Robust Design and Uncertainty Quantification

Monte Carlo, Latin hypercube and Polynomial Chaos
Why robust designs optimizations?

- In many engineering problems **design parameters** may be **uncertain** (e.g. tolerances, fluctuating operating conditions, etc.)

- Input parameters are then defined not by a deterministic value, but as a **Distribution** (each value have a statistical probability to occur)

  e.g. $X=(10 +/- 0.3)\text{mm}$
Why robust designs optimizations?

- The Input parameters **uncertainty** is reflected in the **outputs** of the system: a solution good for a deterministic value of inputs, may be not **robust** for slight variations.

- The **robustness** of the solution is defined as the characteristic of the system response to be insensitive to the variation of the input parameters.

- A **Robust Design Optimisation** searches for Robust solutions.

![Diagram showing best solutions with and without robustness consideration](image)
The Problem

Ordinary (MO) Optimization Problem:

$$\min_{x \in \mathbb{R}^n} (f_1(x), \ldots, f_k(x))$$

How can we define a stochasticized (MO) optimization problem?

Assume $f(x) = f(x_1, \ldots, x_n)$ is still deterministic (as a computer experiment)

Otherwise $x_1, \ldots, x_n$ are affected by uncertainty, i.e.,

$$x_i \mapsto X_i$$

random variable

e. g.:

$$X_i = X_i(x_i)$$ is Normal centered in $x_i$ with fixed $\sigma_i$
A stochastic MOP

Also $f$ is substituted by a random variable, defined as:

$$f(x_1, \ldots, x_n) \longrightarrow F(x_1, \ldots, x_n),$$

random variable

$$F(x_1, \ldots, x_n) \overset{\text{def}}{=} f(X_1(x_1), \ldots, X_n(x_n))$$
Stochastic MOP

Simply rewriting

$$\min_{x \in \mathbb{R}^n} (F_1(x), \ldots, F_k(x))$$

does not make sense.

We could consider, for instance,

$$\min_{x \in \mathbb{R}^n} \left( \mu_{F_1(x)}, \ldots, \mu_{F_k(x)} \right)$$

plainly, or more sophisticatedly,

$$\min_{x \in \mathbb{R}^n} \left( \mu_{F_1(x)} + 3\sigma_{F_1(x)}, \ldots, \mu_{F_k(x)} + 3\sigma_{F_k(x)} \right)$$

This means that, first of all, we have a problem of uncertainty quantification.
Uncertainty Quantification Comparisons

Multiobjective Robust Design Optimization (MORDO)

searches for the optima of the mean and standard deviation of a stochastic response rather than the optima of the deterministic response (the output from the solver)
Uncertainty Quantification

- How can we quantify uncertainty?

1. Analytically
2. Monte Carlo
3. Latin Hypercube
4. Polynomial Chaos
5. ...
Analytically

• There exists several special cases in which the statistical moments can be determined analytically on the basis of the statistical moments of the independent uncertain variables $X_1, \ldots, X_k$.

• Consider for instance the trivial case:

  • $Y = f(X) := 3X$
  • It is clear that $E[Y] = 3E[X]$
  • and that $E[Y^2] = 9E[X^2]$, then $\sigma_Y = 3\sigma_X$

• solving the uncertainty quantification analytically may be difficult (or even impossible) for more complex functions and, moreover, the function should be known
Monte Carlo

The most classical methodology consists in drawing a random sample \( \{x^{(1)}, \ldots, x^{(N)}\} \) from the joint distribution \( D_1 \otimes \cdots \otimes D_d \), i.e., drawing \( N \) random numbers \( \{x^{(1)}_1, \ldots, x^{(N)}_i\} \) from every distribution \( D_i \) and then collecting the vectors

\[
\begin{align*}
x^{(1)} & := \begin{pmatrix} x^{(1)}_1, \ldots, x^{(1)}_d \end{pmatrix}^T, \\
x^{(2)} & := \begin{pmatrix} x^{(2)}_1, \ldots, x^{(2)}_d \end{pmatrix}^T, \\
\vdots & \\
x^{(N)} & := \begin{pmatrix} x^{(N)}_1, \ldots, x^{(N)}_d \end{pmatrix}^T.
\end{align*}
\]

A sample \( s_Y := \{y^{(1)}, \ldots, y^{(N)}\} \) of the uncertain dependent variable \( Y \) is computed through application of \( f \) to each one of the \( x^{(j)} \):

\[
y^{(j)} := f \left( x^{(j)}_1, \ldots, x^{(j)}_d \right), \quad j = 1, \ldots, N.
\] (2)
Monte Carlo

- Monte Carlo is a robust method
- Statistics obtained via Monte Carlo are reliable
- but very poor in accuracy
- Too many points are needed

\[ m_Y \simeq \bar{m}_Y := \langle s_Y \rangle = \frac{1}{N} \sum_{j=1}^{N} y^{(j)}, \]

\[ \sigma_Y \simeq \bar{\sigma}_Y := \sqrt{\frac{1}{N-1} \sum_{j=1}^{N} (y^{(j)} - \bar{m}_Y)^2}. \]
Monte Carlo Accuracy

- The convergence of the method is
  \[
  \frac{1}{\sqrt{N}}
  \]

- To halve the estimation error it is necessary a four times larger sample, to reduce of an order of magnitude it is necessary to take a sample 100 times larger.
Example

Let \( Y = f(X) \), where

\[
f(x) := 20 \left( e^{-\frac{(x+2.7)^2}{2}} + e^{-\frac{(x-2.7)^2}{2(2^2)}} \right),
\]

\( X \sim \mathcal{N}(2, 0.8) \), i.e., \( w(x) = e^{-\frac{(x-2)^2}{2(0.8^2)}} \).

\[
m_Y = E[f(X)] = \int f(x)w(x) \, dx \approx 3.5209849377883446,
\]

\[
\sigma_Y = \sqrt{E[(f(x) - m_Y)^2]} = \sqrt{\int f(x)^2w(x) \, dx - m_Y^2} \approx 0.507313175.
\]
Monte Carlo sampling

Statistics can be computed via Monte Carlo, i.e., *drawing a random sample* from the assigned stochastic distributions from the input variables.

Monte Carlo converges slowly to true values statistics.

Large samples are needed to reach reliable results.

**Former example:**

to reach *1% Error* in

*Mean needs 256 points*

*Standard Deviation needs 8192 points*
Monte Carlo Accuracy

Monte Carlo Simulation

<table>
<thead>
<tr>
<th>nData</th>
<th>Average Error Mean</th>
<th>Average Error StDev</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0.1352321821576068</td>
<td>0.1610272359435758</td>
</tr>
<tr>
<td>16</td>
<td>0.1171965574510605</td>
<td>0.0864659032341357</td>
</tr>
<tr>
<td>32</td>
<td>0.059222020366550845</td>
<td>0.04713102966918838</td>
</tr>
<tr>
<td>64</td>
<td>0.05778452160956625</td>
<td>0.05253445820159186</td>
</tr>
<tr>
<td>128</td>
<td>0.03838430554585017</td>
<td>0.034136016860713006</td>
</tr>
<tr>
<td>256</td>
<td>0.022344036432755</td>
<td>0.020111546424191107</td>
</tr>
<tr>
<td>512</td>
<td>0.02061047535304621</td>
<td>0.015098327730704386</td>
</tr>
<tr>
<td>1024</td>
<td>0.009641651869702561</td>
<td>0.010470922893152998</td>
</tr>
<tr>
<td>2048</td>
<td>0.010312701913743005</td>
<td>0.008504295088329538</td>
</tr>
<tr>
<td>4096</td>
<td>0.007855294629528543</td>
<td>0.006264945824679183</td>
</tr>
<tr>
<td>8192</td>
<td>0.003879754790574208</td>
<td>0.0040735989911990089</td>
</tr>
<tr>
<td>16384</td>
<td>0.0029156602991482483</td>
<td>0.0034092125680575958</td>
</tr>
</tbody>
</table>
Latin Hypercube

• To speed up the convergence of statistical moments to actual moments, there exists a smarter strategy: a Latin Hypercube Sampling (LHS)
• Latin Hypercubes is a sampling strategy derived from *stratified samplings*
• If LHS is composed of N points, and every variable is divided into N stratum with equal probability, every single stratum will be occupied by exactly one point.
Latin hypercube sampling

The random sample can be substituted by a stratified sampling as *Latin hypercube*. Statistics converge faster to exact values.

Medium-large samples are needed to reach good results.

*Former example:*
to reach *1% Error in Mean* needs *16 points*

*Standard Deviation* needs *128 points*
LHS Accuracy

Table of results for the example

<table>
<thead>
<tr>
<th>nData</th>
<th>Average Error Mean</th>
<th>Average Error StDev</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0.03838708340684853</td>
<td>0.0823640656227671</td>
</tr>
<tr>
<td>16</td>
<td>0.03056795021135484</td>
<td>0.040710534223220365</td>
</tr>
<tr>
<td>32</td>
<td>0.00751863928251643</td>
<td>0.01648128041162316</td>
</tr>
<tr>
<td>64</td>
<td>0.003418078142074843</td>
<td>0.006587126182655906</td>
</tr>
<tr>
<td>128</td>
<td>0.0024618618972161777</td>
<td>0.004848681021425105</td>
</tr>
<tr>
<td>256</td>
<td>0.0018873455437873998</td>
<td>0.003133046085056787</td>
</tr>
<tr>
<td>512</td>
<td>5.648087501843202E-4</td>
<td>0.0012508917015900789</td>
</tr>
<tr>
<td>1024</td>
<td>4.2176395159760905E-4</td>
<td>0.001042445154700511</td>
</tr>
<tr>
<td>2048</td>
<td>2.665184159121647E-4</td>
<td>4.888817764855613E-4</td>
</tr>
<tr>
<td>4096</td>
<td>1.1563639765082012E-4</td>
<td>2.1436136917031833E-4</td>
</tr>
<tr>
<td>8192</td>
<td>4.724986318238589E-5</td>
<td>7.123204436120126E-5</td>
</tr>
<tr>
<td>16384</td>
<td>3.0972434058185175E-5</td>
<td>4.675409217411719E-5</td>
</tr>
</tbody>
</table>
Polynomial Chaos

- Polynomial Chaos is very efficient
- This methodology originates from the work of Norbert Wiener and is called **Polynomial Chaos Expansion**.
- This methodology consists essentially in expanding the uncertain variable in a **suitable series** and then **determine analytically the statistical moments** of the truncated expansion.

\[
f(x) := \sum_{i=0}^{k} \alpha_i p_i(x).
\]

\[
\langle p_i(x), p_j(x) \rangle_w := \int p_i(x)p_j(x)w(x)dx = 0, \quad \text{whenever } i \neq j.
\]
Polynomial Chaos Expansion

Alternatively, it is possible to expand the response in a special series, a *Polynomial Chaos*.

\[
f(x) \approx \sum_{i=0}^{k} \alpha_i He_i(x)
\]

Mean\(Y\) \(\approx \alpha_0\)

\[
\sigma^2(Y) \approx \sum_{i=0}^{k} \alpha_i^2(i!)
\]

The statistics (mean and standard dev) of a Polynomial Chaos are computed analytically.

Estimate is extremely precise and sample are very small.

Sample size depend on the number of stochastic variables (nVar) and on the order of the expansion.

\[
\# \{\text{sample}\} \geq \frac{(nVar + \text{order})!}{nVar!\text{order}!}
\]
Polynomial Chaos

The m-th moment may be written as:

\[
\langle y^m \rangle = \int (f(x))^m w(x) dx = \int \left( \sum_{i=0}^{k} \alpha_i p_i(x) \right)^m w(x) dx = \\
= \int \sum \alpha_i p_{i_1}(x) \ldots p_{i_m}(x) w(x) dx = \\
= \sum \alpha_{i_1} \ldots \alpha_{i_m} \int p_{i_1}(x) \ldots p_{i_m}(x) w(x) dx.
\]

If the distribution is Gaussian we can use the Hermite polynomials and compute the moments as:

\[
m_Y = \alpha_0.
\]

\[
\sigma_Y = \sqrt{\sigma_Y^2} = \sqrt{\langle Y^2 \rangle - m_Y^2}.
\]
Polynomial Chaos

- Wiener-Askey scheme on relations between probability distributions and orthogonal polynomials
- In the case of multiple variables, assuming they are independently distributed, it is possible to write “multivariate chaos” considering the tensorial products of the univariate polynomials.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Probability Density</th>
<th>Orthogonal Polynomials</th>
<th>Support Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>( \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}} ) ( \frac{1}{2} )</td>
<td>Hermite ( H_n(x) )</td>
<td>((-\infty, +\infty))</td>
</tr>
<tr>
<td>Uniform</td>
<td>( (1-x)^{\alpha}(1-x)^{\beta} ) ( \frac{2^{\alpha+\beta+1}B(\alpha+1,\beta+1)}{e^{-x}} )</td>
<td>Legendre ( P_n(x) )</td>
<td>((-1, +1))</td>
</tr>
<tr>
<td>Beta</td>
<td>( x^{\alpha}e^{-x} ) ( \frac{1}{\Gamma(\alpha+1)} )</td>
<td>Jacobi ( P_n^{(\alpha,\beta)}(x) )</td>
<td>((-1, +1))</td>
</tr>
<tr>
<td>Exponential</td>
<td></td>
<td>Laguerre ( L_n(x) )</td>
<td>(0, +\infty)</td>
</tr>
<tr>
<td>Gamma</td>
<td></td>
<td>Generalized Laguerre ( L_n^{(\alpha)}(x) )</td>
<td>(0, +\infty)</td>
</tr>
</tbody>
</table>
Computing the coefficients

- To find the coefficients we have to solve a single objective optimization problem (e.g. Levenberg–Marquardt)

\[ \alpha_i : \min_{\alpha_i} \sum_{j=1}^{N} \left| f(x_j) - \sum_{i=1}^{k} \alpha_ip_i(x_j) \right|^2, \]
\[ \{x_1, \ldots, x_N\}, \text{ arbitrary sample} \]

- The size N of the sample has to be at least equal or larger than the number of parameters \( N \geq \frac{(k+d)!}{k!d!} \)
Number of parameters in a chaos of order $k$ in $d$ variables, i.e., minimum size of a sample to be employed for chaos collocation.

$$N \geq \frac{(k+d)!}{k!d!}$$

<table>
<thead>
<tr>
<th>$d$</th>
<th>$k = 1$</th>
<th>$k = 2$</th>
<th>$k = 3$</th>
<th>$k = 4$</th>
<th>$k = 5$</th>
<th>$k = 6$</th>
<th>$k = 7$</th>
<th>$k = 8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>6</td>
<td>10</td>
<td>15</td>
<td>21</td>
<td>28</td>
<td>36</td>
<td>45</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>10</td>
<td>20</td>
<td>35</td>
<td>56</td>
<td>84</td>
<td>120</td>
<td>165</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>15</td>
<td>35</td>
<td>70</td>
<td>126</td>
<td>210</td>
<td>330</td>
<td>495</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>21</td>
<td>56</td>
<td>126</td>
<td>252</td>
<td>462</td>
<td>792</td>
<td>1287</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>28</td>
<td>84</td>
<td>210</td>
<td>462</td>
<td>924</td>
<td>1716</td>
<td>3003</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>36</td>
<td>120</td>
<td>330</td>
<td>792</td>
<td>1716</td>
<td>3432</td>
<td>6435</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>45</td>
<td>165</td>
<td>495</td>
<td>1287</td>
<td>3003</td>
<td>6435</td>
<td>12870</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>55</td>
<td>220</td>
<td>715</td>
<td>2002</td>
<td>5005</td>
<td>11440</td>
<td>24310</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>66</td>
<td>286</td>
<td>1001</td>
<td>3003</td>
<td>8008</td>
<td>19448</td>
<td>43758</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>78</td>
<td>364</td>
<td>1365</td>
<td>4368</td>
<td>12376</td>
<td>31824</td>
<td>75582</td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>91</td>
<td>455</td>
<td>1820</td>
<td>6188</td>
<td>18564</td>
<td>50388</td>
<td>125970</td>
</tr>
</tbody>
</table>
Polynomial Chaos Accuracy

Table of results for the example

<table>
<thead>
<tr>
<th>Chaos Order</th>
<th>Sample Size</th>
<th>Average Error Mean</th>
<th>Average Error StDev</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>0.09344069282015036</td>
<td>0.10429522822805484</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>0.019326055906163126</td>
<td>0.0697189377206368</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>0.01271788209210331</td>
<td>0.02654736033749695</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>8.034675780786764E-4</td>
<td>0.001684371972847451</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>7.87737893433344E-4</td>
<td>0.002864232049342065</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td>6.138461875910162E-4</td>
<td>0.0018260971296785306</td>
</tr>
<tr>
<td>7</td>
<td>16</td>
<td>2.663000586760944E-4</td>
<td>8.60767871524158E-4</td>
</tr>
<tr>
<td>8</td>
<td>18</td>
<td>3.31333063379607E-4</td>
<td>2.9924223556135885E-4</td>
</tr>
<tr>
<td>9</td>
<td>20</td>
<td>3.44414788486044E-4</td>
<td>3.7666167066683075E-4</td>
</tr>
<tr>
<td>10</td>
<td>22</td>
<td>6.67216073108228E-5</td>
<td>1.358801147891109E-4</td>
</tr>
<tr>
<td>11</td>
<td>24</td>
<td>4.41927163589595E-5</td>
<td>1.2805374914218737E-4</td>
</tr>
<tr>
<td>12</td>
<td>26</td>
<td>8.515961157662944E-4</td>
<td>2.491946753026775E-4</td>
</tr>
<tr>
<td>13</td>
<td>28</td>
<td>8.77518444748370E-5</td>
<td>2.5355458755216276E-4</td>
</tr>
<tr>
<td>14</td>
<td>30</td>
<td>1.255903877668274E-5</td>
<td>1.1160349233239675E-5</td>
</tr>
</tbody>
</table>

Summary of the performances and comparison between Monte Carlo, Latin hypercube and polynomial chaos for the example problem

<table>
<thead>
<tr>
<th></th>
<th>Sample size needed to reach 1% accuracy for Mean / St Dev</th>
<th>Sample size needed to reach 0.1% accuracy for Mean / St Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monte Carlo</td>
<td>256 / 8192</td>
<td>16384 / -(≈ 800000?)</td>
</tr>
<tr>
<td>Latin hypercube</td>
<td>16 / 128</td>
<td>64 / 2048</td>
</tr>
<tr>
<td>Polynomial Chaos</td>
<td>8 / 12</td>
<td>12 / 20</td>
</tr>
</tbody>
</table>
Polynomial Chaos Expansion

Former example: to reach 1% Error in Mean needs 8 points

Standard Deviation needs 12 points

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean 1% error</th>
<th>St Dev 1% error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monte Carlo</td>
<td>256</td>
<td>8192</td>
</tr>
<tr>
<td>Latin hypercube</td>
<td>16</td>
<td>128</td>
</tr>
<tr>
<td>Polyn Chaos</td>
<td><strong>8</strong></td>
<td><strong>12</strong></td>
</tr>
</tbody>
</table>
Polynomial Chaos in modeFRONTIER
Create a stochastic project: variable distribution

- Several **Distribution** to model Input variables uncertainties are available:
  - Cauchy
  - Logistic
  - Normal
  - Uniform

- Select it in **MORDO options** for each variable and specify the **parameters** (e.g., Standard Deviation for Normal type)
Polynomial Chaos Interface

- Polynomial Chaos is switched on by default
- Chaos order can vary from 1 to 7
- Number of Samples must be at least \( \frac{(n\Var + \order)!}{n\Var!\order!} \)
- Virtual samples are to be avoided
Create a stochastic project: sampling points

- **The number of samples** indicates how many simulations are run for each design during the optimisation, accordingly to input variable distribution.

- Higher is the value, more accurate will be the outputs robustness analysis, but harder will be the computational effort.
Problems & Ideas

• Open Problems:
  - How to deal with bounds and constraints
  - What if the function cannot be expanded in a series (how can we decide that for a black-box function)
  - ....

• Ideas:
  - Use of chaos collocation to train a meta-model for virtual optimization
  - ...

Exercise 1

- Load polChaos.prj
- Check the function
- Run
  - Mean and variance estimation with polynomial chaos (e.g. polynomial degree 7, 30 sample points)
  - Mean and variance estimation with Latin Hypercube (keep the number of points fixed for a fair comparison)
  - Mean and variance estimation with Monte Carlo
- Verify the errors
Create a stochastic project: objectives definition

- The objectives (constraints) can be defined using **Stochastic value** for each output variable. If the latter is o1, these values are:
  - o1:MEAN
  - o1:STDEV
  - o1:MAX
  - o1:MIN
Create a stochastic project: objectives definition

- For each design, a Distribution is obtained for the outputs, collecting the results of the samples:

- The quantities to be used in optimisation are:

  - you can minimise MEAN and constrain STDEV
  - you can minimise MAX as alternative
Robust Design Table

- Robust design table contains stochastic values for each design
- Design Table reports values for each samples (ID number) of the corresponding design (RID number)
Stability control

- The most useful DOE algorithm for stability control is Monte Carlo / Latin Hypercube
- Several different distributions can be defined for input parameters: Exponential, Gamma, Logistic, Lognormal, Normal, Student, Uniform, Weibull
- Monte Carlo perturbations around a point look for robust solutions that are not influenced by small variations of the design variables
Monte Carlo DOE

Frequency Histogram:
The output variable is not influenced by small variation of the design variables

Monte Carlo Perturbation for input variables
Monte Carlo DOE

Monte Carlo Perturbation for input variables

Frequency Histogram:
The output variable is influenced by small variation of the design variables
Exercise 2
Maximize a Mathematical function

Maximize:

\[ F(x, y) = 20 \exp \left( -\frac{\alpha}{2\sigma^2} \right) - 1.6 \left( x^2 + y^2 \right) \]

\[ \sigma = 0.4 \]

\[ \alpha = (x + 2.5)^2 + (y - 2.5)^2 \]

\[ x, y \in [-5.0, 5.0] \]
Maximize a Mathematical function

Global Maximum in:
\[(x, y) \approx (-2.4360, 2.4360)\]
\[f(x, y) \approx 0.5054\]

Robust Maximum in:
\[(x, y) \approx (0, 0)\]
\[f(x, y) \approx 0\]
Stability control

- The most useful DOE algorithm for stability control is Latin Hypercube-Monte Carlo
- Several different distributions are available: Exponential, Gamma, Logistic, Lognormal, Normal, Student, Uniform, Weibull
- Perturbations around a point look for robust solutions that are not influenced by small variations of the design variables
Latin Hypercube - Monte Carlo

- Frequency chart for the input variable x
- Normal distribution with 100 samples using Latin Hypercube
- The table reports the moments of the distribution (mean, variance, skewness, kurtosis and confidence intervals)
- Frequency chart for the output variable for the non-robust maximum
- Values are in the range [-5.33, 0.48]
- The distribution mean is -0.66, the variance is 1.15
Latin Hypercube - Monte Carlo

- Frequency chart for the output variable for the robust maximum
- Values are in the range [-1.5529e-1,-7.8400e-4]
- The distribution mean is -3.23e-2, the variance is -3.23e-2
Monte Carlo DOE

The Frequency Histogram shows if the output variable is influenced by small variation of the design variables.

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global maximum</td>
<td>-0.66</td>
<td>1.15</td>
</tr>
<tr>
<td>Robust maximum</td>
<td>-3.23e-2</td>
<td>-3.23e-2</td>
</tr>
</tbody>
</table>
References