapted to the problem**.

As a first approach, the solution of a **ROBUST** or **RBDO** problem requires the duplication of calculations, and involves a **HIGH numerical cost** $\Rightarrow$ need for **accelerating strategies** to lower number of calls to the model...

... toward **surrogate modeling** approaches :

- addressing both **design** and **parameters uncertainty** ?
- **adaptive** learning ?
- best harnessing of measure of metamodeling **errors** ?