Robust optimization with OpenTURNS
the **otrobopt** module

Régis LEBRUN

Airbus
22, rue du Gouverneur Général Eboué
92130 ISSY les MOULINEAUX
regis.lebrun@airbus.com

25 octobre 2019

Robust optimization with OpenTURNS

Introduction I

The aim of this presentation is to propose a practical approach to cope with uncertainties in an industrial optimization process .
Even if the accumulated knowledge gives a good view of the optimization problem an optimal design has to solve, and even if all the relevant numerical models are available, some parts of the problem are still subject to uncertainties :

- Some parts of the system may have still to be designed
- Some design quantities are known up to a finite precision
- Some environmental parameters are intrinsically stochastic
- ...

To this end, we focus on

- the introduction of uncertainties in a parametric optimization problem
- the relevant formulations depending on the industrial objectives of the engineer.

We adopt a probabilistic framework to model the sources of uncertainty, and we make the link between a parametric optimization problem and its robust counterpart.

Here robust is a generic term to design both robustness, ie the low sensitivity of the objective function to uncertainties, and reliability, ie the fact that the system remains in the feasible set for all (or most of) the possible values of the uncertainties.

Introduction II

A practical approach is proposed through the description of a four steps generic methodology, as a guide to help engineers to incorporate uncertainties in an optimization process.

This methodology will be demonstrated on an academic example in order to give a concrete feeling of the different new concepts we introduce.

Module *otrobopt*

The *otrobopt* module has been specified by Airbus and implemented by Phimeca in the frame of the IRT project called *ROM* in 2015-2016.
Read the documentation at openturns.github.io/otrobopt/master
Download *otrobopt* here https ://github.com/openturns/otrobopt.

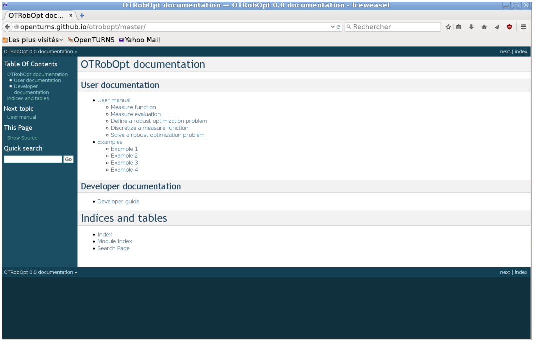

OTRobOpt documentation — OTRobOpt 0.0 documentation - Iceweasel
OTRobOpt doc...
openturns.github.io/otrobopt/master/
Rechercher
Les plus visités
OpenTURNS
Yahoo Mail
OTRobOpt 0.0 documentation »
next | index

Table Of Contents
- OTRobOpt documentation
  - User documentation
  - Developer documentation
- Indices and tables

Next topic
User manual

This Page
Show Source

Quick search
Go

OTRobOpt documentation

User documentation
- User manual
  - Measure function
  - Measure evaluation
  - Define a robust optimization problem
  - Discretize a measure function
  - Solve a robust optimization problem
- Examples
  - Example 1
  - Example 2
  - Example 3
  - Example 4

Developer documentation
- Developer guide

Indices and tables
- Index
- Module Index
- Search Page

All the scripts included into the presentation are available upon request.

Example : manufacturing dispersions I

The performance of a system is governed by two geometrical parameters (x_0, x_1) and is given by a function $h(x_0, x_1)$. The range of parameters is restricted by two constraints $g_1^{det} \geq 0$ and $g_2^{det} \geq 0$:

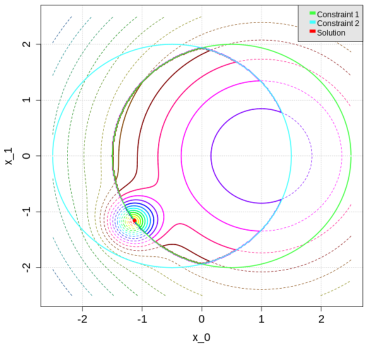
- Objective function : $h(\underline{x}) = 15\left[(x_0 - 1)^2 + x_1^2\right] - 160e^{-5\left[(x_0+1.2)^2+(x_1+1.2)^2\right]}$
- Constraints : $g_1^{det}(\underline{x}) = 4 - \left[(x_0 - 0.5)^2 + x_1^2\right] \geq 0$ et $g_2^{det}(\underline{x}) = 4 - \left[(x_0 + 0.5)^2 + x_1^2\right] \geq 0$.

$$(S_{cl}) : \quad x_{cl}^* = \underset{\substack{g_1^{det}(\underline{x}) \geq 0 \\ g_2^{det}(\underline{x}) \geq 0}}{\text{argmin}} \quad h(\underline{x})$$

$\Longrightarrow \underline{x}_{cl}^* = (-1.02, -1.08)$

…but the manufacturing process introduces dispersions on $\underline{x}_{cl}$!

Deterministic optimization problem



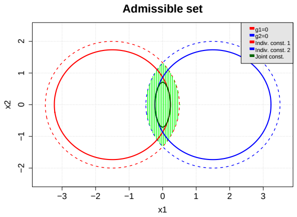
Example : manufacturing dispersions II

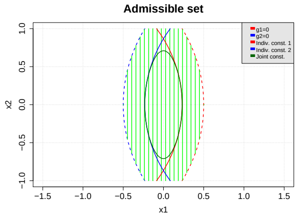
Robust model : the actual value of $\underline{x}$ is the specification plus a deviation $\underline{\Theta}$, eg $\underline{\Theta} \sim \mathcal{N}(\underline{O}, \sigma I_2)$:

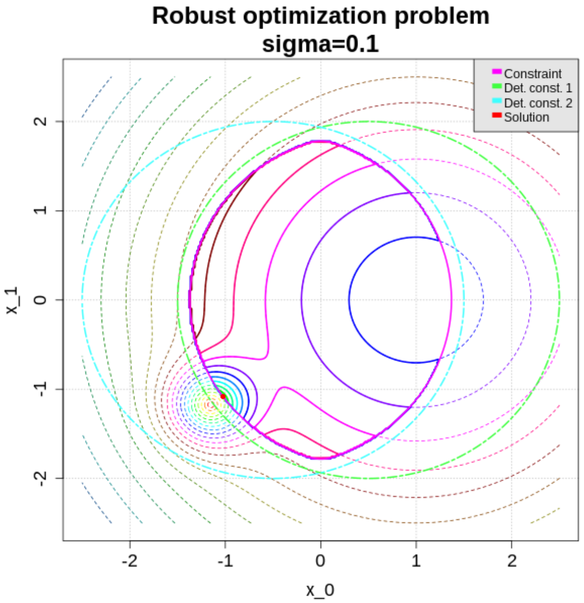
- Objective function : $J(\underline{x}, \underline{\Theta}) = h(\underline{x} + \underline{\Theta}) \Longrightarrow$ **this is a random variable !**
- Constraints : $g_1(\underline{x}, \underline{\Theta}) = g_1^{det}(\underline{x} + \underline{\Theta})$ and $g_2(\underline{x}, \underline{\Theta}) = g_2^{det}(\underline{x} + \underline{\Theta}) \Longrightarrow$ **random variables !**

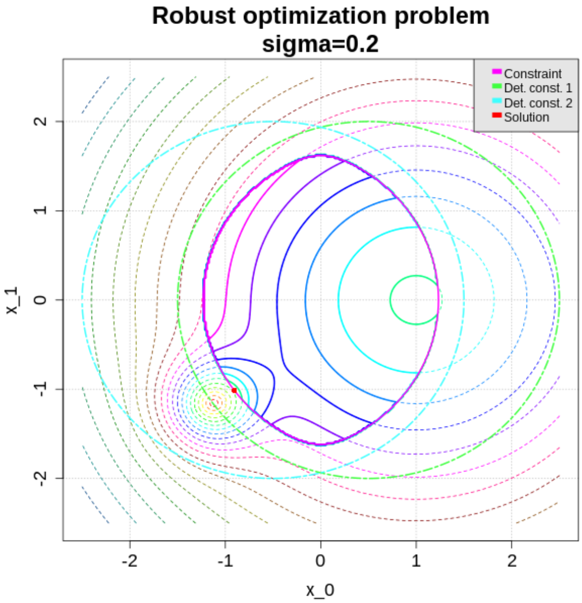
$$(S_{rob}) : \quad x^*_{rob} = \underset{\substack{\mathbb{P}\left(g_1(\underline{x}, \underline{\Theta}) \geq 0 \\ \cap\, g_2(\underline{x}, \underline{\Theta}) \geq 0\right) \geq 0.9}}{\text{argmin}} \mathbb{E}_{\underline{\Theta}}\left[J(\underline{x}, \underline{\Theta})\right]$$

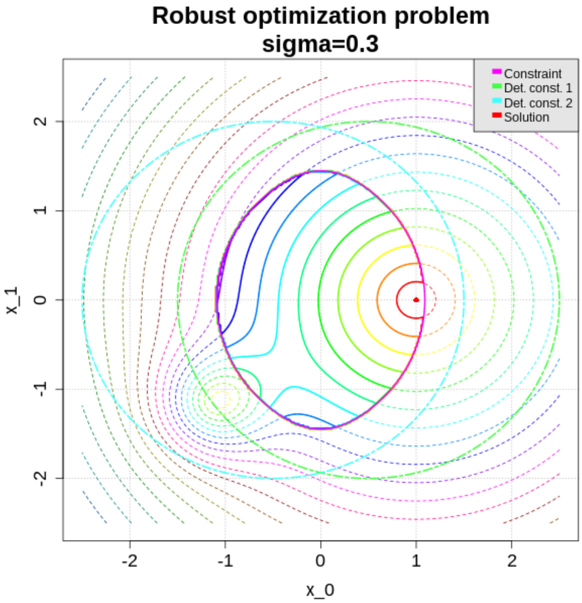
Admissible set :

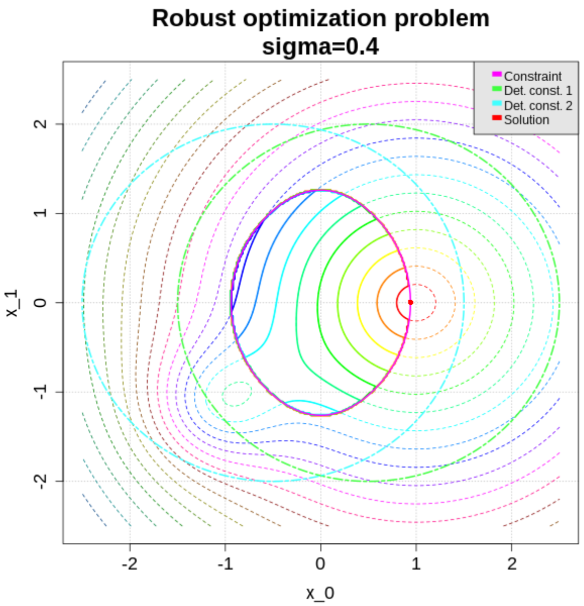


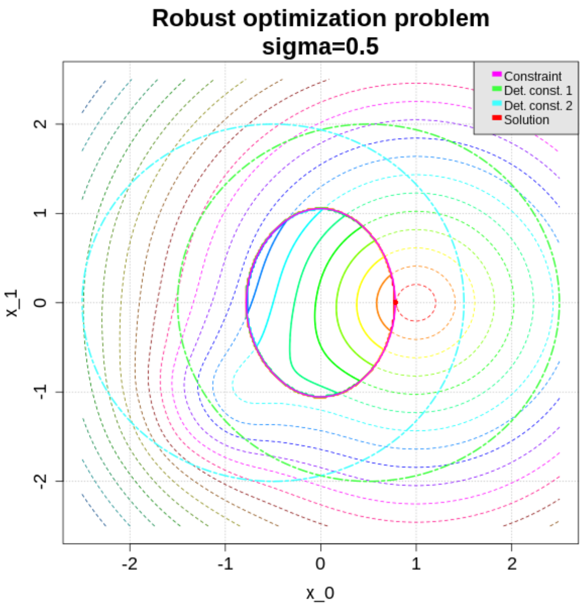


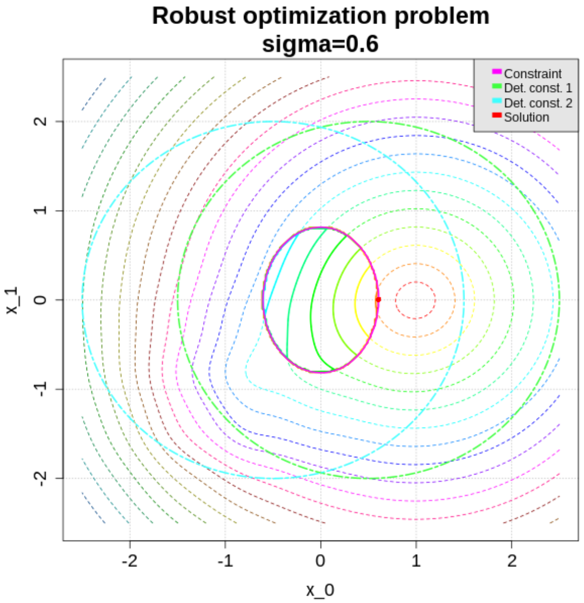

Robust optimization problem
sigma=0.1
Constraint
Det. const. 1
Det. const. 2
Solution
x_0
x_1
-2
-1
0
1
2


Robust optimization problem
sigma=0.2
🟪 Constraint
🟩 Det. const. 1
🟦 Det. const. 2
🟥 Solution
x_1
x_0
-2
-1
0
1
2


Robust optimization problem
sigma=0.3
Constraint
Det. const. 1
Det. const. 2
Solution
x_1
x_0
2
1
0
-1
-2
-2
-1
0
1
2


Robust optimization problem
sigma=0.4
🟪 Constraint
🟩 Det. const. 1
🟦 Det. const. 2
🟥 Solution
x_0
x_1
-2
-1
0
1
2
-2
-1
0
1
2


Robust optimization problem
sigma=0.5
Constraint
Det. const. 1
Det. const. 2
Solution
x_0
x_1
-2
-1
0
1
2
-2
-1
0
1
2


Robust optimization problem
sigma=0.6
Constraint
Det. const. 1
Det. const. 2
Solution
x_0
x_1
-2
-1
0
1
2

Classical formulation of the optimization problem

Objective : find $\underline{x}^*$ **minimizing** $J(\underline{x}, \underline{\theta})$ **subject to the constraints** $G(\underline{x}, \underline{\theta}) \geq 0$ **where** $\underline{\theta} \in D$.

Several ways to think about it :

1. The parameter $\underline{\theta}$ is given a *reasonable* value $\underline{\theta}_0$, eg a mean value, a penalized value, ...then one minimizes $J(\underline{x}, \underline{\theta}_0)$ s.t. $G(\underline{x}, \underline{\theta}_0) \geq 0$

$$(S_{cl}) : \quad x^*_{cl} = \text{argmin} \quad G(\underline{x}, \underline{\theta}_0) \geq 0 \;\; J(\underline{x}, \underline{\theta}_0)$$

2. Or one maximises $J(\underline{x}^*(\underline{\theta}), \underline{\theta})$ wrt $\underline{\theta} \in [\underline{\theta}_1, \underline{\theta}_2]$, the so-called maximin approach

$$(S_{cl}) : \quad x^*_{cl} = \arg \max_{\underline{\theta} \in [\underline{\theta}_1, \underline{\theta}_2]} \min_{G(\underline{x}, \underline{\theta}) \geq 0} J(\underline{x}, \underline{\theta})$$

3. ...

Beware : approaches either with no actual uncertainty model or with very conservative results !

Robust formulation I

The dispersion $\underline{\theta}$ is described by a probability distribution $\mathcal{D}$ supposed to be fully specified.
$J(\underline{x}, \underline{\theta})$ and $G(\underline{x}, \underline{\theta})$ become random variables, and they are replaced by integrated counterparts (moments, quantiles, ...)

- $J(\underline{x}, \underline{\theta}) \to \boldsymbol{\rho_{J,\mathcal{D}}(\underline{x})}$: robustness measure ;
- $G(\underline{x}, \underline{\theta}) \to \boldsymbol{\lambda_{G,\mathcal{D}}(\underline{x})}$: reliability measure .

New problem :

$$(S_{rob}) : \quad x^*_{rob} = \text{argmin} \ \ \lambda_{G,\mathcal{D}}(\underline{x}) \geq 0 \ \ \rho_{J,\mathcal{D}}(\underline{x})$$

Remarks :

1. The measures ρ and λ are only functions of $\underline{x}$: the effect of $\underline{\theta}$ has been integrated wrt its distribution $\mathcal{D}$;
2. So (S_{rob}) is a classical deterministic optimization problem, which could be solved by any classical algorithm... but there is a clever way to solve it !
3. Remark : on can have $\underline{x}^*_{rob} \neq \underline{x}^*_{cl}$ (in fact it is expected !).

Some approaches consider only partially known distributions $\mathcal{D}$:

1. The distribution $\mathcal{D}$ is given only through moments, bounds, dots which are taken as constraints over $\mathcal{D}$

Robust formulation II

2. Then the optimization is done also over the set of admissible distributions.

Beware : this approach looks very promising, but the actual optimal distribution is most of the time discrete, so even if it gives a guarantee wrt the choice of $\mathcal{D}$, it is clearly not suited to model a continuous quantity !
Example : if one fixes the mean and the variance of the distribution, one get a discrete distribution with 2 possible values on both sides of the mean value.
In addition to this drawback, asking a user for the values of higher order moments is a risky game, as the possible values have to satisfy complex algebraic constraints to be the moments of any distribution.

Robust formulation : modeling steps

$$(S_{cl}) : \underline{x}_{cl}^{*} = \underset{G(\underline{x},\underline{\theta}_0) \geq 0}{\operatorname{argmin}} \quad J(\underline{x},\underline{\theta}_0)$$

$$(S_{rob}) : \underline{x}_{rob}^{*} = \underset{\lambda_{G,\mathcal{D}}(\underline{x}) \geq 0}{\operatorname{argmin}} \quad \rho_{J,\mathcal{D}}(\underline{x})$$

The robust counterpart of the initial optimization problem requires **additional modeling steps** :

- choice of $\mathcal{D}$;
- choice of the robustness measure ρ ;
- choice of the reliability measure λ.

Remark : Pay attention to the robust counterpart of equality constraints !

Equality constraints in robust formulations

The equality constraints can be problematic :

- If there is a deterministic equality constraint of the form : $\boldsymbol{h(\underline{x}, \underline{\theta}) = 0}$, how to choose a reliability measure if $\mathcal{D}$ is a continuous distribution ?
  - if one choose a probability measure : problemn because in general, $\mathbb{P}\left(h(\underline{x}, \underline{\theta}) = 0\right) = 0$!
  - if one choose a regular moment such as the expectation, ok.
- If one replace an inequality constraint $\boldsymbol{g(\underline{x}, \underline{\theta}) \geq 0}$ by an equality constraint on a probability measure $\mathbb{P}\left(g(\underline{x}, \underline{\theta}) \geq 0\right) = \alpha$, beware of the value of α !

Example : Let's consider $\mathbb{P}\left(x\theta \geq 0\right) = \alpha$ with $\theta \sim \mathcal{D}$ a continuous random variable which support contains 0.
Then $\mathbb{P}\left(x\theta \geq 0\right)$ can take only 3 different values :

$$\mathbb{P}\left(x\theta \geq 0\right) = \left| \begin{array}{ll} F_{\mathcal{D}}(0) & \text{if } x < 0 \\ 1 - F_{\mathcal{D}}(0) & \text{if } x > 0 \\ 1 & \text{if } x = 0 \end{array} \right.$$

Robust optimization with OpenTURNS

Measures $M_{f,\mathcal{D}}(.)$

Let $f(., \underline{\Theta}) : \mathbb{R}^d \to \mathbb{R}^p$ be a function and $\underline{\Theta} \sim \mathcal{D}$ a random vector. A **measure** $\mathbb{R}^d \to \mathbb{R}^p$: $\underline{x} \to M_{f,\mathcal{D}}(\underline{x})$ is a function of $\underline{x}$ and $\mathcal{D}$, not of specific values of the random vector $\underline{\Theta}$. It can be either :

- a measure of robustness : $\underline{x} \to \rho_{J,\mathcal{D}}(\underline{x})$.
- a measure of reliability : $\underline{x} \to \lambda_{G,\mathcal{D}}(\underline{x})$.

The ***otrobopt*** module proposes the following measures :

- In the scale of f : MeanMeasure, MeanStandardDeviationTradeoffMeasure, QuantileMeasure, WorstCaseMeasure
- In the scale of f^2 : VarianceMeasure
- In a probability scale $[0, 1]$: JointChanceMeasure, IndividualChanceMeasure

otrobopt (current version) : one can define a problem using any measure $M_{f,\mathcal{D}} : \mathbb{R}^d \to \mathbb{R}^{\boldsymbol{p}}$ but only problems with robustness measure of the form $\rho_{J,\mathcal{D}} : \mathbb{R}^d \to \mathbb{R}$ can be solved (no *multiobjective robust optimization* yet). If $p > 1$, one can solve the formulation expressed using *otrobopt* by using third party Python modules.

MeanMeasure

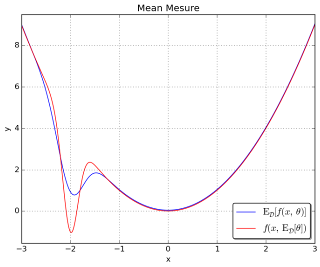
For $f(., \underline{\Theta}) : \mathbb{R}^d \to \mathbb{R}^p$, the MeanMeasure is defined by :

$$M_{f,\mathcal{D}}(\underline{x}) = \mathbb{E}_{\mathcal{D}}[f(x, \Theta)]$$

Example : Additive noise $h(x) = (x-1)^2 - 5e^{-20(2+x)^2}$, $f(x, \Theta) = h(x+\Theta)$ with $\Theta \sim \mathcal{N}(0, 0.2)$.

```
--------------------------------------------------------------------------------- -
>>> thetaDist = ot.Normal(0, 0.2)
>>> f_base = ot.SymbolicFunction(['x', 'theta'], ['(x+theta)^2-5*exp(-20*(x+theta+2)^2)'])
>>> f = ot.ParametricFunction(f_base, [1], thetaDist.getMean())
>>> measure = otrobopt.MeanMeasure(f, thetaDist)

>>> ot.Show(measure.draw(-3.0, 3.0, 128))
--------------------------------------------------------------------------------- -
```



510 calls to f to compute $M_{f,\mathcal{D}}(-2.0)$.

VarianceMeasure

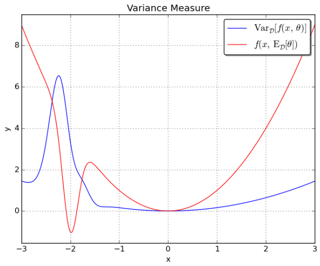
for $f(.,\underline{\Theta}) : \mathbb{R}^d \to \mathbb{R}$ the VarianceMeasure is defined by :

$$M_{f,\mathcal{D}}(\underline{x}) = \mathrm{Var}_{\mathcal{D}}[f(x,\Theta)]$$

For $f(.,\underline{\Theta}) : \mathbb{R}^d \to \mathbb{R}^p$, $M_{f,\mathcal{D}}(\underline{x}) = (M_{f_1,\mathcal{D}}(\underline{x}), \ldots, M_{f_p,\mathcal{D}}(\underline{x}))$.

Example : Additive noise $h(x) = (x-1)^2 - 5e^{-20(2+x)^2}$, $f(x,\Theta) = h(x+\Theta)$ with $\Theta \sim \mathcal{N}(0, 0.2)$.

```
-----------------------------------------------------
>>> measure = otrobopt.VarianceMeasure(f, thetaDist)
-----------------------------------------------------
```



390 calls to f to compute $M_{f,\mathcal{D}}(-2.0)$.

MeanStandardDeviationTradeoffMeasure

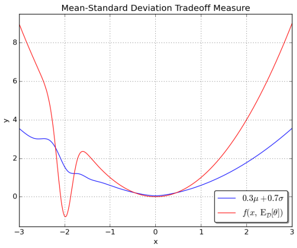
This measure is a tradeoff between the mean and the standard deviation of f.
For $f(.,\underline{\Theta}) : \mathbb{R}^d \to \mathbb{R}$, $\mu = \mathbb{E}_{\mathcal{D}}[f(x,\Theta)]]$ and $\sigma^2 = \mathrm{Var}_{\mathcal{D}}[f(x,\Theta)]$ the MeanStandardDeviationTradeoffMeasure is defined by :

$$M_{f,\mathcal{D},\alpha}(\underline{x}) = (1-\alpha)\mu + \alpha\sigma$$

Pour $f(.,\underline{\Theta}) : \mathbb{R}^d \to \mathbb{R}^p$, $M_{f,\mathcal{D},\alpha}(\underline{x}) = (M_{f_1,\mathcal{D},\alpha_1}(\underline{x}), \dots, M_{f_p,\mathcal{D},\alpha_p}(\underline{x}))$.

Example : Additive noise $h(x) = (x-1)^2 - 5e^{-20(2+x)^2}$, $f(x,\Theta) = h(x+\Theta)$ with $\Theta \sim \mathcal{N}(0, 0.2)$.

```
-------------------------------------------------------------------------------------
>>> alpha = 0.7
>>> measure = otrobopt.MeanStandardDeviationTradeoffMeasure(f, thetaDist, [alpha])
-------------------------------------------------------------------------------------
```



570 calls to f to compute $M_{f,\mathcal{D}}(-2.0)$.

QuantileMeasure

This measure is the quantile of order α of $f(\underline{x}, \underline{\Theta})$.
For $f(., \underline{\Theta}) : \mathbb{R}^d \to \mathbb{R}^p$ the QuantileMeasure is defined by :

$$M_{f,\mathcal{D},\alpha}(\underline{x}) = \inf\{\, \underline{s} \in \mathbb{R}^d \mid \mathbb{P}\left(f(\underline{x}, \underline{\Theta}) \leq \underline{s}\right) \geq \alpha \,\}$$

Example : Additive noise $h(x) = (x-1)^2 - 5e^{-20(2+x)^2}$, $f(x, \Theta) = h(x + \Theta)$ with $\Theta \sim \mathcal{N}(0, 0.2)$.

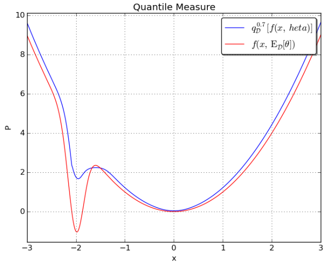
If f is the weight of a part, one looks for the best dimension x^* to mimimize the quantile of order 70% of the weght :

the actual weight $w = f(x + \Theta)$ will be les than the optimal weig

```
----------------------------------------------------------- -
>>> q = 0.7
>>> measure = otrobopt.QuantileMeasure(f, thetaDist, q)
----------------------------------------------------------- -
```

9090 calls to f were needed to compute $M_{f,\mathcal{D}}(-2.0)$.

<u>Remark</u> : The computation of $F_{f(\underline{x},\underline{\Theta})}(\underline{s})$ is done using an adaptive Gauss-Kronrod quadrature. Brent's method (a mix of bisection and inverse quadratic interpolation) is then used to find $\underline{s}$ such that $F_{f(\underline{x},\underline{\Theta})}(\underline{s}) = \alpha \Longrightarrow$ high cost !



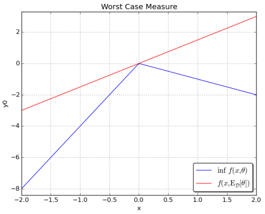
WorstCaseMeasure

For $f(., \underline{\Theta}) : \mathbb{R}^d \rightarrow \mathbb{R}^p$ the WorstCaseMeasure is defined by :

$$M_{f,\mathcal{D},\alpha}(\underline{x}) = \inf_{\underline{\Theta} \in Supp\,\mathcal{D}} f(\underline{x}, \underline{\Theta}) \quad \text{or} \quad M_{f,\mathcal{D},\alpha}(\underline{x}) = \sup_{\underline{\Theta} \in Supp\,\mathcal{D}} f(\underline{x}, \underline{\Theta})$$

Example : $f(x, \Theta) = x\Theta$, $\Theta \sim \mathcal{U}(-1, 4)$ and one takes the *inf* measure.

```
------------------------------------------------------------------
>>> distTheta = ot.Uniform(-1.0, 4.0)
>>> f_base = SymbolicFunction(['x', 'theta'], ['x*theta'])
>>> f = ot.ParametricFunction(f_base, [1], thetaDist.getMean())
>>> myMeasure = WorstCaseMeasure(f, distTheta, True)
------------------------------------------------------------------
```



Remarques :

- We only use the *support* of $\mathcal{D}$, not the actual distribution.
- If $Supp\,\mathcal{D}$ has an infinite bound, it is explored up to the (finite) bounds of the *numerical* range of $\mathcal{D}$. Beware of the problems where $\sup f(\underline{x}, \underline{\Theta})$ does not depend on $\underline{x}$!
- IF $Supp\,\mathcal{D} = [\underline{\theta}_1, \underline{\theta}_2] \Longrightarrow$ what is the benefit of the robust formulation ?

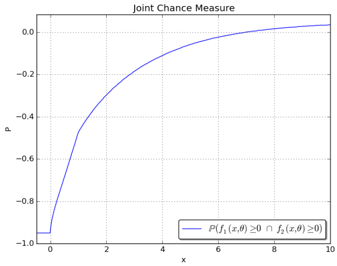
JointChanceMeasure

This measure is mainly used as a reliability measure.
For $f(.,\underline{\Theta}) : \mathbb{R}^d \to \mathbb{R}^p$ the JointChanceMeasure is defined by :

$$M_{f,\mathcal{D},\alpha}(\underline{x}) = \mathbb{P}\left(f_1(\underline{x},\underline{\Theta}) \geq 0 \cap \ldots \cap f_p(\underline{x},\underline{\Theta}) \geq 0\right) - \alpha$$

The reliability constraint is $\mathbb{P}\left(f_1(\underline{x},\underline{\Theta}) \geq 0 \cap \ldots \cap f_p(\underline{x},\underline{\Theta}) \geq 0\right) \geq \alpha$ (or $\leq$).
Example : $f(x,\Theta) = [x-\Theta, x-\Theta^2]$, $\Theta \sim \mathcal{N}(1,1)$ and $\alpha = 0.95$.

```
--------------------------------------------------------------------------------
>>> thetaDist = ot.Normal(1.0, 1.0)
>>> f_base = ot.SymbolicFunction(['x', 'theta'],  ['x-theta', 'x-theta^2'])
>>> f = ot.ParametricFunction(f_base, [1], thetaDist.getMean())
>>> measure = otrobopt.JointChanceMeasure(f, thetaDist, ot.GreaterOrEqual(), 0.95)
--------------------------------------------------------------------------------
```



2340 calls to f to compute $M_{f,\mathcal{D}}(2.0)$.
Remark :
$M_{f,\mathcal{D},\alpha}(\underline{x}) = \int_{\mathbb{R}^p} \prod_{k=1}^{p} 1_{f_k(\underline{x},\underline{\theta})\geq 0} p(\underline{\theta})\, d\underline{\theta} - \alpha.$

IndividualChanceMeasure

This measure is mainly used as a reliability measure.

For $f(.,\underline{\Theta}) : \mathbb{R}^d \to \mathbb{R}^p$, the IndividualChanceMeasure is defined by :

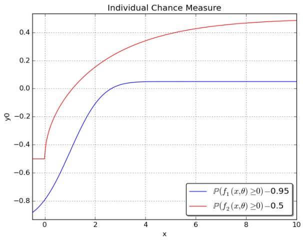
$$M_{f,\mathcal{D},\alpha}(\underline{x}) = \left| \begin{array}{l} \mathbb{P}\left(f_1(\underline{x},\underline{\theta}) \geq 0\right) - \alpha_1 \\ \dots \\ \mathbb{P}\left(f_p(\underline{x},\underline{\theta}) \geq 0\right) - \alpha_p \end{array} \right.$$

The reliability constraint is :

$$\begin{cases} \mathbb{P}\left(f_1(\underline{x},\underline{\theta}) \geq 0\right) \geq \alpha_1 \\ \dots \\ \mathbb{P}\left(f_p(\underline{x},\underline{\theta}) \geq 0\right) \geq \alpha_p \end{cases}$$

Example : $f(x,\Theta) = [x - \Theta, x - \Theta^2]$, $\Theta \sim \mathcal{N}(1,1)$ and $\alpha = 0.95$.

```
>>> thetaDist = ot.Normal(1.0, 1.0)
>>> f_base = ot.SymbolicFunction(['x', 'theta'],  ['x-theta', 'x-theta^2'])
>>> f = ot.ParametricFunction(f_base, [1], thetaDist.getMean())
>>> measure = otrobopt.IndividualChanceMeasure(f, thetaDist, ot.GreaterOrEqual(), [0.95, 0.5])
```



2970 calls to f to compute $M_{f,\mathcal{D}}(2.0)$.

Beware : What is the nonadmissibility level of $\underline{x}^*_{rob}$? the probability to satisfy all the constraints is $\leq \min \mathbb{P}\left(f_i(\underline{x},\underline{\Theta}) \geq 0\right)$.

Aggregated measures

One can stack different measures :

- Robustness measures $\longrightarrow$ for multiobjective optimization
- Reliability measures $\longrightarrow$ for multiconstrained optimization

One creates the measure

$$M_{(f_1,\ldots,f_K),\mathcal{D}}(\underline{x}) = \left| \begin{array}{l} M_{f_1,\mathcal{D}}(\underline{x}) \\ \ldots \\ M_{f_K,\mathcal{D}}(\underline{x}) \end{array} \right.$$

Beware : All the individual measures share the *same* distibution $\mathcal{D}$!

```
-----------------------------------------------------------------
>>> meas1 = otrobopt.QuantileMeasure(f1, thetaDist, q)
>>> meas2 = otrobopt.MeanMeasure(f2, thetaDist)
>>> meas2 = otrobopt.JointChanceMeasure(f3, thetaDist, alpha)
>>> measure = otrobopt.AggregatedMeasure([meas1, meas2, meas3])
>>> ot.Show(measure.getMarginal(0).draw(-2.0, 2.0, 128))
-----------------------------------------------------------------
```

Robust optimization with OpenTURNS

Classification of robust optimization problems

Specific names for specific *robust optimisation* problems :

$\lambda - \rho$	Measure	Function
Measure	Robustness problem with reliability constraint	Reliability problem
Function	Constrained robustness problem	$\times$
$\emptyset$	Robustness problem	$\times$

These names correspond to the following mathematical formulations :

$\lambda - \rho$	Measure	Function
Measure	$\operatorname{argmin}_{\begin{cases}\lambda_{G,\mathcal{D}}(\underline{x}) \geq 0 \\ h(\underline{x}) = 0 \\ \underline{a} \leq \underline{x} \leq \underline{b}\end{cases}} \rho_{J,\mathcal{D}}(\underline{x})$	$\operatorname{argmin}_{\begin{cases}\lambda_{G,\mathcal{D}}(\underline{x}) \geq 0 \\ h(\underline{x}) = 0 \\ \underline{a} \leq \underline{x} \leq \underline{b}\end{cases}} J(\underline{x})$
Function	$\operatorname{argmin}_{\begin{cases}G(\underline{x}) \geq 0 \\ h(\underline{x}) = 0 \\ \underline{a} \leq \underline{x} \leq \underline{b}\end{cases}} \rho_{J,\mathcal{D}}(\underline{x})$	$\times$
$\emptyset$	$\operatorname{argmin} \rho_{J,\mathcal{D}}(\underline{x})$	$\times$

Instantiation with *otrobopt*

The **OptimizationProblem** class (OpenTURNS) allows to define any classical optimization problem.
The **RobustOptimizationProblem** class (*otrobopt*) allows to define a robust optimization problem.
To define :

$$(S_{rob}) : \quad x^* = \underset{\substack{\lambda_{G,\mathcal{D}}(\underline{x}) \geq 0 \\ h(\underline{x}) = 0 \\ \underline{a} \leq \underline{x} \leq \underline{b}}}{\text{argmin}} \quad \rho_{J,\mathcal{D}}(\underline{x})$$

with additional deterministic constraints (here, bound contraints and equality constraints), one write :

```
------------------------------------------------------------------------------------------
>>> myRobProblem = otrobopt.RobustOptimizationProblem(robustnessMeas, reliabilityMeas)
>>> myRobProblem.setEqualityConstraint(h)
>>> myRobProblem.setBounds(Interval(a, b))
------------------------------------------------------------------------------------------
```

Robust optimization with OpenTURNS

How to solve a robust optimization problem ? I

$$(S_{rob}) : x^* = \underset{\substack{\lambda_{G,\mathcal{D}}(\underline{x}) \geq 0 \\ h(\underline{x}) = 0 \\ \underline{a} \leq \underline{x} \leq \underline{b}}}{\text{argmin}} \; \rho_{J,\mathcal{D}}(\underline{x})$$

General comments :

- If $dim(\underline{\theta}) = 1$: the adaptive quadrature methods used to compute integrales are both accurate (error of order 10^{-10}) and reasonably fast (hundreds of calls). It may be reasonable to solve (S_{rob}) as a classical optimization problem :

  ```
  ---------------------------------------
  >>> algo = ot.Cobyla(myRobProblem)
  >>> algo.setStartingPoint([startPoint])
  >>> algo.run()
  ---------------------------------------
  ```

  OpenTURNS has many different algorithms to solve constrained/unconstrained optimization problems, see eg. the OptPP, NLopt, DLib, Ceres classes.
- If $dim(\underline{\theta}) \geq 2$: the adaptive quadrature methods are expansive. *otrobopt* proposes two algorithms to solve robust optimization problems :
  1. The plugin method ;
  2. A sequential Monte Carlo method.

The plugin method I

Given a sampling size $N > 0$, the plugin method build a discretized approximation $(S_{rob}^{(N)})$ of (S_{rob}) where $\mathcal{D}$ is replaced by a discrete distribution with N values :

$$(\boldsymbol{S_{rob}}) : \quad x^* = \underset{\begin{cases} \lambda_{G,\mathcal{D}}(\underline{x}) \geq 0 \\ h(\underline{x}) = 0 \\ \underline{a} \leq \underline{x} \leq \underline{b} \end{cases}}{\text{argmin}} \rho_{J,\mathcal{D}}(\underline{x}) \quad \Longrightarrow \quad (\boldsymbol{S_{rob}^{(N)}}) : \quad x_N^* = \underset{\begin{cases} \lambda_{G,\mathcal{D}^N}(\underline{x}) \geq 0 \\ h(\underline{x}) = 0 \\ \underline{a} \leq \underline{x} \leq \underline{b} \end{cases}}{\text{argmin}} \rho_{J,\mathcal{D}^N}(\underline{x})$$

Remarks :

- The integrals wrt $\underline{\theta}$ are replaced by finite weighted sums $\sum_{i=1}^N$. The resulting optimization problem can then be solved using one of the *classical algorithms*.
- The discrete distribution $\mathcal{D}^N$ is given by : $\mathcal{D}^N = \sum_{i=1}^N \omega_i \delta_{\underline{\theta}_i}$ were $(\omega_i, \underline{\theta}_i)_i$ are *WeightedExperiment : MonteCarloExp, LHSExp, ImportanceSamplingExp, GaussProductExp*, ...
- Even if $\mathcal{D}$ is already discrete, one can gain a lot by discretizing $\mathcal{D}$: it can reduce a lot the number of values of $\mathcal{D}$, thus the number of evaluations of the underlying parametric functions.

The plugin method II

To solve (S_{rob}) using the plugin method, the measures are discretized using the *MeasureFactory* class. The problem is then solved by one of the standard optimization package.

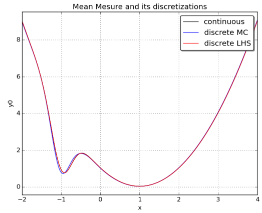
Example : $\mathcal{D}$ is discretized using an LHS design with 50 points.

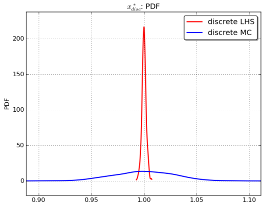
```
-------------------------------------------------------------------------------------------
>>> robustnessMeasure = otrobopt.MeanMeasure(f, thetaDist)
>>> reliabilityMeasure = otrobopt.JointChanceMeasure(g, thetaDist, ot.Greater(), 0.95)
>>> N = 50
>>> experiment = ot.LHSExperiment(N)
>>> factory = otrobopt.MeasureFactory(experiment)
>>> collDiscMeas = factory.buildCollection([robustnessMeasure, reliabilityMeasure])
>>> myPlugInProblem = otrobopt.RobustOptimizationProblem(collDiscMeas[0], collDiscMeas[1])
>>> algo = ot.Cobyla(problem)
-------------------------------------------------------------------------------------------
```

Beware : Use the same discretization $\mathcal{D}^N$ for all the measures !
Example : Additive noise : $h(x) = (x-1)^2 - 5e^{-20(2+x)^2}$, $f(x,\Theta) = h(x+\Theta)$ with $\Theta \sim \mathcal{N}(0, 0.2)$.
One choose the robustness measure $\rho_{f,\mathcal{D}}(\underline{x}) = \mathbb{E}_{\mathcal{D}}[f(x,\theta)]$.
Using the Gauss-Kronrod quadrature, we get the following reference solution $\boldsymbol{x^*_{ref} = 10^{-5}}$.
We discretize $\mathcal{D}$ using different sampling methods :

The plugin method III

- $N = 50$ using either LHS or Monte Carlo : the sampling and the resolution are done $N_{simu} = 10^6$ to capture the distribution of x^*_{disc}
- Gauss-Product : The number of integration nodes is increased (from 1 to 7).





N	x^*_{GP}
1	$4.76e-6$
3	$1.03e-5$
5	$1.03e-5$
7	$1.03e-5$

Sequential Monte Carlo algorithm I

The sequential Monte Carlo algorithm proceeds by successive approximations $(S_{rob}^{(N_k)})$ of (S_{rob}). This is done by replacing $\mathcal{D}$ by a discrete counterpart $\mathcal{D}^{N_k}$ which is updated sequentially using a sample of size N_k, with $N_k \to +\infty$ when $k \to +\infty$:

$$\boldsymbol{(S_{rob}) :} \qquad\qquad\qquad\qquad \textbf{Iteration } k : (S_{rob}^{(N_k)}) :$$

$$x^* = \underset{\begin{cases} \lambda_{G,\mathcal{D}}(\underline{x}) \geq 0 \\ h(\underline{x}) = 0 \\ \underline{a} \leq \underline{x} \leq \underline{b} \end{cases}}{\text{argmin}} \rho_{J,\mathcal{D}}(\underline{x}) \Longrightarrow x_k^* = \underset{\begin{cases} \lambda_{G,\mathcal{D}^{N_k}}(\underline{x}) \geq 0 \\ h(\underline{x}) = 0 \\ \underline{a} \leq \underline{x} \leq \underline{b} \end{cases}}{\text{argmin}} \rho_{J,\mathcal{D}^{N_k}}(\underline{x})$$

Each approximate problem $(S_{rob}^{(N_k)})$ is solved up to a precision ε_k, with $\varepsilon_k \to 0$ when $k \to +\infty$.

Convergence theorem : The sequence $x_k^* \to x^*$ when $k \to +\infty$ if $\varepsilon_k \to 0$ and $N_k \to +\infty$ as soon as ρ is strongly convex and λ defines a strongly convex set (cf [7]).
The iterations proceed this way. We denote by S_k the sample of $\underline{\Theta}$ used to discretize $\mathcal{D}$ at the k-th iteration.

Sequential Monte Carlo algorithm II

At iteration $k+1$, one takes :

$$\begin{cases} \underline{x}_{k+1}^{0} = \underline{x}_{k}^{*} \\ N_{k+1} = 2N_k \\ S_k \subset S_{k+1} \\ \varepsilon_{k+1} = \dfrac{\varepsilon_0}{\sqrt{N_{k+1}}} \end{cases}$$

Roughly speaking, this method make the bet that if $\boldsymbol{\underline{x}_k^* \in \mathcal{DA}(\underline{x}^*)}$, then $\boldsymbol{\underline{x}_{k+1}^* \in \mathcal{DA}(\underline{x}^*)}$. So the **choice of the starting point** $\underline{x}_0^*$ is a key element of the method when used outside of the hypotheses of the theorem !
The initial step is here to put the algorithm on the good convergence path. **For step $k=0$**, one seek a solution $\underline{x}_0^*$ of $(S_{rob}^{(N_0)})$ in the attraction set of the global optimum $\underline{x}^*$.
In the current implementation of the algorithm, this globalization search is done using **multi start** :

1. We choose N_{start} starting points de départ répartis selon un plan LHS (loi uniforme sur un domaine borné $[\underline{a}, \underline{b}]$) : $(\underline{x}_{0,1}, \dots, \underline{x}_{0,N_{start}})$
2. We compute the N_{start} solutions of $(S_{rob}^{(N_0)})$: $(\underline{x}_{0,1}^*, \dots, \underline{x}_{0,N_{start}}^*)$;

Sequential Monte Carlo algorithm III

3. We select the best solution $\underline{x}_0^* = \text{argmin}\, q = 1, \ldots, N_{start} \rho_{J,\mathcal{D}^{N_k}}(\underline{x}_{0,q}^*)$

Remarks :

- All the starting points $\underline{x}_{0,q}^*$ are admissible.
- Beware, this step may be very expansive !
- Under the hypotheses of the theorem, on can choose $N_{start} = 1$!

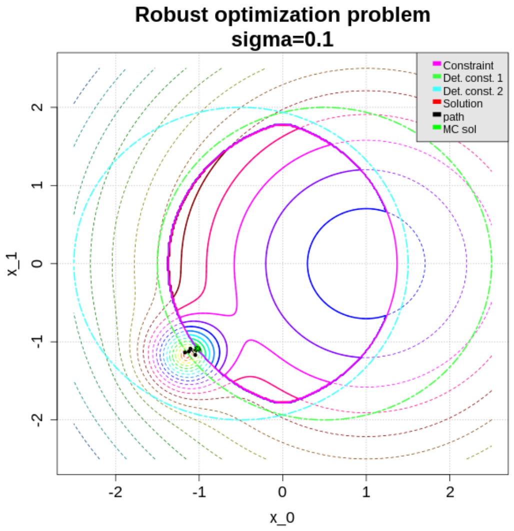
Sequential Monte Carlo algorithm IV

With otrobopt : *SequentialMonteCarloRobustAlgorithm*.

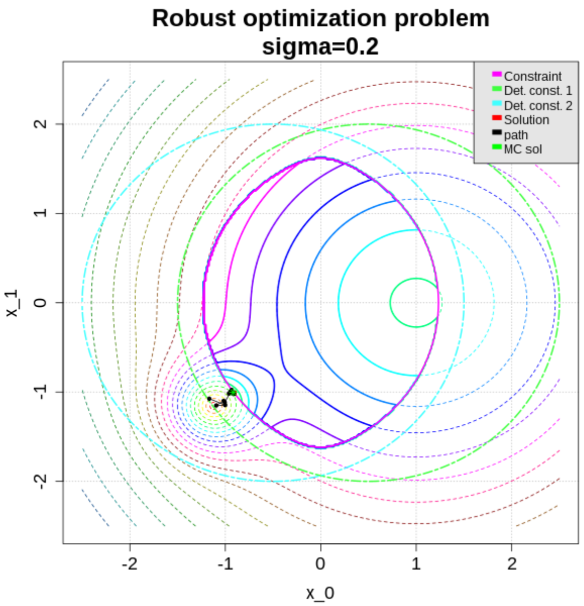
```
----------------------------------------------------------------------------------------------
>>> robustnessMeasure = otrobopt.MeanMeasure(f, thetaDist)
>>> reliabilityMeasure = otrobopt.JointChanceMeasure(g, thetaDist, ot.Greater(), 0.95)
>>> myRobProblem = otrobopt.RobustOptimizationProblem(robustnessMeasure, reliabilityMeasure)
>>> mySolver = ot.Cobyla()
>>> algo = otrobopt.SequentialMonteCarloRobustAlgorithm(myRobProblem, mySolver)
>>> algo.run()
----------------------------------------------------------------------------------------------
```

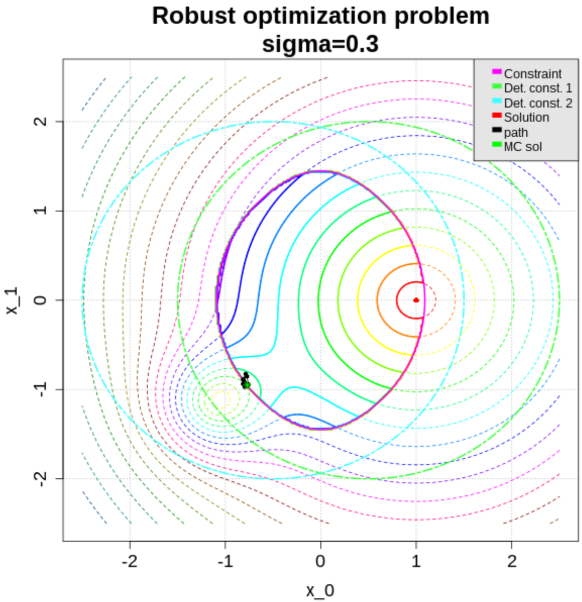
Example : random noise. We take an initial sampling size $N_0 = 2$ to discretize $\mathcal{D}$, then the size is doubled at each iteration.

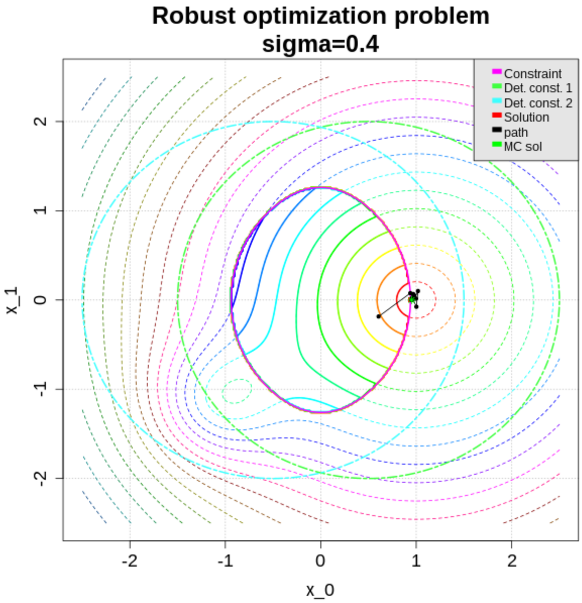
Sequential Monte Carlo algorithm V

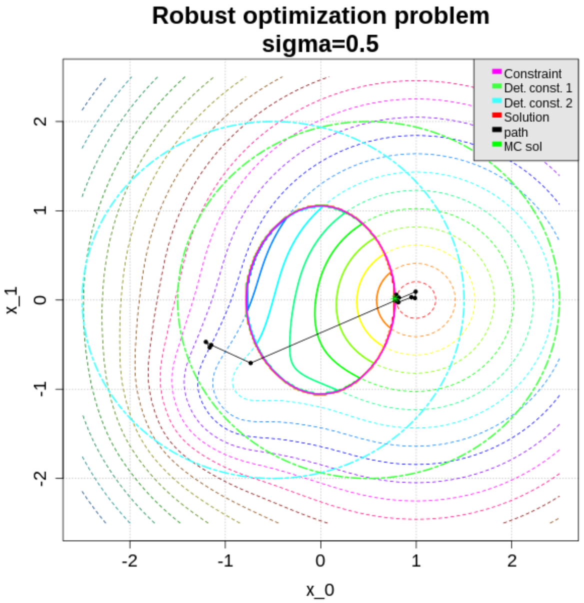


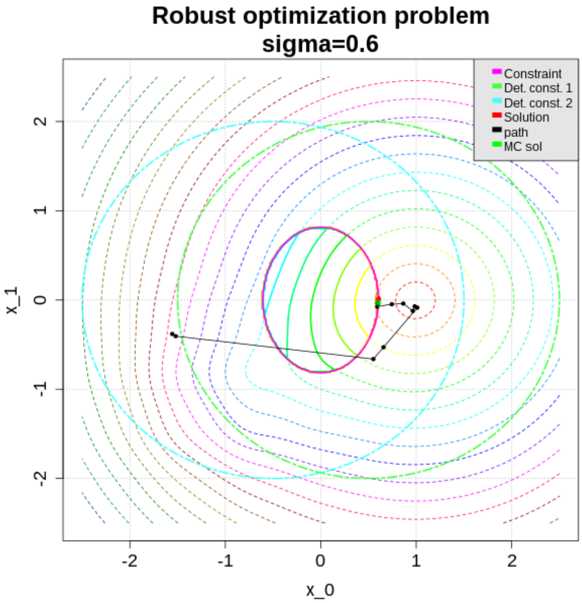
Remark : the admissible set changes with the discretization of $\mathcal{D}$. Only the final admissible set is shown on the figure.


Robust optimization problem
sigma=0.2
Constraint
Det. const. 1
Det. const. 2
Solution
path
MC sol
x_1
x_0
-2
-1
0
1
2


Robust optimization problem
sigma=0.3
Constraint
Det. const. 1
Det. const. 2
Solution
path
MC sol
x_1
x_0
-2
-1
0
1
2


Robust optimization problem
sigma=0.4
Constraint
Det. const. 1
Det. const. 2
Solution
path
MC sol
x_1
x_0
2
1
0
-1
-2
-2
-1
0
1
2


Robust optimization problem
sigma=0.5
🟪 Constraint
🟩 Det. const. 1
🟦 Det. const. 2
🟥 Solution
⬛ path
🟩 MC sol
x_1
x_0
-2
-1
0
1
2



Robust optimization problem
sigma=0.6
🟪 Constraint
🟩 Det. const. 1
🟦 Det. const. 2
🟥 Solution
⬛ path
🟩 MC sol
x_1
x_0
2
1
0
-1
-2
-2
-1
0
1
2

Robust optimization with OpenTURNS

Presentation I

We propose a four - step methodology, in order to organize and to sum - up the different aspects we presented on how to take into account uncertainties in an industrial optimization process :

- Step 1 : Problem Specification
- Step 2 : Robustness & Risk formulation
- Step 3 : Resolution
- Step 4 : Feedback Analysis

Step 1 : Problem Specification I

This is the very first step of the methodology. Its goal is to gather all the information to set up the robust optimization problem, to identify the sources of uncertainty, and also the expected behaviour of the system regarding the uncertainties in order to select later the relevant measures of robustness and reliability.
Here are the information about the different functions appearing in ?? :

- What are the cost/objective function(s) J to minimize/maximize ?
- What are the constraint functions $G = (g_1, \dots, g_M)$?
- What is the mathematical and computational complexity of these functions (nonlinearity, smoothness, availability of gradients, CPU cost, approximations, etc.) ?

Here are the information about the quantification of the uncertainties :

- What are the variables that would be subjected to uncertainties ?...
- Needs in terms of robustness : which behaviour (regarding uncertainties) do we want to avoid in the cost function ?...
- Needs in terms of risk and Reliability : which behaviour (regarding uncertainties) do we want to avoid in the constraints ?...

Step 2 : Robustness & Risk formulation I

The goal of this step is to select the relevant measures of robustness and reliability.

- What is the joint distribution of the uncertainties and how is it assessed (statistical data, engineering model) ?
- Which robustness measure ρ_J is relevant to express the industrial needs regarding the robustness of the design ?
- Which reliability measure ρ_G is relevant according to the internal or external regulation or to a given reliability target. At this point it is always possible to introduce a worst - case approach by specifying a joint probability constraint of level 1.

Step 3 : Resolution I

The goal of this step is to define a numerical strategy to solve the robust optimization problem efficiently given a level of precision.

- What are the computing resources (memory, cores, etc.) ?
- What is the software availability ?
- What is the error tolerance allowed ?
- Define a numerical strategy to compute the robust version of the objective function $\rho_{\mathcal{J}}(\mathcal{J}(\mathbf{x}, \omega))$
- Define a numerical strategy to explore the domain $\mathcal{D}_\alpha$ or $\mathcal{D}_{\underline{\beta}}$

It exists several algorithmic strategies to solve the problem, one of the most versatile is the sequential Monte Carlo approach.
The key point is that the sample counterparts of the robustness and reliability measures have the same smoothness as the true ones, so there is no noise effect as if the sample was redrawn for each value of $\underline{x}$.
Many other options are available to compute approximations of the robustness and reliability methods, such as the perturbation methods, the asymptotic approximations FORM and SORM. . .

Step 4 : Analysis I

The goal of this step is the qualification of the generated optimum.

- We analyse the robustness of the solution through the value of ρ_J at the optimum and see if the optimal design is good with respect to the performance target ;
- We analyse the reliability of the solution, in particular the reliability gap in case of disjoint chance constraints : what are the inactive constraints and what is the probability gap, to be compared with the sampling error.
- We analyze the sensitivity of the optimum w.r.t the uncertainties : does a small change in the distribution generates a large change in the robust optimization problem ? This last step is much less expansive than a full robust optimization resolution as all the analysis is done for $\underline{x} = \underline{x}^*$.
- The expected feedbacks are :
  - An improvement in the model $\mathcal{J}$ and/or the constraints G (high fidelity model, etc.) $\rightarrow$ return to Step 1 "Problem Specification"
  - A change in the formulation (robustness, reliability, confidence, etc.) $\rightarrow$ return to Step 2 "Robustness & Reliability"

Robust optimization with OpenTURNS

Conclusion I

The introduction of uncertainties in an industrial optimization problem can be done using a probabilistic approach in an efficient and structured way. It allows to quantify the sources of uncertainty and to propagate them into the optimization process in a mathematically founded way.
The drawback is that for many problems, the numerical cost of solving the resulting robust optimization problem is significantly higher than the cost of solving one deterministic optimization problem. In order to solve this issue, one has to resort to meta - models, which introduces an additional source of uncertainties which has to be taken into account.
In its final form, the robust optimization problem cannot be solved efficiently using a blind approach in reusing of - the - shelf optimization libraries. An adaptation of the numerical methods is needed in order to recover the smoothness needed by most of these libraries.
The very good point is that when these difficulties have been solved, one get a solution of the optimization problem with uncertainties wich has a direct interpretation in terms of reliability and robustness, the main objective of the optimization under uncertainties.

References

- Airbus Group. "MDO/MDA using simplified models", Technical Report, 2014.
- Airbus Group, "Report on the Implementation of the Sparse Polynomial Chaos Expansion Method", UMRIDA Technical Report D2.1 - 18, 2015.
- Ben - Tal, A., El Ghaoui, L., Nemirovski, A., "Robust Optimization", Princeton University Press, 2009.
- Forrester, A., Sobester, A., Keane, A., "Engineering Design via Surrogate Modelling : A Practical Guide", Wiley, 2008.
- IRT. "Méthodologie et formulation de problèmes d'optimisation robuste", Technical Report, 2014.
- Prekopa, A., "Stochastic Programming", Springer, 1995.
- Shapiro, A., Dentcheva, D. and Ruszczynski, A., "Lectures on Stochastic Programming : Modeling and Theory", SIAM, 2009.