
Methodological reference guide for Uranie v4.7.0

August 8, 2022



THE URANIE TEAM, support-uranie@cea.fr

Contents

I	Glossary	11
II	Basic statistical elements	13
II.1	Random variable modelisation	13
II.1.1	The probability distributions	13
II.2	Statistical treatments and operations	27
II.2.1	Normalising the variable	27
II.2.2	Computing the ranking	27
II.2.3	Computing the elementary statistic	27
II.2.4	The quantile computation	28
II.2.5	Correlation matrix	30
II.3	Combining these aspects: performing PCA	30
II.3.1	Theoretical introduction	30
III	The Sampler module	33
III.1	Introduction	33
III.2	The Stochastic methods	34
III.2.1	Introduction	34
III.2.2	Correlating samples drawn from different marginals	36
III.2.3	The maximin LHS	38
III.2.4	The constrained LHS	40
III.3	QMC method	44

IV	Generating surrogate models	47
IV.1	Introduction	47
IV.1.1	Quality criteria definition	48
IV.1.2	Adapting the fitting strategy	49
IV.2	The linear regression	51
IV.3	Chaos polynomial expansion	52
IV.3.1	Introduction	52
IV.3.2	Nisp in a nutshell	54
IV.4	The artificial neural network	55
IV.4.1	Introduction to the formal neuron	55
IV.4.2	The working principle	56
IV.5	The kriging method	58
IV.5.1	Theoretical introduction	58
IV.5.2	Running a kriging	63
V	Sensitivity analysis	65
V.1	Brief reminder of theoretical aspects	65
V.1.1	Theoretical aspects	65
V.1.2	List of available methods	70
V.2	The finite differences method	70
V.2.1	General presentation of finite difference sensitivity indices	70
V.3	The regression method	71
V.3.1	General presentation of regression's coefficients	71
V.3.2	Getting a confidence-interval estimation	71
V.4	The Morris screening method	72
V.4.1	Principle of the Morris' method	72
V.5	The Sobol method	74
V.5.1	Sobol's sensitivity indices	74
V.6	Fourier-based methods	76
V.6.1	Introducing the method	76
V.6.2	Implementation of methods	77
V.7	The Johnson relative weight	78
V.7.1	Introducing the method	78

VI Dealing with optimisation issues	79
VI.1 Introduction	79
VI.1.1 Single criterion case	79
VI.1.2 The pareto concept in a nutshell	80
VI.2 Multicriteria optimisation	81
VI.2.1 Hitchhiker's guide to genetic algorithms	81
VI.2.2 General discussion on multi and many criteria problem.	83
 VII The Calibration module	 87
VII.1 Brief reminder of theoretical aspects	87
VII.1.1 The distance used to compare observations and model predictions	88
VII.1.2 Discussing assumptions and theoretical background	89
VII.2 Using minimisation techniques	91
VII.3 Analytical linear Bayesian estimation	91
VII.3.1 Prediction values	92
VII.4 The Approximation Bayesian Computation techniques (ABC)	92
VII.4.1 Rejection ABC algorithm	93
VII.5 The Markov-chain approach	94
VII.5.1 Markov-chain principle	94
VII.5.2 The Metropolis-Hasting algorithm	94
 VIII The Uncertainty modeler module	 97
VIII.1 Introduction	97
VIII.2 Tests based on the <i>Empirical Distribution Function</i> ("EDF tests")	97
VIII.3 The <i>Circe</i> method	98
VIII.3.1 Main principle of the CIRCE method	98
 IX References	 101

List of Figures

II.1	Principle of the truncated PDF generation (right-hand side) from the original one (left-hand side). . .	14
II.2	Example of PDF, CDF and inverse CDF for Uniform distribution.	15
II.3	Example of PDF, CDF and inverse CDF for LogUniform distributions.	16
II.4	Example of PDF, CDF and inverse CDF for Triangular distributions.	17
II.5	Example of PDF, CDF and inverse CDF for Logtriangular distributions.	17
II.6	Example of PDF, CDF and inverse CDF for Normal distributions.	18
II.7	Example of PDF, CDF and inverse CDF for LogNormal distributions.	19
II.8	Example of PDF, CDF and inverse CDF for Trapezium distributions.	19
II.9	Example of PDF, CDF and inverse CDF for UniformByParts distributions.	20
II.10	Example of PDF, CDF and inverse CDF for Exponential distributions.	21
II.11	Example of PDF, CDF and inverse CDF for Cauchy distributions.	21
II.12	Example of PDF, CDF and inverse CDF for GumbelMax distributions.	22
II.13	Example of PDF, CDF and inverse CDF for Weibull distributions.	22
II.14	Example of PDF, CDF and inverse CDF for Beta distributions.	23
II.15	Example of PDF, CDF and inverse CDF for GenPareto distributions.	24
II.16	Example of PDF, CDF and inverse CDF for Gamma distributions.	24
II.17	Example of PDF, CDF and inverse CDF for InvGamma distributions.	25
II.18	Example of PDF, CDF and inverse CDF for Student distribution.	26
II.19	Example of PDF, CDF and inverse CDF for generalized normal distributions.	26
II.20	Illustration of the results of 100000 quantile determinations, applied to a reduced centered gaussian distribution, comparing the usual and Wilks methods. The number of points in the reduced centered gaussian distribution is varied, as well as the confidence level.	29
III.1	Schematic view of the input/output relation through a code	33
III.2	Comparison of the two sampling methods SRS (left) and LHS (right) with samples of size 8.	35
III.3	Comparison of deterministic design-of-experiments obtained using either SRS (left) or LHS (right) algorithm, when having two independent random variables (uniform and normal one)	35
III.4	Transformation of a classical LHS (left) to its corresponding maximin LHS (right) when considering a problem with two uniform distributions between 0 and 1.	39

III.5	Matrix of distribution of three uniformly distributed variables on which three linear constraints are applied. The diagonal are the marginal distributions while the off-diagonal are the two-by-two scatter plots.	41
III.6	Comparison of both quasi Monte-Carlo sequences with both LHS and SRS sampling when dealing with two uniform variables.	44
III.7	Comparison of design-of-experiments made with Petras algorithm, using different level values, when dealing with two uniform variables.	45
IV.1	Sketch of the evolution of the bias, the variance and their sum, as a function of the complexity of the model.	49
IV.2	Sketches of under-trained (left), over-trained (middle) and properly trained (right) surrogate models, given that the black points show the training database, while the yellow ones show the testing database	50
IV.3	Evolution of the different kinds of error used to determine when does one start to over-train a model	50
IV.4	Schematical view of the projection of the original value from the code onto the subspace spanned by the column of H (in blue).	51
IV.5	Schematic view of the Nisp methodology	54
IV.6	Schematic description of a formal neuron, as seen in McCulloch and Pitts [21].	55
IV.7	Example of transfer functions: the hyperbolic tangent (left) and the logistical one (right)	56
IV.8	Schematic description of the working flow of an artificial neural network as used in Uranie	57
IV.9	Influence of the variance parameter in the Matern function once fix at 0.5, 1 and 2 (from left to right). The correlation length is set to 1 while the smoothness is set to $3/2$	60
IV.10	Influence of the correlation length parameter in the Matern function once fix at 0.5, 1 and 2 (from left to right). The variance is set to 1 while the smoothness is set to $3/2$	60
IV.11	Influence of the smoothness parameter in the Matern function once fix at 0.5, 1.5 and 2.5 (from left to right). Both the variance and the correlation length are set to 1.	60
IV.12	Evolution of the different covariance functions implemented in Uranie.	61
IV.13	Example of kriging method applied on a simple uni-dimensional function, with a training site of six points, and tested on a basis of about hundred points, with either a gaussian correlation function (left) or a matern $3/2$ one (right).	63
IV.14	Schematic description of the kriging procedure as done within Uranie	64
V.1	Schematic view of two trajectories drawn randomly in the discretised hyper-volume (with $p=6$) for two different values of the elementary variation (the optimal one in black and the smallest one in pink, as detailed on the figure itself).	73
VI.1	Naive example of an imaginary optimisation case relying on two objectives that only depend on a single input variable.	80
VI.2	Description of the children production process in the Uranie implementation of the genetic algorithm	83
VI.3	Comparison of two Pareto sets (left) and fronts (right) from vizir (blue) and MOEAD (ref) when the hollow bar case is studied with very low number of points, <i>i.e.</i> about 20 (simulating higher dimensions).	84

List of Tables

II.1	List of Uranie classes representing the probability laws	14
III.1	Proposed list of parameters value for simulated annealing algorithm, depending on the number of points requested (N) and the number of inputs under consideration (d)	40
IV.1	List of best adapted polynomial-basis to develop the corresponding stochastic law	52

Chapter I

Glossary

Analysis of variance or **ANOVA** (*Analyse de variance*): decomposition of the variance (as a breakdown) to elementary pieces (also known as HDMR, Hoeffding's decomposition, Sobol's decomposition... c.f. Section V.1.1.3).

Cumulative distribution function or **CDF** (*Fonction de répartition*): function of a real-valued random variable X which, once evaluated at x , gives the probability that X will take a value less than or equal to x (c.f. Section II.1.1).

Kriging or **Gaussian process** (*Krigeage ou processus gaussien*): is a family of interpolation methods that uses information about the "spatial" correlation between observations to make predictions with a confidence interval at new locations (c.f. Section IV.5).

Latin hypercube sampling or **LHS** (*Échantillonnage par hypercube latin*): sampling methods that stratifies the probability space by dividing it in equal probabilities (c.f. Section III.2.1).

Leave-one-out or **LOO** (*validation croisée un contre tous*): type of cross-validation for which a surrogate model is re-trained on the learning database removing just one point, in order to obtain an estimation of this new model on this precise point (c.f. Section IV.1.2).

Likelihood (*vraisemblance*): is the hypothetical probability that an event that has already occurred would yield a specific outcome. The concept differs from that of a probability in that a probability refers to the occurrence of future events, while a likelihood refers to past events with known outcomes [2].

Low discrepancy sequence: (*Suite à faible discrétance*): sequence for which the discrepancy is low, meaning the proportion of points in the sequence falling into an arbitrary set B is close to proportional to the measure of B (c.f. Section III.3).

Pareto front (*front de Pareto*): a set of nondominated solutions, being chosen as optimal, if no objective can be improved without sacrificing at least one other objective (c.f. Section VI.1.2).

Pearson coefficient (*Coefficient de Pearson*): it is the linear correlation coefficient (c.f. Section V.1.1.2).

Principal component analysis or **PCA** (*Analyse en conclusion principale*): the process of computing the principal components and using them to perform a change of basis on the data, sometimes using only the first few principal components and ignoring the rest (c.f. Section II.3).

Probability density function or **PDF** (*Densité de probabilité*): function whose value at any given sample (or point) in the sample space (the set of possible values taken by the random variable) can be interpreted as providing a relative likelihood that the value of the random variable would equal that sample (c.f. Section II.1.1).

Quantile (*Quantile*): the quantile x_p , for a probability $p \in [0, 1]$, is the lowest value of a random variable X so that $P\{X \leq x_p\} = p$ (c.f. Section II.2.4).

Screening method (*méthode de criblage*): process that extracts, isolates and identifies a compound or group of components in a sample with the minimum number of steps and the least manipulation of the sample (c.f. Chapter V).

Simple random sampling or **SRS** (*Échantillonnage simple aléatoire*): independent generation of samples following provided PDFs (*c.f.* Section III.2.1).

Sparse grids: numerical techniques to represent, integrate or interpolate high dimensional functions.

Chapter II

Basic statistical elements

This chapter introduces the various probability laws implemented in Uranie and illustrates, for each every one of them, with a few sets of parameters, the resulting shape of three of their characteristic functions. Some of the basic statistical operations are also described in a second part.

II.1 Random variable modelisation

II.1.1 The probability distributions

There are several already-implemented statistical laws in Uranie, that can be called marginal laws as well, used to describe the behaviour of a chosen input variable. They are usually characterised by two functions which are intrinsically connected: the PDF (probability density function) and CDF (cumulative distribution function). One can recap briefly the definition of these two functions for every random variable $X : \Omega \rightarrow \mathbb{R}$:

- PDF: if the random variable X has a density f_X , where f_X is a non-negative Lebesgue-integrable function, then

$$P\{a \leq X \leq b\} = \int_a^b f_X(s)ds$$

- CDF: the function $F_X : \mathbb{R} \rightarrow [0, 1]$, given by

$$F_X(x) = \int_{-\infty}^x f_X(s)ds, x \in \mathbb{R}$$

For some of the distributions discussed later on, the parameters provided to define them are not limiting the range of their PDF and CDF: these distributions are said to be infinite-based ones. It is however possible to set boundaries in order to truncate the span of their possible values. One can indeed define an lower bound L and or an upper bound U so that the resulting distribution range is not infinite anymore but only in $[L, U]$. This truncation step affects both the PDF and CDF: once the boundaries are set, the CDF of these two values are computed to obtain P_L (the probability to be lower than the lower edge) and P_U (the probability to be lower than the upper edge). Two new functions, the truncated PDF $f_X^{[L,U]}$ and the truncated CDF $F_X^{[L,U]}$ are simply defined as

$$f_X^{[L,U]}(x) = \frac{f_X(x)}{P_U - P_L}, F_X^{[L,U]}(x) = \frac{F_X(x) - P_L}{P_U - P_L}.$$

These steps to produce a truncate distribution are represented in Figure II.1 where the original distribution is shown on the left along with the definition of L (the blue shaded part) and U (the green shaded part). The right part of the plot is the resulting truncated PDF.

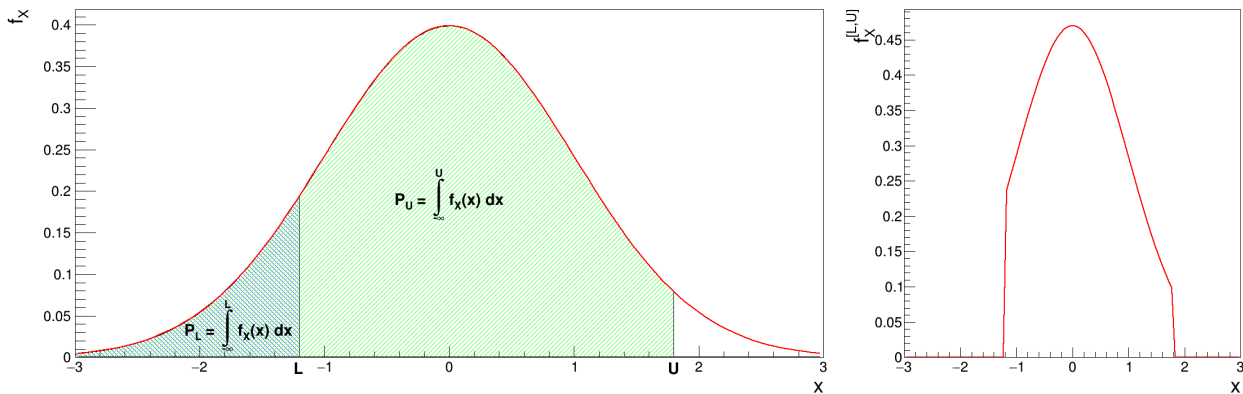


Figure II.1: Principle of the truncated PDF generation (right-hand side) from the original one (left-hand side).

These distributions can be used to model the behaviour of variables, depending on chosen hypothesis, probability density function being used as a reference more often by physicist, whereas statistical experts will generally use the cumulative distribution function [3].

Table II.1 gathers the list of implemented statistical laws, along with the list of parameters used to define them. For every possible law, a figure is displaying the PDF, CDF and inverse CDF for different sets of parameters (the equation of the corresponding PDF is reminded as well on every figure). The inverse CDF is basically the CDF whose x and y -axis are inverted (it is convenient to keep in mind what it looks like, as it will be used to produce design-of-experiments, later-on).

Law	Class Uranie	Parameter 1	Parameter 2	Parameter 3	Parameter 4
Uniform	TUniformDistribution	Min	Max		
Log-Uniform	TLogUniformDistribution	Min	Max		
Triangular	TTriangularDistribution	Min	Max	Mode	
Log-Triangular	TLogTriangularDistribution	Min	Max	Mode	
Normal (Gauss)	TNormalDistribution	Mean (μ)	Sigma (σ)		
Log-Normal	TLogNormalDistribution	Mean (M)	Error factor (E_f)	Min	
Trapezium	TTrapeziumDistribution	Min	Max	Low	Up
UniformByParts	TUniformByPartsDistribution	Min	Max	Median	
Exponential	TExponentialDistribution	Rate (λ)	Min		
Cauchy	TCauchyDistribution	Scale (γ)	Median		
GumbelMax	TGumbelMaxDistribution	Mode (μ)	Scale (β)		
Weibull	TWeibullDistribution	Scale (λ)	Shape (k)	Min	
Beta	TBetaDistribution	alpha (α)	beta (β)	Min	Max
GenPareto	TGenParetoDistribution	Location (μ)	Scale (σ)	Shape (ξ)	
Gamma	TGammaDistribution	Shape (α)	Scale (β)	Location (ξ)	
InvGamma	TInvGammaDistribution	Shape (α)	Scale (β)	Location (ξ)	
Student	TStudentDistribution	DoF (k)			
GeneralizedNormal	TGeneralizedNormalDistribution	Location (μ)	Scale (α)	Shape (β)	

Table II.1: List of Uranie classes representing the probability laws

```
//Uniform law
```

```
TUniformDistribution *pxu = new TUniformDistribution("x1", -1.0 , 1.0); ❶[1]
// Gaussian Law
TNormalDistribution *pxn = new TNormalDistribution("x2", -1.0 , 1.0); ❷[2]
```

- ❶ Allocation of a pointer pxu to a random uniform variable $x1$ in interval $[-1.0, 1.0]$.
- ❷ Allocation of a pointer pxn to a random normal variable $x2$ with mean value $\mu=-1.0$ and standard deviation $\sigma=1.0$.

```
# Uniform law
pxu = DataServer.TUniformDistribution("x1", -1.0 , 1.0) ❶[1]
# Gaussian Law
pxn = DataServer.TNormalDistribution("x2", -1.0 , 1.0) ❷[2]
```

- ❶ Allocation of a pointer pxu to a random uniform variable $x1$ in interval $[-1.0, 1.0]$.
- ❷ Allocation of a pointer pxn to a random normal variable $x2$ with mean value $\mu=-1.0$ and standard deviation $\sigma=1.0$.

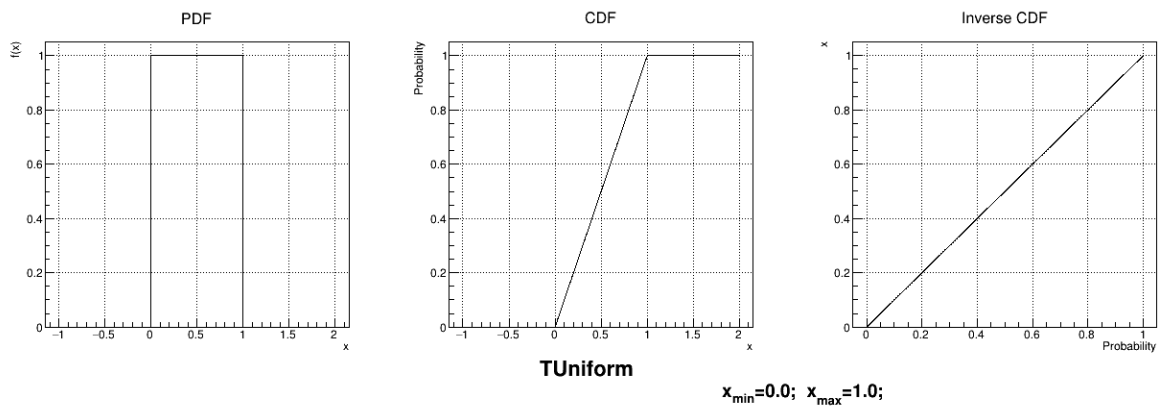
II.1.1.1 Uniform Law

The Uniform law is defined between a minimum and a maximum, as

$$f(x) = \frac{1}{(x_{\max} - x_{\min})} \mathbb{I}_{[x_{\min}, x_{\max}]}(x)$$

The property of the law lies on the fact that all points of the interval $[x_{\min}, x_{\max}]$ have the same probability. The mean value of the uniform law can then be computed as $\mu = \frac{x_{\max} + x_{\min}}{2}$ while its variance can be written as $\sigma^2 = \frac{(x_{\max} - x_{\min})^2}{12}$.

Figure II.2 shows the PDF, CDF and inverse CDF generated for a given set of parameters.



$$f(x) = \frac{1}{(x_{\max} - x_{\min})} \text{ for } x \in [x_{\min}, x_{\max}]$$

2022-08-08 - Uranie v4.7/0

Figure II.2: Example of PDF, CDF and inverse CDF for Uniform distribution.

II.1.1.2 Log Uniform Law

The LogUniform law is well adapted for variations of high amplitudes. If a random variable x follows a LogUniform distribution, the random variable $\ln(x)$ follows a Uniform distribution, so

$$f(x) = \frac{1}{(x \times \ln(x_{\max}/x_{\min}))} \mathbb{I}_{[x_{\min}, x_{\max}]}(x)$$

Figure II.3 shows the PDF, CDF and inverse CDF generated for different sets of parameters.

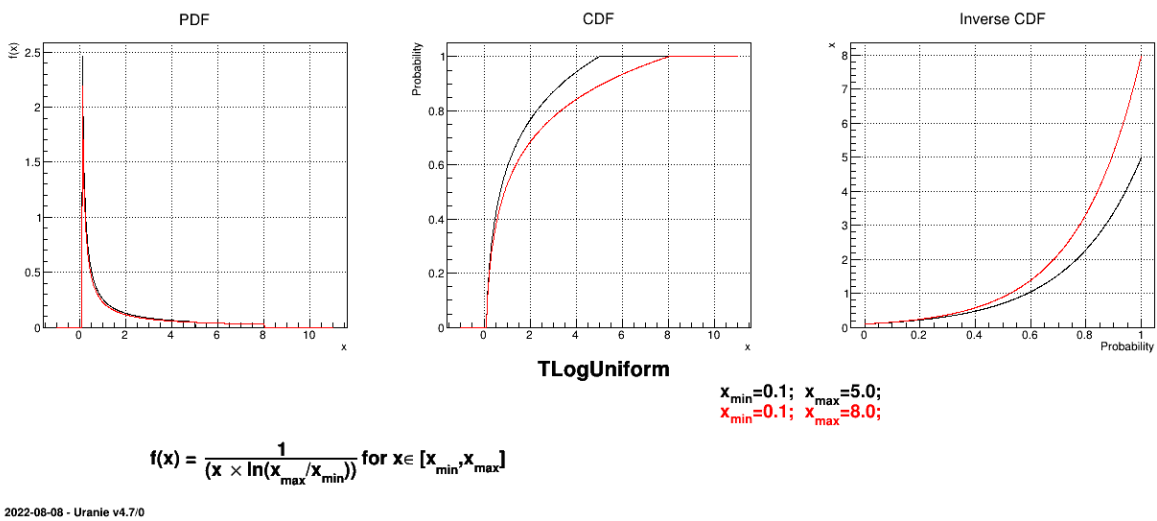


Figure II.3: Example of PDF, CDF and inverse CDF for LogUniform distributions.

II.1.1.3 Triangular law

This law describes a triangle with a base between a minimum and a maximum and a highest density at a certain point x_{mode} , so

$$f(x) = \frac{2 \times (x - x_{\min})}{(x_{\max} - x_{\min}) \times (x_{\text{mode}} - x_{\min})} \mathbb{I}_{[x_{\min}, x_{\text{mode}}]}(x) \quad \text{and} \quad f(x) = \frac{2 \times (x_{\max} - x)}{(x_{\max} - x_{\min}) \times (x_{\max} - x_{\text{mode}})} \mathbb{I}_{[x_{\text{mode}}, x_{\max}]}(x)$$

The mean value of the triangular law can then be computed as $\mu = \frac{x_{\max} + x_{\min} + x_{\text{mode}}}{3}$ while its variance can be written as $\sigma^2 = \frac{(x_{\max}^2 + x_{\min}^2 + x_{\text{mode}}^2 - x_{\max}x_{\min} - x_{\max}x_{\text{mode}} - x_{\text{mode}}x_{\min})}{18}$.

Figure II.4 shows the PDF, CDF and inverse CDF generated for different sets of parameters.

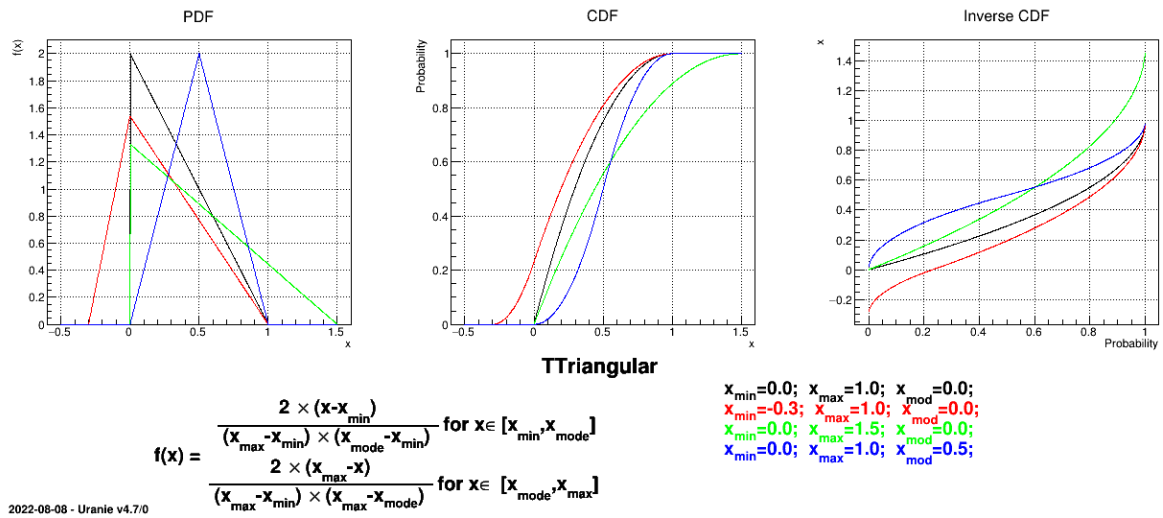


Figure II.4: Example of PDF, CDF and inverse CDF for Triangular distributions.

II.1.1.4 LogTriangular law

If a random variable x follows a LogTriangular distribution, the random variable $\ln(x)$ follows a Triangular distribution, so

$$f(x) = \frac{2 \times \ln(x/x_{\min})}{x \times \ln(x_{\max}/x_{\min}) \times \ln(x_{\text{mode}}/x_{\min})} \mathbb{I}_{[x_{\min}, x_{\text{mode}}]}(x)$$

and

$$f(x) = \frac{2 \times \ln(x_{\max}/x)}{x \times \ln(x_{\max}/x_{\min}) \times \ln(x_{\max}/x_{\text{mode}})} \mathbb{I}_{[x_{\text{mode}}, x_{\max}]}(x)$$

Figure II.5 shows the PDF, CDF and inverse CDF generated for different sets of parameters.

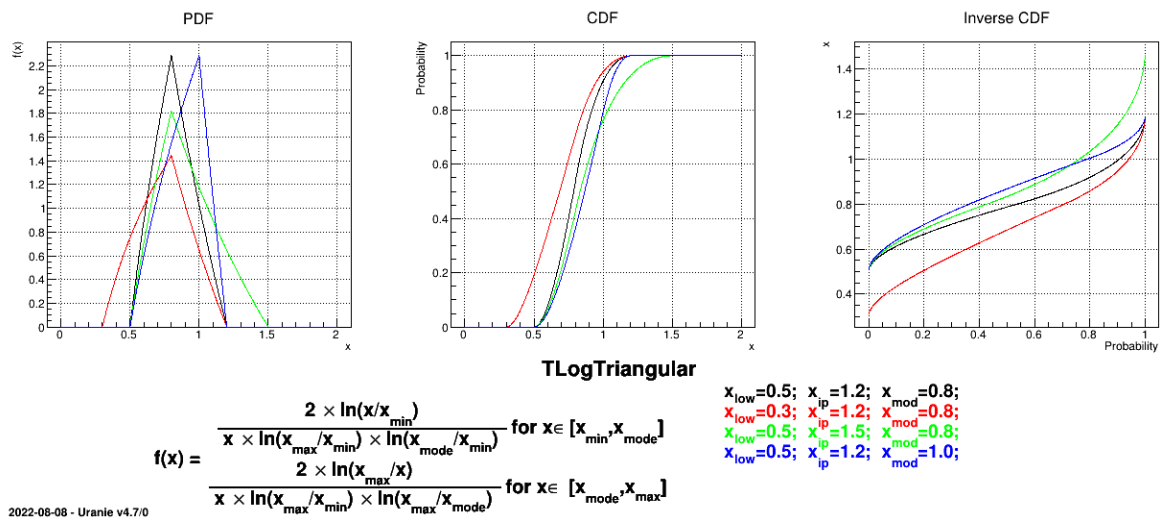


Figure II.5: Example of PDF, CDF and inverse CDF for Logtriangular distributions.

II.1.1.5 Normal law

A normal law is defined with a mean μ and a standard deviation σ , as

$$f(x) = e^{-\frac{(x-\mu)^2}{2\sigma^2}} \times \frac{1}{\sqrt{2\pi}\sigma^2}$$

Figure II.6 shows the PDF, CDF and inverse CDF generated for different sets of parameters.

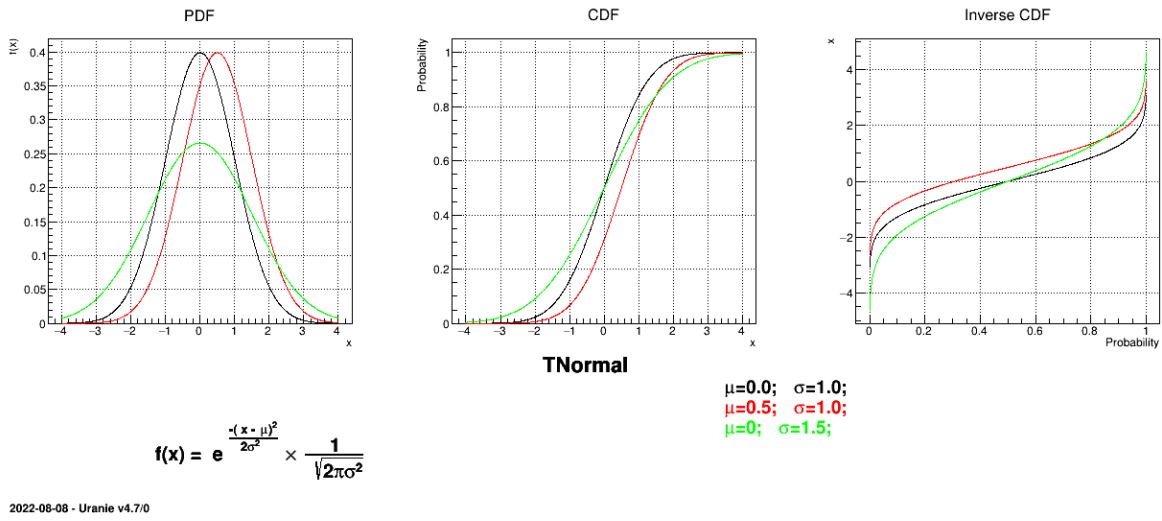


Figure II.6: Example of PDF, CDF and inverse CDF for Normal distributions.

II.1.1.6 LogNormal law

If a random variable x follows a LogNormal distribution, the random variable $\ln(x)$ follows a Normal distribution (whose parameters are μ and σ), so

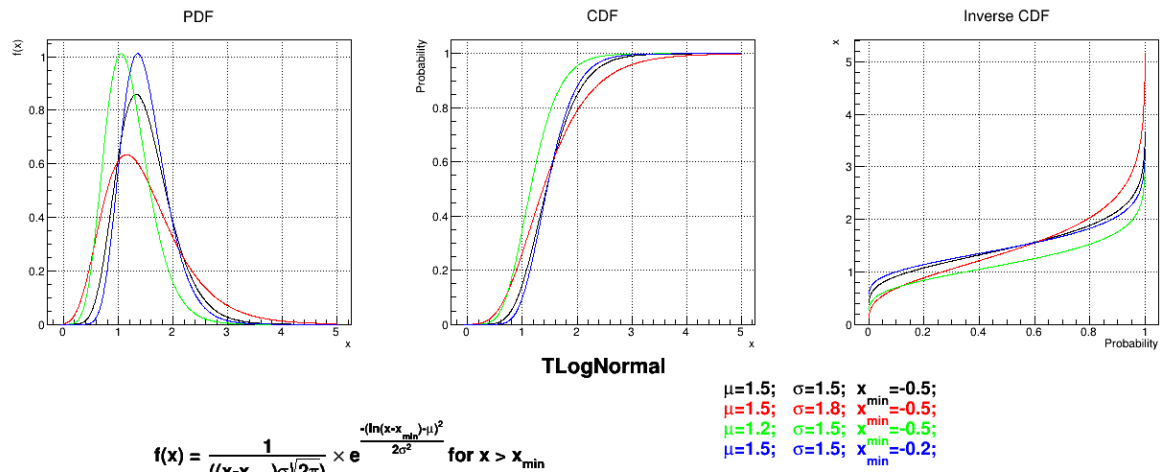
$$f(x) = \frac{1}{(x - x_{\min})\sigma\sqrt{2\pi}} \times e^{-\frac{(\ln(x-x_{\min})-\mu)^2}{2\sigma^2}} \mathbb{I}_{[x_{\min}, +\infty[}(x)$$

In Uranie, it is parametrised by default using M , the mean of the distribution, E_f , the Error factor that represents the ration of the 95% quantile and the median ($E_f = q_{0.95}/q_{0.50}$) and the minimum x_{\min} . One can go from one parametrisation to the other following those simple relations

$$\begin{aligned} M &= e^{\mu+\sigma^2/2} + x_{\min} &\Leftrightarrow & \mu = \ln(M - x_{\min}) - \sigma^2/2 \\ E_F &= e^{1.645 \times \sigma} &\Leftrightarrow & \sigma = \ln(E_f)/1.645. \end{aligned}$$

The variance of the distribution can be estimated as $\text{Var} = (e^{\sigma^2} - 1)e^{2\mu+\sigma^2} = (e^{(\frac{\ln(E_f)}{1.645})^2} - 1) \times (M - x_{\min})^2$

Figure II.7 shows the PDF, CDF and inverse CDF generated for different sets of parameters.



2022-08-08 - Uranie v4.7/0

Figure II.7: Example of PDF, CDF and inverse CDF for LogNormal distributions.

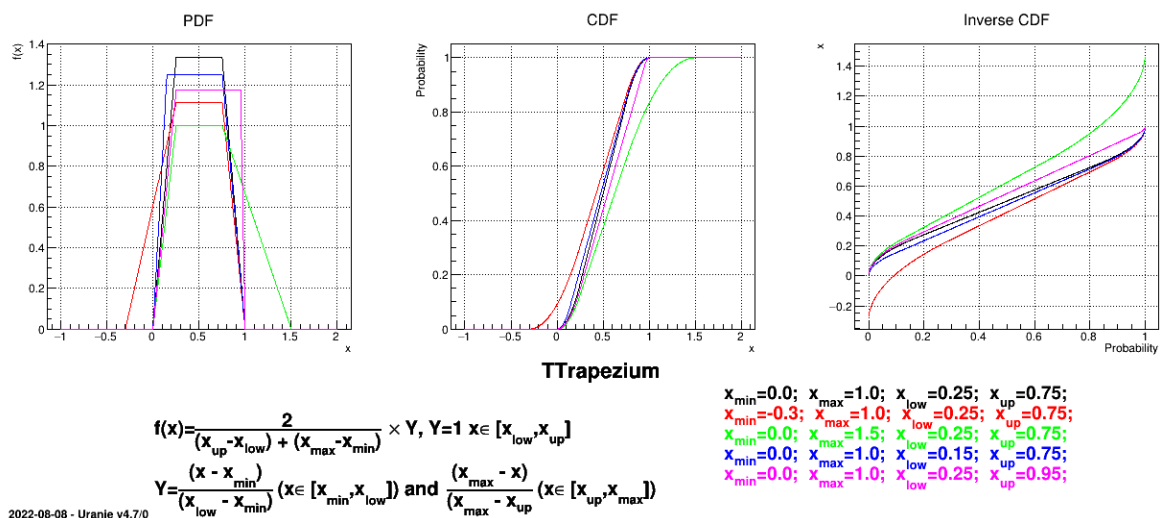
II.1.1.7 Trapezium law

This law describes a trapezium whose large base is defined between a minimum and a maximum and its small base lies between a low and an up value, as

$$f(x) = \frac{2}{(x_{\text{up}} - x_{\text{low}}) + (x_{\text{max}} - x_{\text{min}})} \times Y$$

where $Y = 1$ for $x \in [x_{\text{low}}, x_{\text{up}}]$, $Y = \frac{(x - x_{\text{min}})}{(x_{\text{low}} - x_{\text{min}})}$ for $x \in [x_{\text{min}}, x_{\text{low}}]$ and $Y = \frac{(x_{\text{max}} - x)}{(x_{\text{max}} - x_{\text{up}})}$ for $x \in [x_{\text{up}}, x_{\text{max}}]$

Figure II.8 shows the PDF, CDF and inverse CDF generated for different sets of parameters.



2022-08-08 - Uranie v4.7/0

Figure II.8: Example of PDF, CDF and inverse CDF for Trapezium distributions.

II.1.1.8 UniformByParts law

The UniformByParts law is defined between a minimum and a median and between the median and a maximum, as

$$f(x) = \frac{0.5}{(x_{\text{med}} - x_{\text{min}})} \mathbb{I}_{[x_{\text{min}}, x_{\text{med}}]}(x) \quad \text{and} \quad f(x) = \frac{0.5}{(x_{\text{max}} - x_{\text{med}})} \mathbb{I}_{[x_{\text{med}}, x_{\text{max}}]}(x)$$

Figure II.9 shows the PDF, CDF and inverse CDF generated for different sets of parameters.

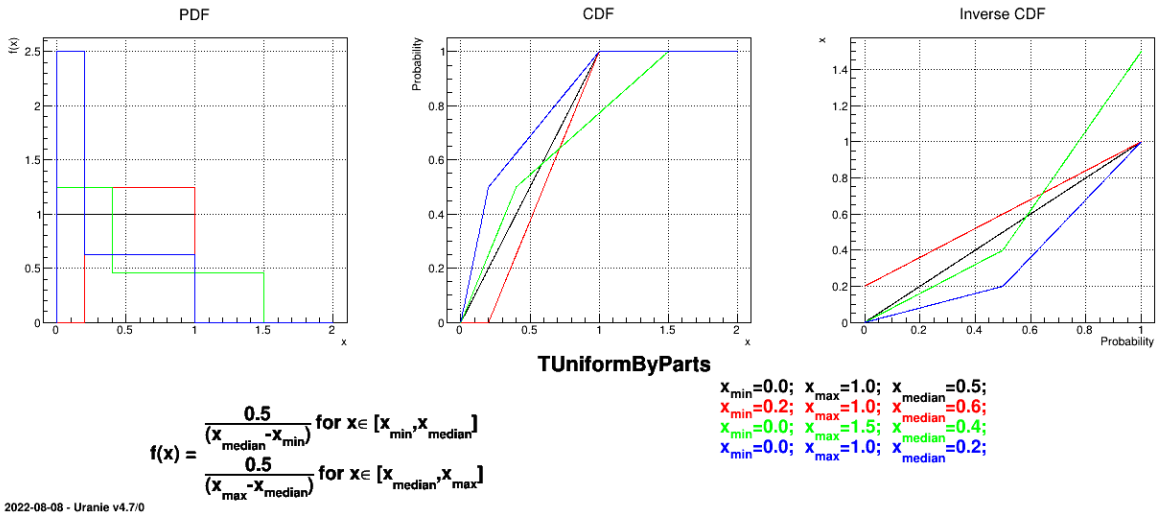


Figure II.9: Example of PDF, CDF and inverse CDF for UniformByParts distributions.

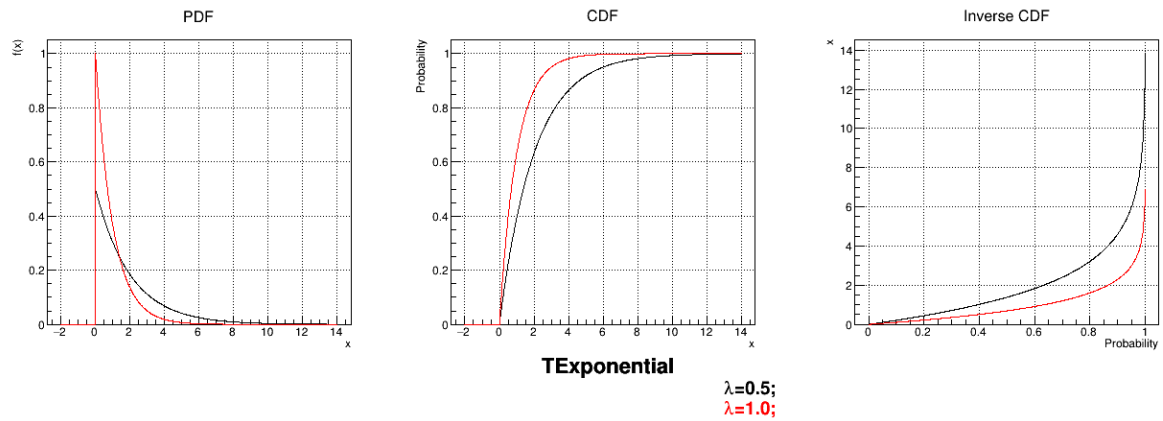
II.1.1.9 Exponential law

This law describes an exponential with a rate parameter λ and a minimum x_{min} , as

$$f(x) = \lambda \times e^{-\lambda \times (x - x_{\text{min}})} \mathbb{I}_{[x_{\text{min}}, +\infty[}(x)$$

The rate parameter λ should be positive. The mean value of the exponential law can then be computed as $\mu = \lambda^{-1} + x_{\text{min}}$ while its variance can be written as $\sigma^2 = \lambda^{-2}$.

Figure II.10 shows the PDF, CDF and inverse CDF generated for different sets of parameters.



$$f(x) = \lambda \times e^{-\lambda \times (x - x_{\min})}, \text{ for } x \geq x_{\min}$$

2022-08-08 - Uranie v4.7/0

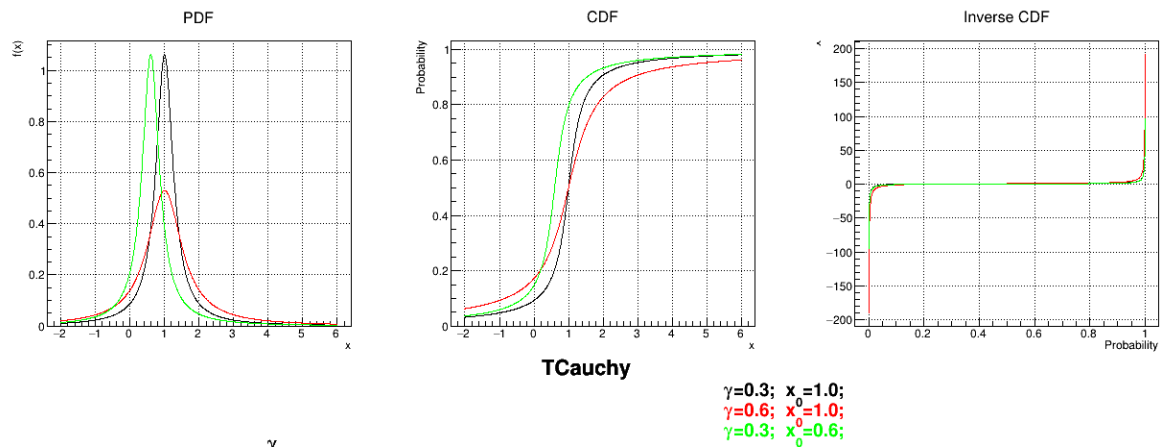
Figure II.10: Example of PDF, CDF and inverse CDF for Exponential distributions.

II.1.1.10 Cauchy law

This law describes a Cauchy-Lorentz distribution with a location parameter x_0 and a scale parameter γ , as

$$f(x) = \frac{\gamma}{\pi \times (\gamma^2 + (x - x_0)^2)}$$

Figure II.11 shows the PDF, CDF and inverse CDF generated for different sets of parameters.



2022-08-08 - Uranie v4.7/0

Figure II.11: Example of PDF, CDF and inverse CDF for Cauchy distributions.

II.1.1.11 GumbelMax law

This law describes a Gumbel max distribution depending on the mode μ and the scale β , as

$$f(x) = z \times \frac{e^{-z}}{\beta}, \text{ where } z = e^{\frac{-(x-\mu)}{\beta}}$$

The mean value of the Gumbel max law can then be computed as $\text{mean} = \mu + \beta\gamma$, where γ is the Euler Mascheroni constant and its variance can be written as $\sigma^2 = \frac{\pi^2}{6}\beta^2$.

Figure II.12 shows the PDF, CDF and inverse CDF generated for different sets of parameters.

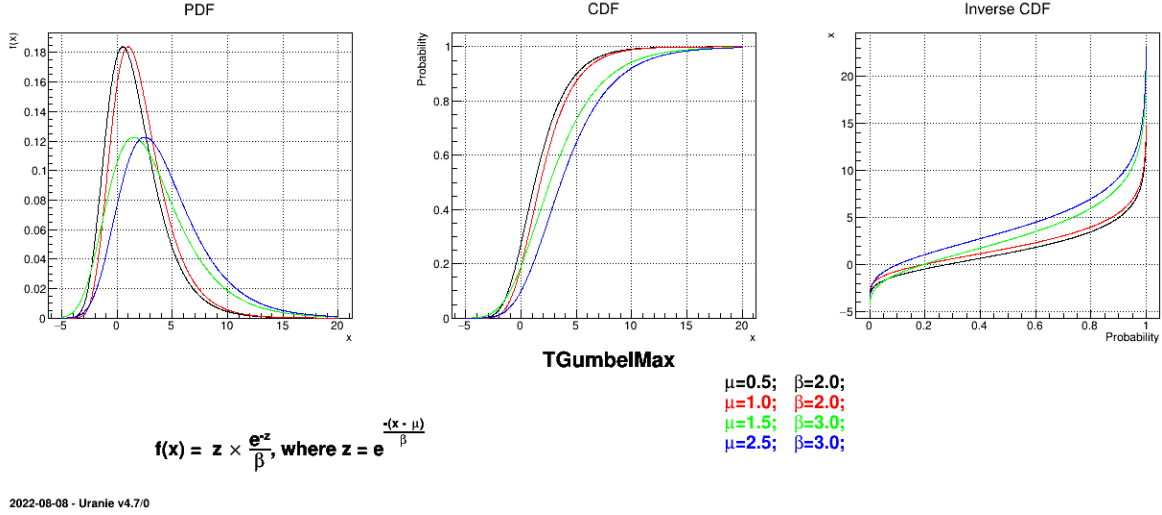


Figure II.12: Example of PDF, CDF and inverse CDF for GumbelMax distributions.

II.1.1.12 Weibull law

This law describes a weibull distribution depending on the location x_{\min} , the scale λ and the shape k , as

$$f(x) = \frac{k}{\lambda} \times \left(\frac{x - x_{\min}}{\lambda} \right)^{k-1} \times e^{-\left(\frac{x - x_{\min}}{\lambda} \right)^k} \mathbb{I}_{[x_{\min}, +\infty[}(x)$$

The mean value of the Weibull law can then be computed as $\mu = \lambda \Gamma(1 + 1/k) + x_{\min}$ while its variance can be written as $\sigma^2 = \lambda [\Gamma(1 + 2/k) - (\Gamma(1 + 1/k))^2]$.

Figure II.13 shows the PDF, CDF and inverse CDF generated for different sets of parameters.

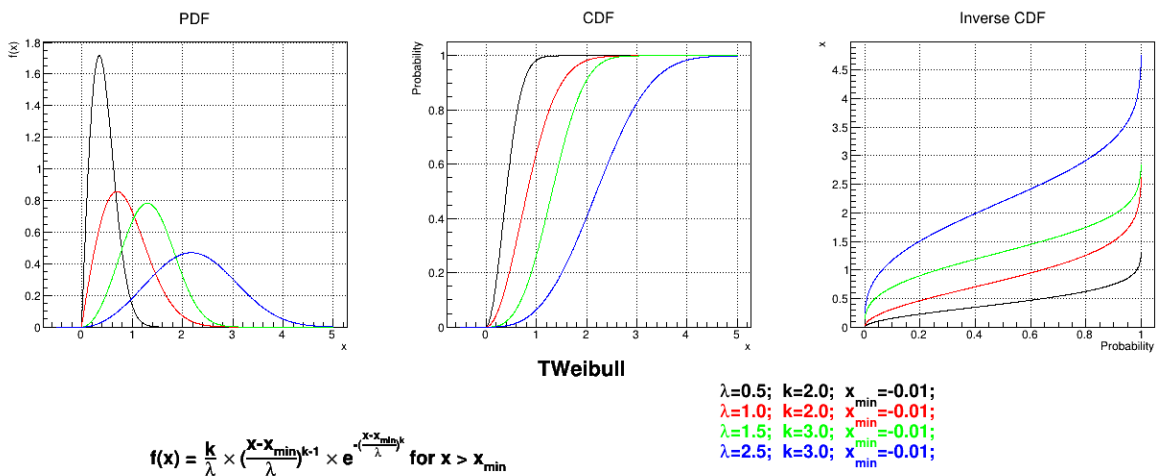


Figure II.13: Example of PDF, CDF and inverse CDF for Weibull distributions.

II.1.1.13 Beta law

Defined between a minimum and a maximum, it depends on two parameters α and β , as

$$f(x) = \frac{Y^{\alpha-1} \times (1-Y)^{\beta-1}}{B(\alpha, \beta)} \mathbb{I}_{[x_{\min}, x_{\max}]}(x)$$

where $Y = \frac{(x - x_{\min})}{(x_{\max} - x_{\min})}$ and $B(\alpha, \beta)$ is the beta function.

Figure II.14 shows the PDF, CDF and inverse CDF generated for different sets of parameters.

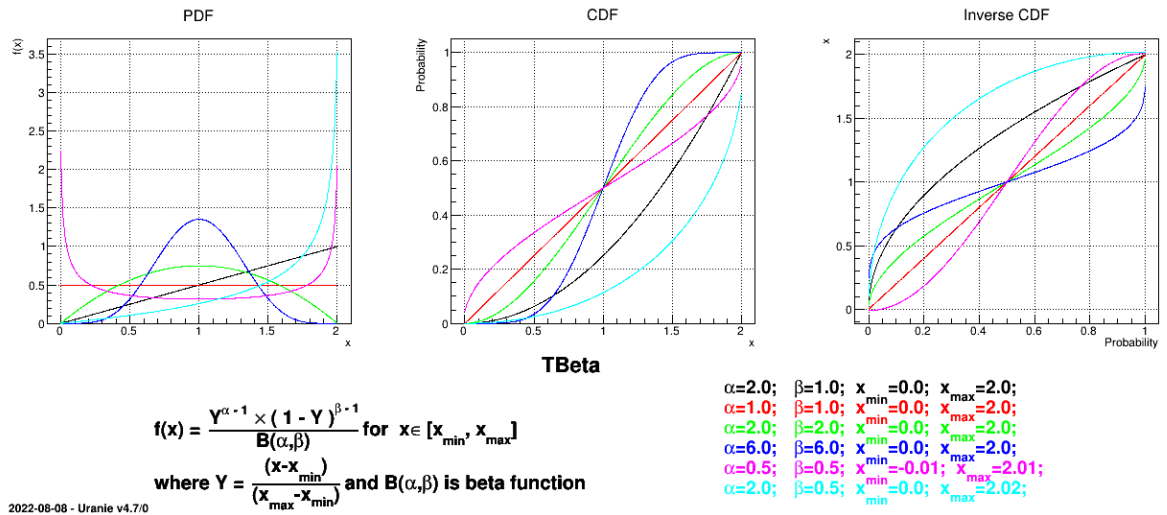


Figure II.14: Example of PDF, CDF and inverse CDF for Beta distributions.

II.1.1.14 GenPareto law

This law describes a generalised Pareto distribution depending on the location μ , the scale σ and a shape ξ , as

$$f(x) = \frac{1}{\sigma} \times \left(1 + \xi \left(\frac{x - \mu}{\sigma} \right) \right)^{-(1/\xi + 1)}$$

In this formula, σ should be greater than 0. The resulting mean for this distribution can be estimated as $\mu + \sigma/(1 - \xi)$ (for $\xi < 1$) while its variance can be computed as $\frac{\sigma^2}{(1 - \xi)^2(1 - 2\xi)}$ (for $\xi < 0.5$).

Figure II.15 shows the PDF, CDF and inverse CDF generated for different sets of parameters.

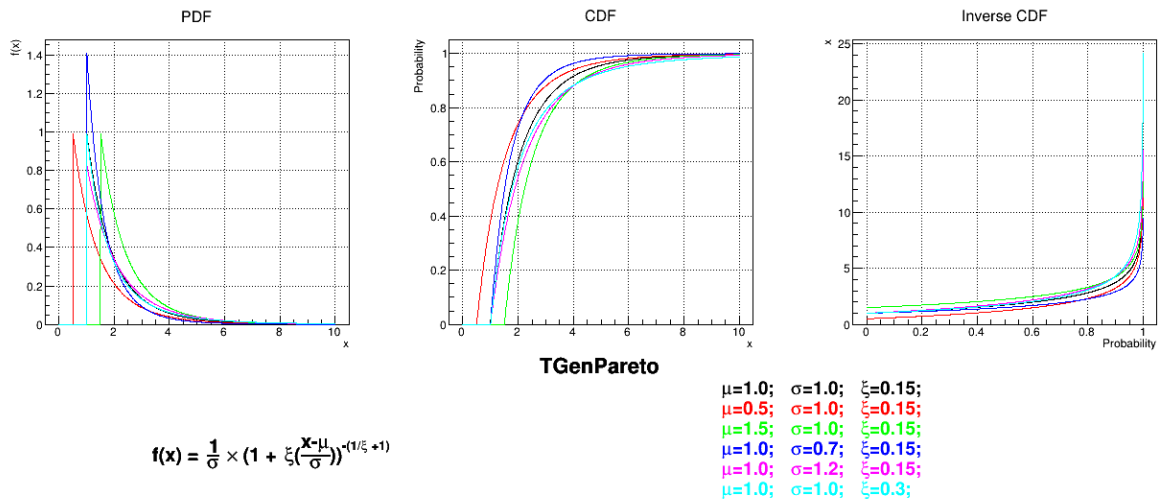


Figure II.15: Example of PDF, CDF and inverse CDF for GenPareto distributions.

II.1.1.15 Gamma law

The Gamma distribution is a two-parameter family of continuous probability distributions. It depends on a shape parameter α and a scale parameter β . The function is usually defined for x greater than 0, but the distribution can be shifted thanks to the third parameter called location (ξ) which should be positive. This parametrisation is more common in Bayesian statistics, where the gamma distribution is used as a conjugate prior distribution for various types of laws:

$$f(x) = \frac{(x - \xi)^{\alpha-1} e^{-(x-\xi)/\beta}}{\Gamma(\alpha)\beta^\alpha} \mathbb{I}_{[\xi, +\infty[}(x)$$

The mean value of the Gamma law can then be computed as $\mu = \alpha\beta + \xi$ while its variance can be written as $\sigma^2 = \alpha\beta^2$.

Figure II.16 shows the PDF, CDF and inverse CDF generated for different sets of parameters.

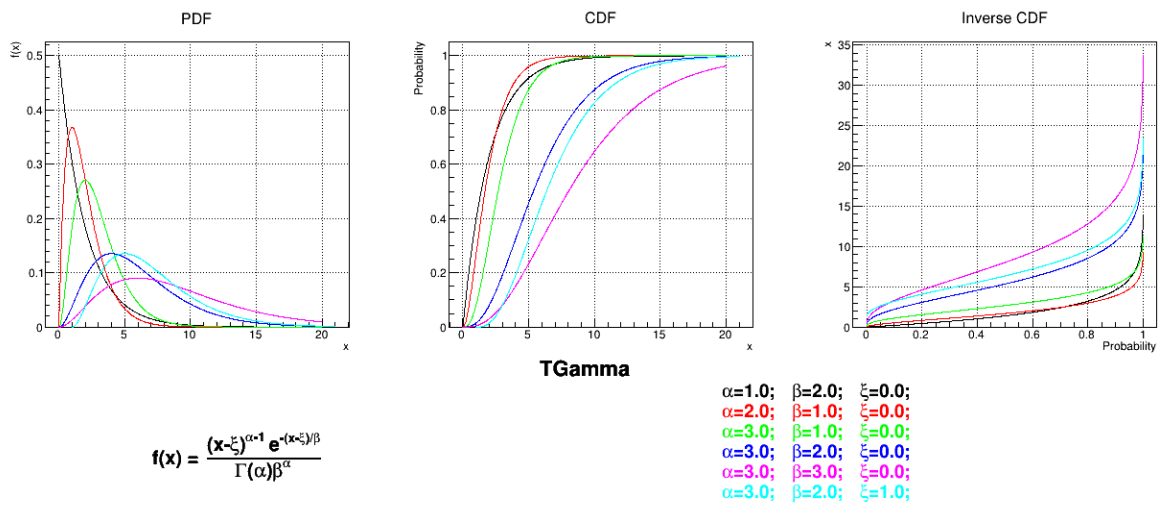


Figure II.16: Example of PDF, CDF and inverse CDF for Gamma distributions.

II.1.1.16 InvGamma law

The inverse-Gamma distribution is a two-parameter family of continuous probability distributions. It depends on a shape parameter α and a scale parameter β . The function is usually defined for x greater than 0, but the distribution can be shifted thanks to the third parameter called location (ξ) which should be positive.

$$f(x) = \frac{\beta^\alpha (x - \xi)^{-\alpha-1} e^{-\beta/(x-\xi)}}{\Gamma(\alpha)} \mathbb{I}_{[\xi, +\infty[}(x)$$

The mean value of the Inverse-Gamma law can then be computed as $\mu = \beta/(\alpha - 1) + \xi$ (for $\alpha > 1$) while its variance can be written as $\sigma^2 = \frac{\beta^2}{(\alpha - 1)^2(\alpha - 2)}$ (for $\alpha > 2$).

Figure II.17 shows the PDF, CDF and inverse CDF generated for different sets of parameters.

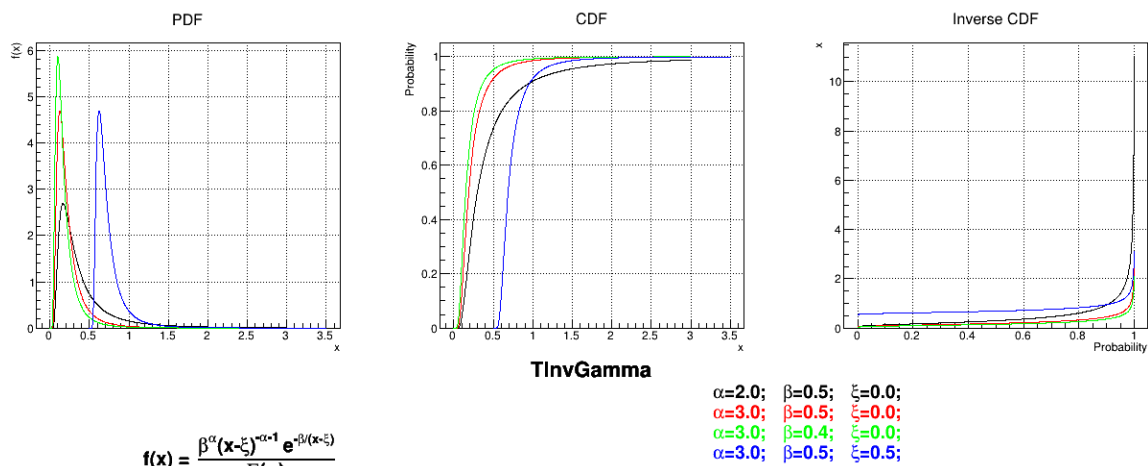


Figure II.17: Example of PDF, CDF and inverse CDF for InvGamma distributions.

II.1.1.17 Student Law

The Student law is simply defined with a single parameter: the degree-of-freedom (**DoF**). The probability density function is then set as

$$f(x) = \frac{1}{\sqrt{k\pi}} \frac{\Gamma(\frac{k+1}{2})}{\Gamma(\frac{k}{2})} \left(1 + \frac{t^2}{k}\right)^{-\frac{k+1}{2}}$$

where Γ is the Euler's gamma function. This distribution is famous for the t-test, a test-hypothesis developed by Fisher to check validity of the null hypothesis when the variance is unknown and the number of degree-of-freedom is limited. Indeed, when the number of degree-of-freedom grows, the shape of the curve looks more and more like the centered-reduced normal distribution. The mean value of the student law is 0 as soon as $k > 1$ (and is not determined otherwise). Its variance can be written as $\sigma^2 = \frac{k}{k-2}$ as soon as $k > 2$, infinity if $1 < k \leq 2$, and is not determined otherwise.

Figure II.18 shows the PDF, CDF and inverse CDF generated for different sets of parameters.

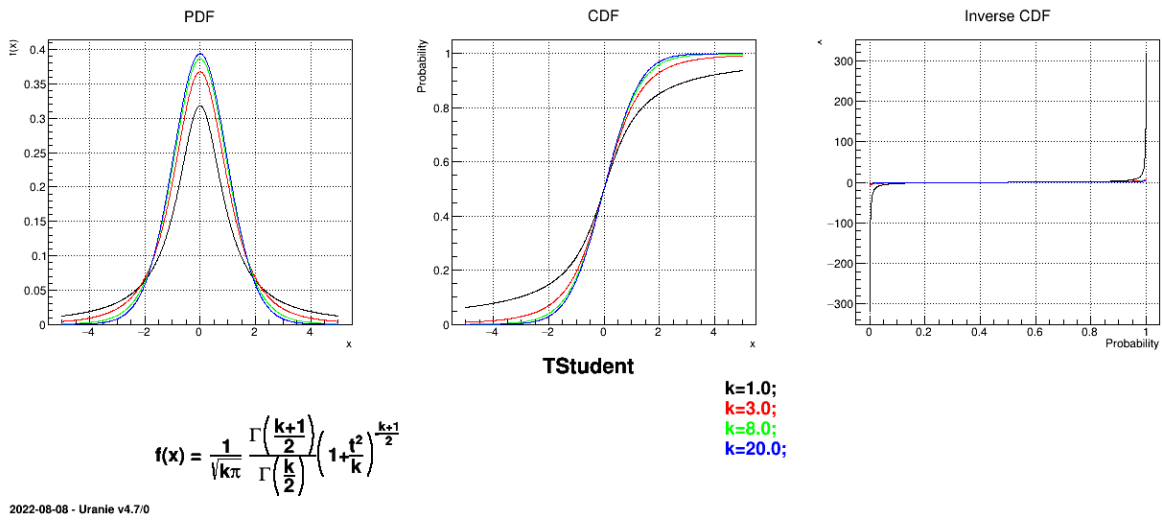


Figure II.18: Example of PDF, CDF and inverse CDF for Student distribution.

II.1.1.18 Generalized normal law

This law describes a generalized normal distribution depending on the location μ , the scale α and the shape β , as

$$f(x) = \frac{\beta}{2\alpha\Gamma(1/\beta)} \times e^{-\left(\frac{x-\mu}{\alpha}\right)^\beta}$$

The mean value of the generalized normal law is μ while its variance can be written as $\frac{\sigma^2 = \alpha^2 \Gamma(3/\beta)}{\Gamma(1/\beta)}$.

Figure II.19 shows the PDF, CDF and inverse CDF generated for different sets of parameters.

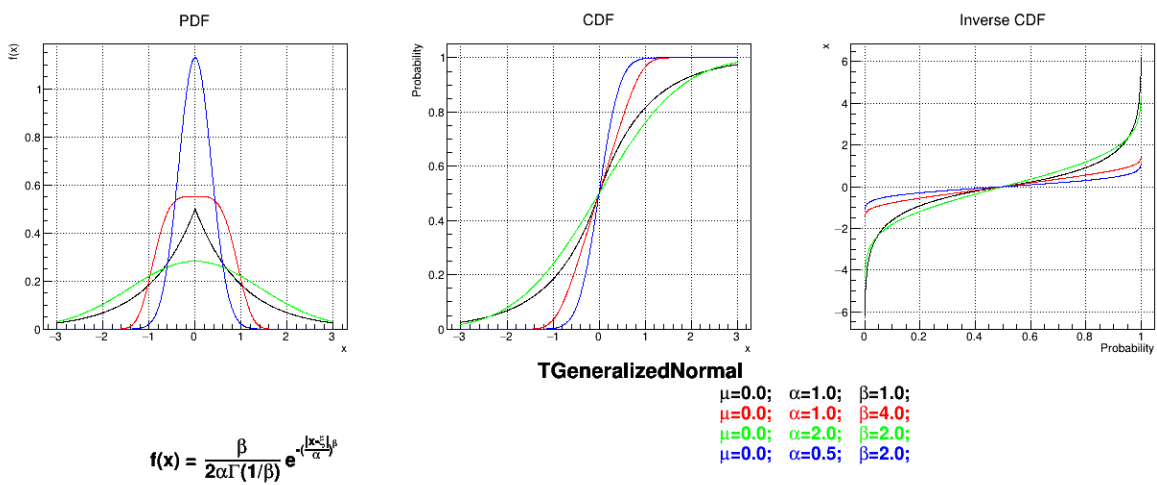


Figure II.19: Example of PDF, CDF and inverse CDF for generalized normal distributions.

II.2 Statistical treatments and operations

There are many different kinds of operations that can be applied on an existing set of data (disregarding their origin, *i.e.* whether they come from experiments, simulations...). They are all listed below and the main ones are described in more details in the following subsections. For the sake of simplicity, the input variable is called x leading to an output variable called y . The dataset contains N points and i is be an iterator that goes from 0 to $N - 1$. In few words, here what's easily calculable with Uranie:

- The normalisation of variable, in Section II.2.1
- The ranking of variable, in Section II.2.2
- The elementary statistic computation, in Section II.2.3
- The quantile estimation, in Section II.2.4
- The correlation matrix determination, in Section II.2.5

II.2.1 Normalising the variable

The normalisation function can be called to create new variables, for every requested normalisation, whose range and dispersion depend on the chosen normalisation method. Up to now, there are four different ways to perform this normalisation:

- centered-reduced: the new variable values are computed as $\tilde{x} = \frac{x - \mu_x}{\sigma_x}$
- centered: the new variable values are computed as $\tilde{x} = x - \mu_x$
- reduced to $[-1, 1]$: the new variable values are computed as $\tilde{x} = 2.0 \times \frac{x - x_{\text{Min}}}{x_{\text{Max}} - x_{\text{Min}}} - 1.0$
- reduced to $[0, 1]$: the new variable values are computed as $\tilde{x} = \frac{x - x_{\text{Min}}}{x_{\text{Max}} - x_{\text{Min}}}$

II.2.2 Computing the ranking

The ranking of variable is used in many methods that are focusing more on monotony than on linearity (this is discussed throughout this documentation when coping with regression, correlation matrix, *c.f.* for instance Section V.1.1.2). The way this is done in Uranie is the following: for every variable considered, a new variable is created, whose name is constructed as the name of the considered variable with the prefix "Rk_". The ranking consists in assigning to each variable entry an integer, that goes from 1 to the number of patterns, following an order relation (in Uranie it is chosen so that 1 is the smallest value and N is the largest one).

II.2.3 Computing the elementary statistic

When considering an existing set of points, it exists a method to determine the four simplest statistical notions: the minimum, maximum, average and standard deviation. The minimum and maximum are trivially estimated by running over all the possible values. The average and standard deviation are estimated on the fly, using the following recursive formulae (where ζ_i represents the value of ζ using all data points up to i for $i = 1, \dots, n_S$):

- average: μ_{x_0} is set to 0 and then

$$\mu_{x_i} = \mu_{x_{i-1}} \times \frac{i}{i+1} + \frac{x_i}{i+1}$$

- standard deviation: σ_{x_0} is set to 0 and then, for i strictly greater than 0,

$$\sigma_{x_i} = \sigma_{x_{i-1}} \times \frac{i-1}{i} + \frac{i+1}{i} \times \frac{(x_i - \mu_{x_i})^2}{i}$$

II.2.4 The quantile computation

There are several ways of estimating the quantiles implemented in Uranie. This part describes the most commonly used and starts with a definition of quantile.

A quantile x_p , as discussed in the following parts, for p a probability going from 0 to 1, is the lowest value of the random variable X leading to $P\{X \leq x_p\} = p$. This definition holds equally if one is dealing with a given probability distribution (leading to a theoretical quantile), or a sample, drawn from a known probability distribution or not (leading to an empirical quantile). In the latter case, the sample is split into two sub-samples: one containing pN points, the other one containing $(1-p)N$ points.

II.2.4.1 Empirical computation

For a given probability p , the corresponding quantile q is given by:

$$q = (1-p)x_k + px_{k+1}$$

where x_k is the k -Th smallest value of the variable set-of-value (whose size is N). The way the index k is computed depends on how conservative one wants to be, but also on the case under consideration. For discontinuous cases, one can choose amongst the following list:

- $k = \lfloor p \times N \rfloor$; if $p \times N = k$, $q = x_k$. $q = x_{k+1}$ otherwise.
- $k = \lfloor p \times N \rfloor$; if $p \times N = k$, $q = 1/2 \times (x_k + x_{k+1})$. $q = x_{k+1}$ otherwise.
- $k = \lfloor p \times N - 0.5 \rfloor$; if $p \times N - 0.5 = k$ and k is even, $q = x_k$. $q = x_{k+1}$ otherwise.

For piece-wise linear interpolations, the estimation of k can be done in Uranie amongst the following cases:

- $k = \lfloor p \times N \rfloor$
- $k = \lfloor p \times N - 0.5 \rfloor$
- $k = \lfloor p \times (N+1) \rfloor$
- $k = \lfloor p \times (N-1) + 1 \rfloor$
- $k = \lfloor p \times (N+1/3) + 1/3 \rfloor$, approximately median unbiased.
- $k = \lfloor p \times (N+1/4) + 3/8 \rfloor$, approximately unbiased if x is normally distributed.

II.2.4.2 Wilks-quantile computation

The Wilks quantile computation is an empirical estimation, based on order statistic which allows to get an estimation on the requested quantile, with a given confidence level β , independently of the nature of the law, and most of the time, requesting less estimations than a classical estimation. Going back to the empirical way discussed in Section II.2.4.1: it consists, for a 95% quantile, in running 100 computations, ordering the obtained values and taking the one at either the 95-Th or 96-Th position (see the discussion on how to choose k in Section II.2.4.1). This can be repeated several times and will result in a distribution of all the obtained quantile values peaking at the theoretical value, with a standard deviation depending on the number of computations made. As it peaks on the theoretical value, 50% of the estimation are larger than the theoretical value while the other 50% are smaller (see Figure II.20 for illustration purpose).

Wilks computation on the other hand request not only a probability value but also a confidence level. The quantile x_p^β represents the x_p quantile given the p probability but this time, the value is provided with a $\beta\%$ confidence level, meaning that $\beta\%$ of the obtain value is larger than the theoretical quantile. This is a way to be conservative and to be able to quantify how conservative one wants to be. To do this, the size of the sample must follow a necessary condition:

$$n > \frac{\ln(1 - \beta)}{\ln p}$$

This is the smallest sample size to get an estimation, and, in most cases, the accuracy reached (for a given sample size) is better than the one achieved with the simpler solution provided above. It is also possible to increase the sample size to get a better description of the quantile estimation.

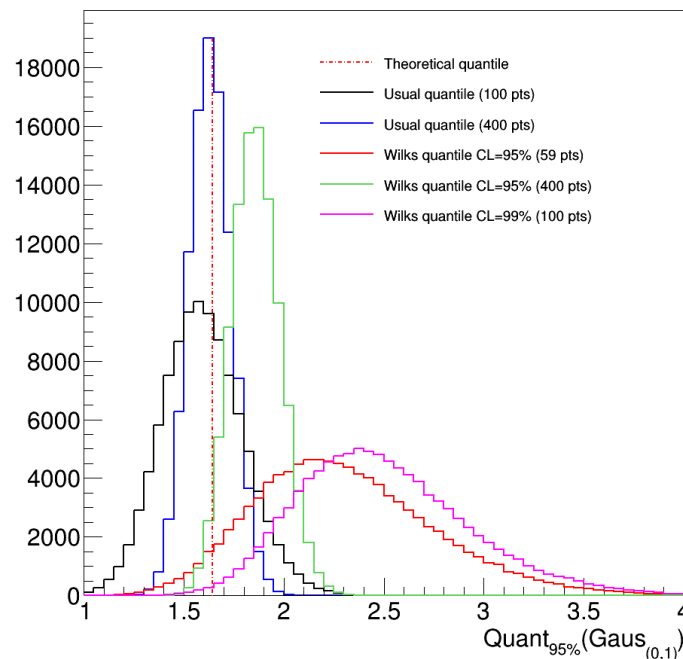


Figure II.20: Illustration of the results of 100000 quantile determinations, applied to a reduced centered gaussian distribution, comparing the usual and Wilks methods. The number of points in the reduced centered gaussian distribution is varied, as well as the confidence level.

Figure II.20 shows a simple case: the estimation of the value of the 95% quantile of a centered-reduced normal distribution. The theoretical value (red dashed line) is compared to the results of 100000 empirical estimation, following

the simple recipe (black and blue curves) or the Wilks method (red, green and magenta curves). Several conclusions can be drawn:

- The simpler quantile estimation average is slightly biased with respect to the theoretical value. This is due to the choice of k , discussed in Section II.2.4.1 which can lead to under or over estimation of the quantile value. The bias becomes smaller with the increasing sample size.
- The standard deviation of the distributions (whatever method is considered) is becoming smaller with the increasing sample size.
- When using the Wilks method, the fraction of event below the theoretical value is becoming smaller with the increasing confidence level.

II.2.5 Correlation matrix

The computation of the correlation matrix can be done either on the values (leading to the Pearson coefficients) or on the ranks (leading to the Spearmann coefficients).

Correlation matrices are computed in a 3 steps procedure detailed below:

1. An overall M matrix is created and filled, every line being a new entry while every column is a variable
2. This matrix is centred and reduced: for every variable under consideration μ_x is subtracted and the results is divided by σ_x .
3. The resulting correlation matrix is obtained from the product ${}^tM \times M$

II.3 Combining these aspects: performing PCA

This part is introducing an example of analysis that combines all the aspects discussed up to now: handling data, perform a statistical treatment and visualise the results. This analysis is called PCA for *Principal Component Analysis* and is often used to

- gather event in a sample that seem to have a common behaviour;
- reduce the dimension of the problem under study.

There is a very large number of articles, even books, discussing the theoretical aspects of principal component analysis (for instance one can have a look at [43]).

II.3.1 Theoretical introduction

II.3.1.1 Purpose

The principle of this kind of analysis is to analyse a provided ensemble, called hereafter \mathcal{D} , whose size is n_S , and which can be written as

$$\mathcal{D} = \{\mathbf{x}^i\}_{i=1, \dots, n_S}$$

where \mathbf{x}^i is the i -Th input vector, written as $\mathbf{x}^i = (x_1^i, \dots, x_{n_X}^i)$ where n_X is the number of quantitative variable. It is basically a set of realisation of n_X random variables whose properties are completely unknown.

The aim is then to summarise (project/reduce) this sample into a smaller dimension space q (with $1 \leq q \leq n_X$) these q factors being chosen in order to maximise the inertia and being orthogonal one to another¹. By doing so, the goal is to be able to reduce the dimension of our problem while losing as few information as possible.

II.3.1.2 Implementation in a nutshell

If one calls \mathbf{X} the original sample whose dimension is (n_X, n_S) , the idea behind PCA is to find the projection matrix \mathbf{P} , whose dimension is (n_X, n_X) , that would re-express the data optimally as a new sample, called hereafter \mathbf{Y} , with the same dimension (n_X, n_S) . The rows of \mathbf{P} are forming a new basis to represent the column of \mathbf{X} and this new basis will later become our principal component directions.

Now recalling the aim of PCA, the way to determine this projection matrix is crucial and should be designed as to

- find out the best linear combinations between variables so that the minimum number of rows (principal components) of \mathbf{P} are considered useful to carry on as much inertia as possible;
- rank the principal component so that, if not satisfy with the new representation, it would be simple to add an extra principal component to improve it.

This can be done by investigating the covariance matrix \mathbf{C}_X of \mathbf{X} that, by definition, describes the linear combination between variables and that could be computed from the centered matrix sample \mathbf{X}_C ² as

$$\mathbf{C}_X = \frac{1}{n_S - 1} \mathbf{X}_C \mathbf{X}_C^T$$

If one consider the resulting covariance matrix \mathbf{C}_Y , the aim is to maximise the signal measured by variance (diagonal entries that represents the variance of the principal components) while minimising the covariance between them. As the lowest covariance value reachable is 0, if the desired covariance matrix \mathbf{C}_Y would append to be diagonal, this would mean our objectives are achieved. From the very definition of the covariance matrix, one could see that

$$\mathbf{C}_Y = \frac{1}{n_S - 1} \mathbf{Y}_C \mathbf{Y}_C^T = \frac{1}{n_S - 1} (\mathbf{P} \mathbf{X}_C) (\mathbf{P} \mathbf{X}_C)^T = \frac{1}{n_S - 1} \mathbf{P} \mathbf{X}_C \mathbf{X}_C^T \mathbf{P}^T = \mathbf{P} \mathbf{C}_X \mathbf{P}^T$$

As \mathbf{C}_X is symmetric, it is orthogonal diagonalisable, and can be written $\mathbf{C}_X = \mathbf{E} \mathbf{S} \mathbf{E}^T$. In this equation, \mathbf{E} is an orthonormal matrix whose columns are the orthonormal eigenvectors of \mathbf{C}_X , and \mathbf{S} is a diagonal matrix which has the eigenvalues of \mathbf{C}_X . Given this, if we choose $\mathbf{P} = \mathbf{E}^T$, this leads to

$$\mathbf{C}_Y = \mathbf{P} \mathbf{C}_X \mathbf{P}^T = \mathbf{E}^T \mathbf{E} \mathbf{S} \mathbf{E}^T \mathbf{E} = \mathbf{S}$$

At this level, there is no unicity of the \mathbf{S} matrix as one can have many permutations of the eigenvalues along the diagonal, as long as one changes \mathbf{E} accordingly.

Finally, an interesting link can be drawn between this protocol and a very classical method of linear algebra, already mentioned in other places of this document, called the Singular Value Decomposition (SVD)³ leading to

$$\mathbf{X}_C^T = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T \text{ where } \mathbf{X}_C^T(n_S, n_X), \mathbf{U}(n_S, n_S), \mathbf{V}(n_X, n_X) \text{ and } \mathbf{\Sigma}(n_S, n_X)$$

EQUATION II.1: General form of a SVD

¹ As a reminder, the dispersion of a quantitative variable is usually represented with its variance (or standard deviation), the inertia criteria is, for multi-dimension problems, the sum of all the variable's variance.

² The centered matrix is defined as $\mathbf{X}_C = \mathbf{X} - \bar{\mathbf{x}}^T \mathbf{1}_{n_S}$ where $\bar{\mathbf{x}}$ is the vector of mean value for every variable and $\mathbf{1}_{n_S}$ is a vector of 1 whose dimension is n_S .

³ SVD is applied to matrix whose number of rows should be greater than its number of columns.

In this context \mathbf{U} and \mathbf{V} are unitary matrices (also known as respectively the *left singular vectors* and *right singular vectors* of \mathbf{X}_C^T) while Σ is a diagonal matrix storing the singular values of \mathbf{X}_C^T in decreasing order. The last step is then to state the linear algebra theorem which says that the non-zero singular values of \mathbf{X}_C^T are the square roots of the nonzero eigenvalues of $\mathbf{X}_C\mathbf{X}_C^T$ and $\mathbf{X}_C^T\mathbf{X}_C$ (the corresponding eigenvectors being the columns of respectively \mathbf{V} and \mathbf{U}).

Gathering all this, one can see that by doing the SVD on the centered original sample matrix, the resulting projection matrix can be identified as $\mathbf{P} = \mathbf{V}^T$ and the resulting covariance matrix will be proportional to Σ^2 . The final interesting property is coming from the SVD itself: as Σ^2 gathers the eigenvalues in decreasing order, it assures the unicity of the transformation and give access to the principal component in a hierarchical way.

II.3.1.3 Limitation of PCA

From what has been discussed previously it can appear very appealing, but there are few drawbacks or at least limitations that can be raised:

- This method is very sensitive to extreme points: correlation coefficient can be perturbed by them.
- In the case of non-linear phenomenon, the very basic concept of PCA collapses. Imagine a simple circle-shaped set of points, there are no correlation between the two variables, so no smaller space can be found using linear combinations.
- Even if the PCA is working smoothly, one has to be able to find an interpretation of the resulting linear combinations that have been defined to create the principle component. Moreover, it might not be possible to move along on more refined analysis, such as sensitivity analysis for instance.

Chapter III

The Sampler module

III.1 Introduction

The Uranie-sampling module is used to produce design-of-experiments knowing the expected behaviour of the input variables for the problem under consideration. The framework of our approach can be illustrated in the following schematic view:

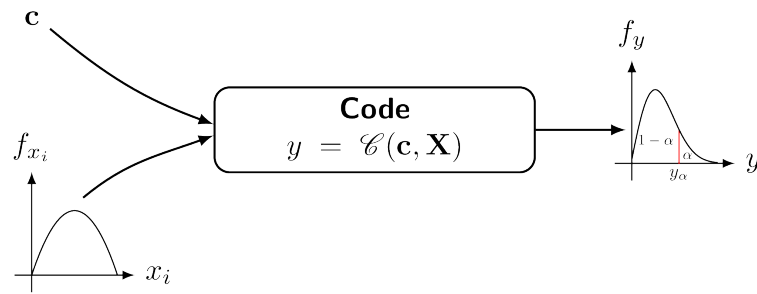


Figure III.1: Schematic view of the input/output relation through a code

- We will denote as \mathcal{C} the studied computational code which, generally, has two types of inputs:
 - The **constant** parameters which are gathered in the vector $\mathbf{c} \in \mathbb{R}^{n_c}$. They represent constants.
 - The **uncertain** parameters which are gathered in the vector $\mathbf{X} \in \mathbb{R}^{n_x}$
It shall be noticed that these parameters are supposed to be **uncertain** either because of a lack of knowledge on their actual value or because of their intrinsic random nature.
- The result of the code \mathcal{C} for a given set of parameters (\mathbf{c}, \mathbf{X}) gives the vector $y \in \mathbb{R}^{n_y} = \mathcal{C}(\mathbf{c}, \mathbf{X})$ which contains all the output variables of the analysis.

Most of the time, the code \mathcal{C} implies solving equations with partial derivatives in more or less complex configurations. The most well-known method to handle this is to generate a sample, as representative as possible of the behaviour of all input variables, in order to fully cover the input parameter phase space. This is the definition of a design-of-experiments which can be generated in many different ways (depending on the analysis purpose, the laws under consideration...). The rest of this section mainly introduces concepts used throughout this documentation; more details can be found either in the user manual or in references.

Different methods exist to obtain a design-of-experiments from uncertain parameters which can be classified into two categories:

1. **stochastic** methods (see Section III.2). These methods consist in using a random number generator to produce new samples. This is also called Monte-Carlo.
2. **deterministic** methods (see Section III.3). Two distinct calls with the same parameters will always give the same point in a design-of-experiments. Some of these methods (those discussed below) are sequences which are sometimes called quasi-Monte Carlo (qMC).

III.2 The Stochastic methods

III.2.1 Introduction

In these methods the knowledge (or mis-knowledge) of the model is encoded in the choice of probability law used to describe the inputs x_i , for $i \in [0, n_X]$. These laws are usually defined by:

- a range that describes the possible values of x_i
- the nature of the law, which has to be taken in the list of pre-defined laws already presented in Section II.1.1

A choice has frequently to be made between two implemented methods of drawing:

SRS (Simple Random Sampling): This method consists in independently generating the samples for each parameter following its own probability density function. The obtained parameter variance is rather high, meaning that the precision of the estimation is poor leading to a need for many repetitions in order to reach a satisfactory precision. An example of this sampling when having two independent random variables (uniform and normal one) is shown in Figure III.3-left. In order to get this drawing, the variable are normalised from 0 to 1 and a random drawing is performed in this range. The obtained value is computed calling the inverse CDF function corresponding to the law under study (that one can see from Figure II.2 until Figure II.18).

LHS (Latin Hypercube Sampling): this method [11] consists in partitioning the interval of each parameter so as to obtain segments of equal probabilities, and afterwards in selecting, for each segment, a value representing this segment. An example of this sampling when having two independent random variables (uniform and normal one) is shown in Figure III.3-right. In order to get this drawing, the variable are normalised from 0 to 1 and this range is split into the requested number of points for the design-of-experiments. Thanks to this, a grid is prepared, assuring equi-probability in every sub-space. Finally, a random drawing is performed in every sub-range. The obtained value is computed calling the inverse CDF function corresponding to the law under study (that one can see from Figure II.2 until Figure II.18).

The first method is fine when the computation time of a simulation is "satisfactory". As a matter of fact, it has the advantage of being easy to implement and to explain; and it produces estimators with good properties not only for the mean value but also for the variance. Naturally, it is necessary to be careful in the sense to be given to the term "satisfactory". If the objective is to obtain quantiles for extreme probability values α (*i.e.* $\alpha = 0.99999$ for instance), even for a very low computation time, the size of the sample would be too large for this method to be used. When a computation time becomes important, the LHS sampling method is preferable to get robust results even with small-size samples (*i.e.* $N_{\text{calc}} = 50$ to 200) [12]. On the other hand, it is rather trivial to double the size of an existing SRS sampling, as no extra caution has to be taken apart from the random seed.

In Figure III.2, we present two samples of size $N_{\text{calc}} = 8$ coming from these two sampling methods for two random variables U_1 according to a gaussian law, and U_2 a uniform law. To make the comparison easier, we have represented on both figures the partition grid of equiprobable segments of the LHS method, keeping in mind that it is not used by the SRS method. These figures clearly show that for LHS method each variable is represented on the whole domain of

variation, which is not the case for the SRS method. This latter gives samples that are concentrated around the mean vector; the extremes of distribution being, by definition, rare.

Concerning the LHS method (right figure), once a point has been chosen in a segment of the first variable U_1 , no other point of this segment will be picked up later, which is hinted by the vertical red bar. It is the same thing for all other variables, and this process is repeated until the N_{calc} points are obtained. This elementary principle will ensure that the domain of variation of each variable is totally covered in a homogeneous way. On the other hand, it is absolutely not possible to remove or add points to a LHS sampling without having to regenerate it completely. A more realistic picture is drawn in Figure III.3 with the same laws, both for SRS on the left and LHS on the right which clearly shows the difference between both methods when considering one-dimensional distribution.

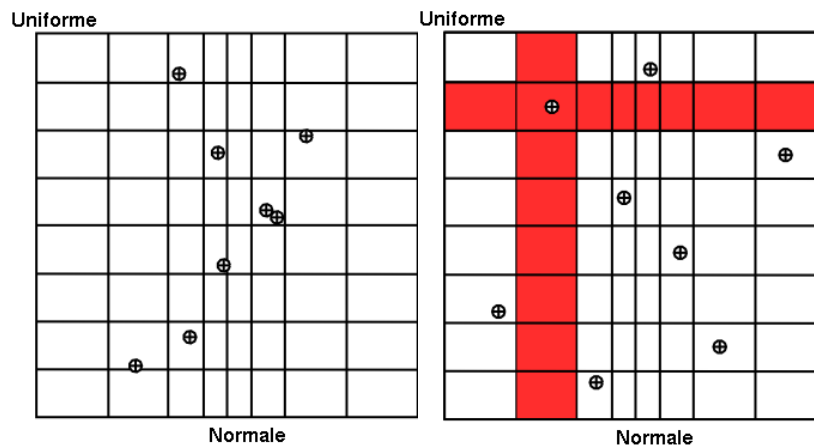
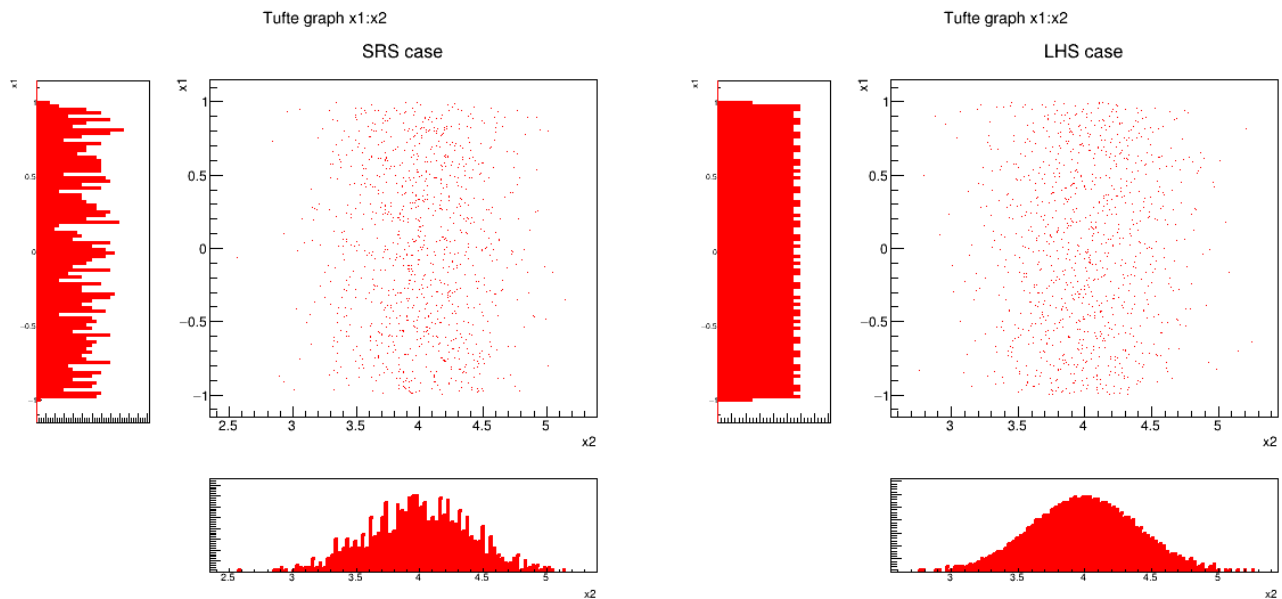


Figure III.2: Comparison of the two sampling methods SRS (left) and LHS (right) with samples of size 8.



2022-08-07 - Uranie v4.7/0

Figure III.3: Comparison of deterministic design-of-experiments obtained using either SRS (left) or LHS (right) algorithm, when having two independent random variables (uniform and normal one)

There are two different sub-categories of LHS design-of-experiments discussed here and whose goal might slightly differs from the main LHS design discussed above:

- the maximin LHS: this category is the result of an optimisation whose purpose is to maximise the minimal distance between any sets of two locations. This is discussed later-on in Section III.2.3.
- the constrained LHS: this category is defined by the fact that someone wants to have a design-of-experiments fulfilling all properties of a Latin Hypercube Design but adding one or more constraints on the input space definition (generally inducing correlation between variables). This is also further discussed in Section III.2.4.

Once the nature of the law is chosen, along with a variation range, for all inputs x_i , the correlation between these variables has to be taken into account. It is doable by defining a correlation coefficient but the way it is treated from one sampler to the other is tricky and is further discussed in the next section.

III.2.2 Correlating samples drawn from different marginals

This section is introducing the way Uranie classes are introducing correlation between random variables when considering either the Pearson or Spearman correlation coefficients. The idea is to better explain the expected behaviour while remaining at this level of correlation description (not going deep into the copula notion).

III.2.2.1 Notation convention

Let start by discussing the definition of a correlation matrix that connect (or not) a variable with the others. For a given problem with n_X variables, the covariance between two variables (denoted $\text{Cov}(X_i, X_j)$) and their linear correlation (denoted $\rho_{X_i X_j}$) can be estimated as

$$\text{Cov}(X_i, X_j) = \mathbb{E}[(x_i - \mu_i)(x_j - \mu_j)] \quad \text{and} \quad \rho_{X_i X_j} = \frac{\text{Cov}(X_i, X_j)}{\sigma_{X_i} \sigma_{X_j}}$$

EQUATION III.1: Covariance and correlation between two variables

In the equation above μ and σ are respectively the mean and standard deviation of the random variable under consideration. The coefficients that should be provided by the user are the correlation one, called the Pearson ones (as they've been estimated using values of the random variables, but this is further discussed at the end of this section and also in Section V.1.1). The idea is to gather all these coefficients in matrix, called hereafter the correlation matrix, that can be written as

$$\mathbf{C}(n_X, n_X) = \begin{pmatrix} 1 & \rho_{X_1 X_2} & \dots & \rho_{X_1 X_{n_X}} \\ \rho_{X_2 X_1} & 1 & \dots & \rho_{X_2 X_{n_X}} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{X_{n_X} X_1} & \rho_{X_{n_X} X_2} & \dots & 1 \end{pmatrix}.$$

Depending on the reference, one can discuss either the correlation matrix (\mathbf{C}) or the covariance matrix (\mathbf{C}_0). Going from one to the other is trivial if one defines \mathbf{D} the diagonal matrix of dimension n_X whose coefficients are the standard deviation of the random variables, then

$$\mathbf{C} = \mathbf{D}^{-1} \mathbf{C}_0 \mathbf{D}^{-1} \Leftrightarrow \mathbf{C}_0 = \mathbf{D} \mathbf{C} \mathbf{D}$$

III.2.2.2 Correlation / de-correlation

Let's assume we have a random drawing $\mathbf{X}(n_S, n_X)$, where every column is the drawing of a given random variable of size n_S . One can then compute the following matrix $\mathbf{T} = n_S^{-1} \mathbf{X}^T \mathbf{X}$ which is the correlation matrix (respectively covariance matrix) of our sample, if the columns have been centered and reduced (respectively only centered). If n_S were to be infinite, we would be able to state that the resulting empirical correlation of the drawn marginal would asymptotically be the identity matrix of dimension n_X , noted $\mathbf{1}_{n_X}$.

The next step is then to correlate the variable so that \mathbf{T} is not the identity anymore but the target correlation matrix \mathbf{C}^* . Knowing that when one multiplies a matrix of random samples by a matrix \mathbf{W} to get $\mathbf{Y} = \mathbf{W}\mathbf{X}$, the resulting variance is estimated as [6]

$$\begin{aligned} \mathbf{C}^* = \text{Var}[\mathbf{Y}] &= \mathbf{W} \text{Var}[\mathbf{X}] \mathbf{W}^T \\ &= \mathbf{W} \mathbf{W}^T, \text{ when } \text{Var}[\mathbf{X}] \rightarrow \mathbf{1}_{n_X} \end{aligned}$$

EQUATION III.2: Simple correlation / de-correlation principle

This leads to the fact that the transformation matrix that provides such a correlation matrix in the end should satisfy the last line of previous equation which is the definition of the Cholesky decomposition of an hermitian positive-definite matrix (\mathbf{W} being a lower triangular matrix)

$$\mathbf{C}^* = \mathbf{W} \mathbf{W}^T$$

EQUATION III.3: General form of a Cholesky decomposition with lower triangular matrix

These steps are the one used to correlate the variables in both in the `TBasicSampling` and `TGaussianSampling` classes. Let's call this method the simple decomposition.

The implementation done in the `TSampling` class, is far more tricky to understand and the aim is not to explain the full concept of the method, called the Iman and Conover method [13]. The rest of this paragraph will just provide insight on what's done specifically to deal with the correlation part which is different from what's been explained up to now. The main difference is coming from the underlying hypothesis, written in the second line of Equation III.2: in a perfect world, for a given random drawing of uncorrelated variables the correlation matrix should satisfy the relation $\mathbf{T} = \mathbf{1}_{n_X}$. This is obviously not the case¹, so one of the proposal to overcome this is to perform a second Cholesky decomposition, on the drawn sample correlation matrix, to get the following decomposition: $\mathbf{T} = \mathbf{K} \mathbf{K}^T$. As \mathbf{K} is lower triangular, it is rather trivial to invert, we can then consider to transform the generated sample using this relation: $\mathbf{Y} = \mathbf{X}(\mathbf{K}^{-1})^T \mathbf{W}^T$. If one consider that these multiplication does not change the fact that columns are centered and reduced, then one can write the following equations

$$\begin{aligned} n_S^{-1} \mathbf{Y}^T \mathbf{Y} &= n_S^{-1} \mathbf{W} \mathbf{K}^{-1} \mathbf{X}^T \mathbf{X} (\mathbf{K}^{-1})^T \mathbf{W}^T \text{ but as } n_S^{-1} \mathbf{X}^T \mathbf{X} = \mathbf{T} = \mathbf{K} \mathbf{K}^T \\ &= \mathbf{W} (\mathbf{K}^{-1} \mathbf{K}) (\mathbf{K}^T (\mathbf{K}^{-1})^T) \mathbf{W}^T \\ &= \mathbf{W} \mathbf{W}^T \\ &= \mathbf{C}^* \end{aligned}$$

Thanks to this procedure (and many more technicalities such as, for instance, working with Spearman coefficient to be able to handle correlation with stratified samples) the resulting correlation matrix is designed to be as close as possible to the target one.

¹Just considering statistical fluctuation should convince people, see for instance the discussion about Fisher's z-transformation and confidence interval on correlation factors developed later on in Section V.3.2

The final part of this discussion is a limitation of both methods: relying on Cholesky decomposition to decompose the target correlation matrix. If one considers the case where \mathbf{C}^* is a singular matrix, then two important points can be raised:

- this case means that one or more variables can be completely defined thanks to the others. The number of properly defined variable can then be estimated by the rank of the correlation matrix. This situation can occur, as in some complicated problem variables can be highly-intricated leading to this kind of situation.
- with this kind of correlation matrix, the Cholesky decomposition is not doable anymore so both methods are meant to stop brutally.

In order to overcome this situation we propose to use a workaround based on the Singular Value Decomposition (SVD) which leads to, knowing that \mathbf{C}^* is real symmetric, $\mathbf{C}^* = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^T$. This writing emphasise the connection between SVD and eigenvalue decomposition (for a more general form and SVD, see for instance Equation II.1). In this context, $\mathbf{U}(n_X, n_X)$ is an unitary matrices while $\mathbf{\Sigma}(n_X, n_X)$ is a diagonal matrix storing the singular values of \mathbf{C}^* in decreasing order. In our case, where the correlation is singular, it means that one or more of the singular values are very close or equal to 0. By rewriting our decomposition as below

$$\mathbf{C}^* = \mathbf{U}\mathbf{\Sigma}^{1/2}\mathbf{\Sigma}^{1/2}\mathbf{U}^T = \mathbf{U}\mathbf{\Sigma}^{1/2}(\mathbf{U}\mathbf{\Sigma}^{1/2})^T$$

one can redefine the matrix $\mathbf{W} = \mathbf{U}\mathbf{\Sigma}^{1/2}$ and get the usual formula discussed above (see Equation III.3). This decomposition can then be used instead of the Cholesky decomposition in both method (as it is either for the simple form or along with another Cholesky to decompose the correlation matrix of the drawn sample, in our modified Iman and Conover algorithm).

The usage of an SVD instead of a Cholesky decomposition for the target correlation matrix relies on the underlying hypothesis that the *left singular vectors* (\mathbf{U}) can be used instead of the *right singular vectors* (\mathbf{V}) in the general SVD formula shown for instance in Equation II.1. This holds even for the singular case, as the only differences seen between both singular vector basis arise for the singular values close to zero. Since in this method we are always using the singular vector matrix weighted by the square roots of the singular values, these differences are vanishing by construction.

III.2.3 The maximin LHS

III.2.3.1 Introduction

Considering the definition of a LHS sampling, introduced in Section III.2.1, it is clear that permutating a coordinate of two different points, will create a new sampling. If one looks at the x-coordinate (corresponding to a normal distribution) in Figure III.2, one could put the point in the second equi-probable range, in the sixth one, and move the point which was in the sixth equi-probable range into the second one, without changing the y-coordinate. The results of this permutation is a new sampling with the interesting property of remaining a LHS sampling. A follow-up question can then be: what is the difference between these two samplings, and would there be any reason to try many permutations ?

This is a very brief introduction to a dedicated field of research: the optimisation of a design-of-experiments with respect to the goals of the ongoing analysis. In Uranie, a new kind of LHS sampling has been recently introduced, called maximin LHS, whose purpose is to maximise the minimal distance between two points. The distance under consideration is the **mindist** criterion: let $D = [\mathbf{x}_1, \dots, \mathbf{x}_N] \subset [0, 1]^d$ be a design-of-experiments with N points. The mindist criterion is written as:

$$\min_{i,j} \|\mathbf{x}_i - \mathbf{x}_j\|_2 \quad (\text{III.1})$$

where $\|\cdot\|_2$ is the euclidian norm. The designs which maximise the mindist criterion are referred to as maximin LHS, but generally speaking, a design with a large value of the mindist criterion is referred to as maximin LHS as well.

It has been observed that the best designs in terms of maximising (III.1) can be constructed by minimising its L^p regularisation instead. It is written as

$$\phi_p := \left[\sum_{i < j} \|\mathbf{x}_i - \mathbf{x}_j\|_2^{-p} \right]^{\frac{1}{p}}$$

Figure III.4 shows, on the left, an example of a LHS when considering a problem with two uniform distributions between 0 and 1 but also, on the right, its transformation through the maximin optimisation. The mindist criterion is displayed on top for comparison purpose.

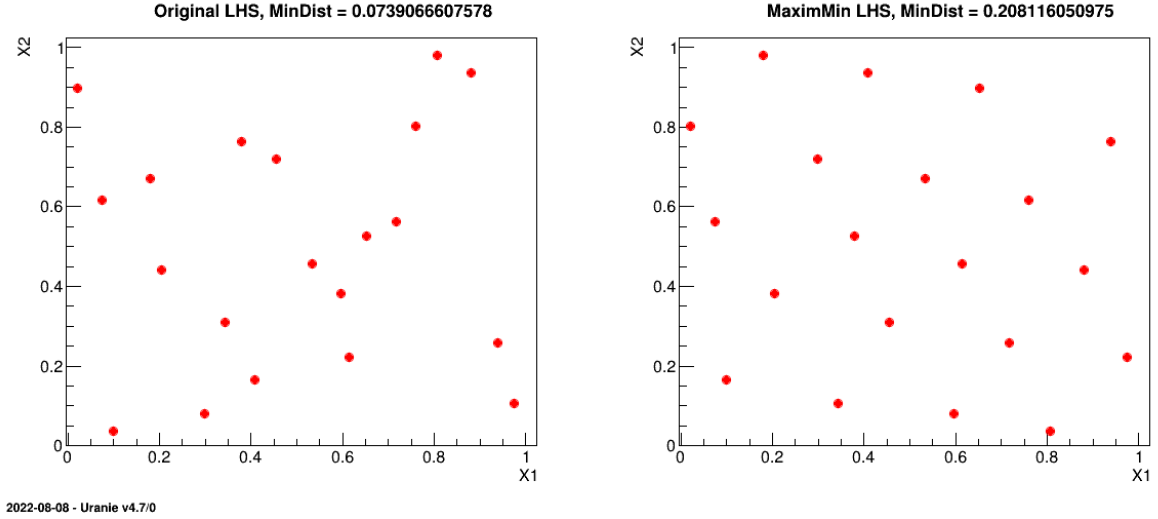


Figure III.4: Transformation of a classical LHS (left) to its corresponding maximin LHS (right) when considering a problem with two uniform distributions between 0 and 1.

From a theoretical perspective, using a maximin LHS to build a Gaussian process (GP) emulator can reduce the predictive variance when the distribution of the GP is exactly known. However, it is not often the case in real applications where both the variance and the range parameters of the GP are actually estimated from a set of learning simulations run over the maximin LHS. Unfortunately, the locations of maximin LHS are far from each other, which is not a good feature to estimate these parameters with precision. That is why maximin LHS should be used with care. Relevant discussions dealing with this issue can be found in [16].

III.2.3.2 The simulated annealing method

The Simulated Annealing (SA) algorithm is a probabilistic metaheuristic which can solve a global optimisation problem. It is here applied to the construction of maximin Latin Hypercube Designs (maximin LHS). The SA algorithm consists in exploring the space of LHS through elementary random perturbations of both rows and columns in order to converge to maximin ones. We have implemented in Uranie the algorithm of Morris and Mitchell [14, 15], which is driven by the following parameters

- T_0 is the initial temperature
- the decreasing of the temperature is controlled by c
- the number of iterations in the outer loop I
- the number of iterations in the inner loop I_{inner}

It is important to keep in mind that the performances of the simulated annealing method can strongly depend on c and thus changing parametrisation can lead to disappointing results. Below, in Table III.1, we provide some parametrisation examples working well with respect both to the number of the input variables d and the size of the requested design N .

d	$N = 10 \times d$		$N = 20 \times d$		$N = 30 \times d$	
2,3,4	c	0.99	c	0.99	c	0.99
	T_0	0.1	T_0	0.1	T_0	0.1
	I	300	I	300	I	300
	I_{inner}	300	I_{inner}	300	I_{inner}	300
5,6,7	c	0.99	c	0.99	c	0.99
	T_0	0.001	T_0	0.001	T_0	0.001
	I	300	I	300	I	300
	I_{inner}	300	I_{inner}	300	I_{inner}	300
8,9,10	c	0.99	c	0.99	c	0.99
	T_0	0.0001	T_0	0.0001	T_0	0.0001
	I	300	I	300	I	300
	I_{inner}	300-1000	I_{inner}	300-1000	I_{inner}	300-1000

Table III.1: Proposed list of parameters value for simulated annealing algorithm, depending on the number of points requested (N) and the number of inputs under consideration (d)

III.2.4 The constrained LHS

III.2.4.1 Introduction

Considering the definition of a LHS sampling, introduced in Section III.2.1 and also discussed in Section III.2.3.1, it is clear that permutating a coordinate of two different points, will create a new sampling. The idea here is to use this already discussed property to create fulfill requested constraints on the design-of-experiments to be produced. Practically, the constraint will have one main limitation: it should only imply two variables. This limitation is set, so far, for simplicity purpose, as the constraint matrix might blow up and the solutions in term of permutation will also become very complicated (see the heuristic description below in Section III.2.4.2 to get a glimpse at the possible complexity).

Before this algorithm, the solution to be sure to fulfill a constraint was to generate a large sample and apply the constraint as a cut, meaning that no control on the final number of locations in the design-of-experiments and on the marginal distribution shape was possible. From a theoretical perspective, using a constrained LHS is allowing both to have the correct expected marginal distributions and to have precisely the requested number of locations to be submitted to a code or function. This is shown in Figure III.5, where in a simple case with only three variables uniformly distributed, it is possible to apply three linear constraints: two of them are applied on the (x_0, x_1) plane while the last one is applied on the (x_1, x_2) one.

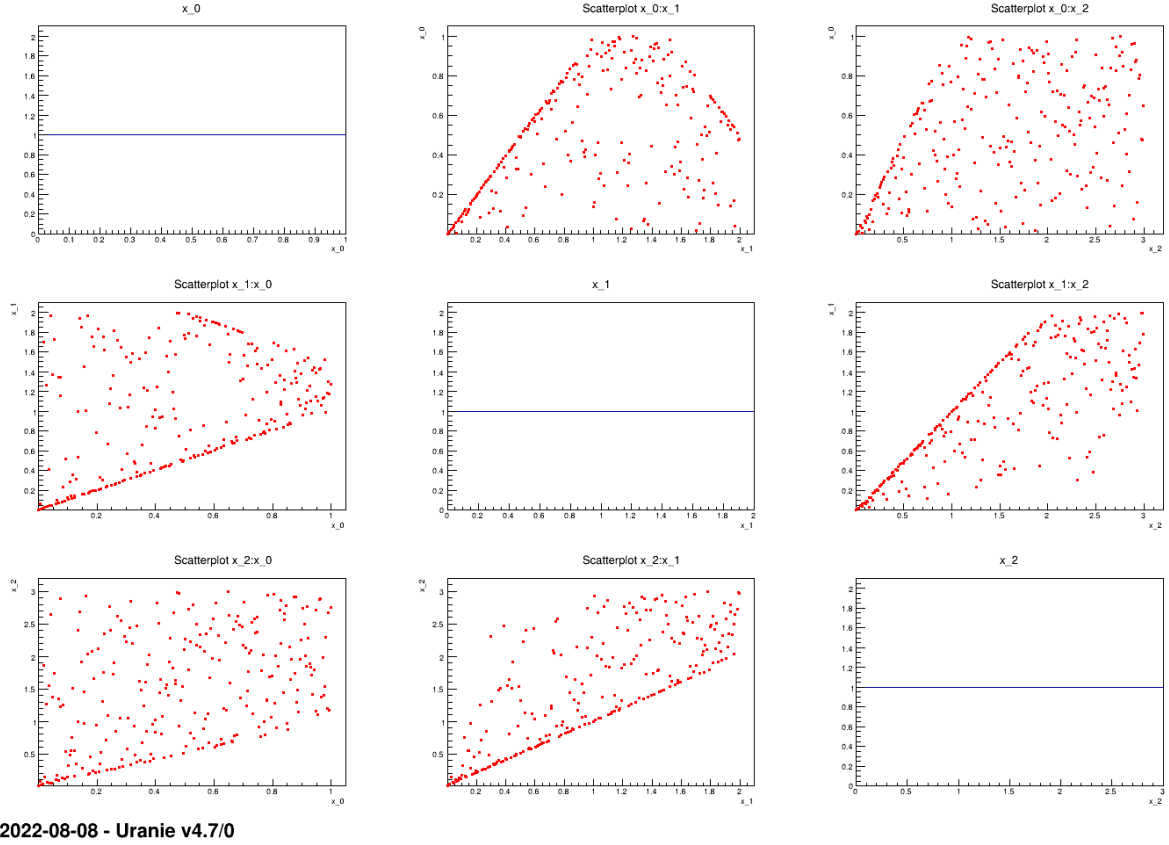


Figure III.5: Matrix of distribution of three uniformly distributed variables on which three linear constraints are applied. The diagonal are the marginal distributions while the off-diagonal are the two-by-two scatter plots.

III.2.4.2 The heuristic

The idea behind this empirical heuristic is to rely on permutations and to decide on the best permutation to be done thanks to the content of the constraint matrix and also the distributions of solutions along the row and columns. This will be discussed further in the rest of this section but first a focus is done on the definition of a constraint. If one considers a simple constraint, its implementation can be decompose in the following steps:

1. define it simply with an equation, for instance one wants to reject all locations for which $x_0 < x_1$;
2. define a constraint function that can compute the margin of success, for instance in our simple case $c(x_0, x_1) = x_0 - x_1$;
3. define a characteristic constraint function that only states whether the constraint is fulfilled, for instance in our simple case $1_c(x_0, x_1) = \begin{cases} 0 & \text{if } x_0 < x_1 \\ 1 & \text{if } x_0 > x_1 \end{cases}$

If formally the constraint function can provide more information when reading it (in terms of margin), one needs to know the way to apply a selection on these results which is not the simplest aspect to provide which explains why in the rest of this section the focus will be put on the characteristic constraint function. In order to illustrate our method, one can start from a provided LHS design-of-experiments, called hereafter $\mathcal{L} = \{\mathbf{x}^i, i = 1, \dots, n_S\}$ where \mathbf{x}^i is the i -Th location

which can be written as $\mathbf{x}^i = (x_1^i, \dots, x_{n_X}^i)$: it is a sample of size n_S in the n_X input space. From there, the constraint matrix is indeed defined as done below:

$$\mathbf{C}(n_S, n_S) = \begin{pmatrix} 1_c(x_{row}^1, x_{col}^1) & 1_c(x_{row}^1, x_{col}^2) & \dots & 1_c(x_{row}^1, x_{col}^{n_S}) \\ 1_c(x_{row}^2, x_{col}^1) & 1_c(x_{row}^2, x_{col}^2) & \dots & 1_c(x_{row}^2, x_{col}^{n_S}) \\ \vdots & \vdots & \ddots & \vdots \\ 1_c(x_{row}^{n_S}, x_{col}^1) & 1_c(x_{row}^{n_S}, x_{col}^2) & \dots & 1_c(x_{row}^{n_S}, x_{col}^{n_S}) \end{pmatrix}.$$

The constraint matrix is, as visible from the definition above, a (n_S, n_S) matrix which only contains 0 and 1 depending on whether the current configuration of the (x_{row}, x_{col}) plane fulfill the constraint. This shows why restraining the constraint to only two-variables function is a reasonable approach: for a given configuration the number of permutations will blow up if one will consider larger dimension constraint. Giving this object, one can introduces more useful objects:

- the constraint row solution vector, that sums up for every row the number of solutions, *i.e.* the number of columns that allow to fulfill the constraint $\{c_{row}^i = \sum_{j=1}^{n_S} \mathbf{C}^{ij}, \forall i \in [1, n_S]\}$;
- the constraint column solution vector, that sums up for every column the number of solutions, *i.e.* the number of row that allow to fulfill the constraint $\{c_{col}^i = \sum_{j=1}^{n_S} \mathbf{C}^{ji}, \forall i \in [1, n_S]\}$;
- the constraint diagonal vector *i.e.* the diagonal of the constraint matrix $\{c_{diag}^i(n_S) = \mathbf{C}^{ii}, \forall i \in [1, n_S]\}$

Bearing in mind that the objective is to have a constraint fulfilled, the heuristic will use the following criterion as a stopping signal: $c_{stop} = \sum_{i=1}^{n_S} c_{diag}^i = n_S$. This heuristic starts from a provided LHS design-of-experiments for which one can consider three cases:

- A** it cannot fulfill the constraint, meaning that there is no sets of permutation to have $c_{stop} = n_S$;
- B** it can fulfill the constraint with only one set of permutation leading to the only solution \mathcal{L}^{constr} ;
- C** it can fulfill the constraint with many different sets of permutation leading to a very large number of configurations and so to a very large number of constrained design-of-experiments.

Practically, the heuristic is organised in a step-by-step approach in which the variable used as first argument in the constraint definition will be used as row indicator (it will be called x_{row}) and it is considered fixed. This implies that the permutation will be done by reorganising the second variable, called x_{col} . The heuristic is then described below:

1. The constraint matrix is estimated along with all the objects discussed above $(\mathbf{C}, c_{row}, c_{col}, c_{diag})$.
 - If c_{row} contains one 0, it means that the design-of-experiments cannot fulfill the constraint. If this design-of-experiments, has been provided, the method stops, on the other hand if it was generated on-the-fly, then another attempt is done (up to 5 times).
 - A bipartite graph method is called to check that there can be at least one solution for every row. If not, if the design-of-experiments has been provided, the method stops, on the other hand if it was generated on-the-fly, then another attempt is done (up to 5 times).

This step allows to sort out the design-of-experiments that will fall into the category A (defined above) from those that might fall either in the B or C ones.

2. If $c_{stop} < n_S$, then all the rows for which the diagonal elements is 0 are kept aside and sorted out by increasing number of solutions over all the columns (their value of c_{row}), which defines $\Omega_{row} = \{i \in [1, n_S], c_{diag}^i = 0\}$. The row considered, whose index will be written k for kept hereafter, is the one with the lowest number of solutions (the most urgent one in a way), so it can be written $k = \min_{c_{row}} \Omega_{row}$

3. For x_{row}^k the chosen value, all the columns that provide a solution are kept aside and sorted out by increasing order² of solutions over all rows (their value of c_{col}), which defines $\Omega_{col}^k = \{i \in [1, n_S], C^{ki} = 1\}$. This sorting provides information on the marging, as the highest values of solutions over the rows means that this value of x_{col} is compatible with many other x_{row} instances. A loop is performed over all these solutions, so that $\forall t \in \Omega_{col}^k$:
 - By definition we know that $C^{kk} = 0$ and $C^{kt} = 1$, so if $C^{tk} = 1$ then the permutation will only increase the stopping criterion (c_{stop}), as the actual column k , will become the new column t after permutation. In this case, one can move to the permutation step. This can be written, by calling s the actual index of the column (for selected), $s = \min_{c_{col}} \{t \in \Omega_{col}^k, C^{tk} = 1\}$. From there, one moves to the permutation step.
 - If none of the solution t under investigation can satisfy the requirement $C^{tk} = 1$, then $s = \min_{c_{col}} \{t \in \Omega_{col}^k, t \notin \Omega_{col,perm}^k\}$. In this definition, the ensemble $\Omega_{col,perm}^k$ is the ensemble of column index which has already been used previously (as this is an interative heuristic) in a permutation for the row k under investigation. This precaution has been introduced in order to prevent from having a loop in the permutation process. From there, one moves to the permutation step.
 - If $\{t \in \Omega_{col}^k, t \notin \Omega_{col,perm}^k\} = \emptyset$, meaning that all possible solutions have been tested, then the selected column index is the result of a random drawing in the nsemble of solutions, meaning that $s = \text{rand}(\Omega_{col}^k)$. From there, one moves to the permutation step.
4. Now that both k and s are known, the permutation will be done, it consists in:
 - changing content of the design-of-experiments, meaning doing $x_{col}^k \leftrightarrow x_{col}^s$;
 - changing the columns in the constraint matrix, meaning doing $C^{ik} \leftrightarrow C^{is}, \forall i \in [1, n_S]$;
 - changing the content of the constraint column solution vector, meaning doing $c_{col}^k \leftrightarrow c_{col}^s$;
 - fill once more the constraint diagonal vector (c_{diag}) and compute once more the stopping criterion (c_{stop}) to check whether the permutation process has to be continued. If so, then one moves to the second step.

This presentation has been simplified as, here, there is only one constraint applied to the design-of-experiments. Indeed, when there are more than one constraint, let's call $(\mathbf{C}, c_{row}, c_{col}, c_{diag})$ and $(\mathbf{D}, d_{row}, d_{col}, d_{diag})$ the objects associated to two constraints defined in both planes (x_{row}^c, x_{col}^c) and (x_{row}^d, x_{col}^d) , few cautions and extra steps have to be followed as long as both stopping criteria, c_{stop} or d_{stop} are different from n_S depending on the variables in the constraint plane definition:

- if planes (x_{row}^c, x_{col}^c) and (x_{row}^d, x_{col}^d) have no common variable, or if $x_{row}^c = x_{row}^d$ but $x_{col}^c \neq x_{col}^d$ then the constraints can be considered orthogonal;
- if x_{col}^c is the same variable as x_{row}^d then any permutations for the constraint c have to be propagated to the objects of the constraint d , meaning that on top of the list in step 4, the extra steps consist in:
 - changing the rows in the constraint matrix for the constraint d , meaning doing $\mathbf{D}^{ki} \leftrightarrow \mathbf{D}^{si}, \forall i \in [1, n_S]$;
 - changing the content of the constraint row solution vector for constraint d , meaning doing $d_{row}^k \leftrightarrow d_{row}^s$;
 - fill also the constraint diagonal vector (d_{diag}) and compute once more the stopping criterion (d_{stop}) to check whether the permutation process has to be continued.
- if $x_{col}^c = x_{col}^d$, then any permutations from on the constraint can undo a previous one defined from the other one, meaning that one can create a loop that will (thanks to our heuristic definition) will end up into the random permutation area. To prevent this, so far, one can not create a second constraint using the same variable as second argument.

²the decreasing order has also been tested, but it has show a lower discrepancy in the resulting design-of-experiments.

III.3 QMC method

The deterministic samplings can produce design-of-experiments with well defined properties, that can be very useful in specific cases such as:

- to cover at best the space of the input variables
- to explore the extreme cases
- to study combined or non-linearity effect

There are two kinds of quasi Monte-Carlo sampling methods implemented in Uranie: the regular ones and the sparse grid ones. On the first hand, the former can be generated using two different sequences:

1. Sequences of **Sobol** [17]
2. Sequences of **Halton** [19]

Figure III.6 shows a comparison of the design-of-experiments obtained with both sequences, along with the ones produced with a basic stochastic sampling, following the LHS and SRS "recipes", all when dealing with two uniform variables. The coverage is clearly more regular in the case of quasi Monte-Carlo sequences which is the origin of their name: low-discrepancy sequences. There are plenty definitions for the notion of discrepancy (see litterature for them) but they all quantify how close the sequence is to a perfect equidistribution of points.

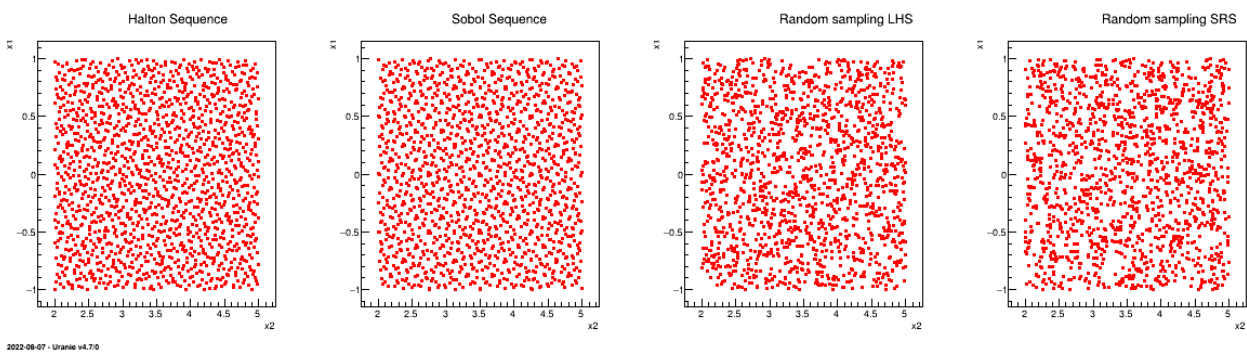


Figure III.6: Comparison of both quasi Monte-Carlo sequences with both LHS and SRS sampling when dealing with two uniform variables.

On the other hand, the sparse grid sampling can be very useful for integration purposes and can be used in some of the meta-modelling definition, see, for instance, in Section IV.3.2.2. In Uranie we can use the Petras algorithm [20] to produce these sparse grids, shown for different levels in Figure III.7, that can be compared to regular algorithms ones in Figure III.6 (in both cases, the problem is described with two uniform variables).

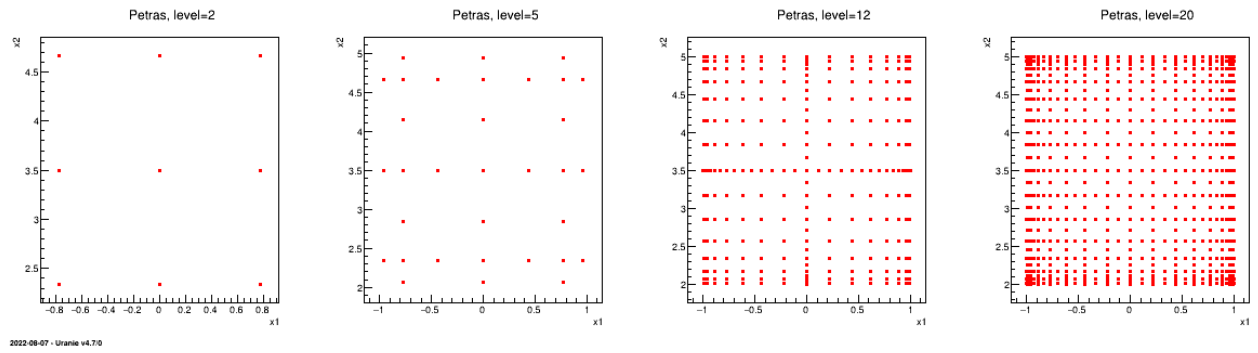


Figure III.7: Comparison of design-of-experiments made with Petras algorithm, using different level values, when dealing with two uniform variables.

Chapter IV

Generating surrogate models

IV.1 Introduction

This part discusses the generation of *surrogate models* which aim to provide a simpler, and hence faster, model in order to emulate the specified output of a more complex model (and generally time and memory consuming) as a function of its inputs and parameters. The input dataset can either be an existing set of elements (provided by someone else, resulting from simulations or experiments) or it can be a design-of-experiments generated on purpose, for the sake of the ongoing study. This ensemble (of size n_S) can be written as

$$\mathcal{L} = \{(\mathbf{x}^i, y^i), i = 1, \dots, n_S\}$$

where \mathbf{x}^i is the i-Th input vector which can be written as $\mathbf{x}^i = (x_1^i \dots x_{n_x}^i)$ and the output $y^i = y(\mathbf{x}^i)$.

There are several predefined surrogate-models proposed in the Uranie platform:

- The linear regression, discussed in Section IV.2
- The chaos polynomial expansion, discussed in Section IV.3
- The artificial neural networks, discussed in Section IV.4
- The Kriging method, or gaussian process, discussed in Section IV.5

It is recommended to follow the law of parsimony (also called Ockham's razor) meaning that the simplest model should be tested first, unless one has insight that it is not well suited for the problem under consideration. There have been many analysis performed to try to provide better guideline than the Ockham's razor on what model to choose, knowing a bit about the physical problem under study. Among these, one can look at this reference [7] that proposes the following recommendations:

- Polynomial models: well-established and easy to use, they are best suited for applications with random error and appropriate for applications with <10 factors.
- Neural networks: good for highly nonlinear or very large problems ($\sim 10\,000$ parameters), they might be best suited for deterministic applications but they imply high computational expense (often .10,000 training data points); best for repeated application.
- Kriging: extremely flexible but complex, they are well-suited for deterministic applications and can handle applications with <50 factors.

As already stated previously, surrogate models rely on a training database \mathcal{L} whose size n_S should be sufficient to allow a proper estimation of the hyper-parameters, providing a nice estimation of the quantity of interest. The next two parts will introduce the concept of quality criteria and the basic problem of under- and over-fitting.

IV.1.1 Quality criteria definition

Once the model hyper-parameters are set (this step depends heavily on the chosen model, as discussed in the rest of this section), the quantity of interest can be estimated as $\hat{y} = M(\mathbf{x})$, where M represents the surrogate model. Only using the training database \mathcal{L} , one can have a first hint on whether this estimation can be considered reliable or not, thanks to various quality criteria. Among these, one can state for instance

$$MSE = \frac{1}{n_S} \sum_{i=1}^{n_S} (y_i - \hat{y}_i)^2, \quad R^2 = 1 - \frac{\sum_{i=1}^{n_S} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n_S} (y_i - \bar{y})^2}, \quad \text{and} \quad R_{\text{adj}}^2 = 1 - |1 - R^2| \left| \frac{n_S - 1}{n_S - (1 + n_X)} \right|,$$

EQUATION IV.1: Training database quality criteria definition

where $\hat{y}_i = \hat{y}(\mathbf{x}_i) = M(\mathbf{x}_i)$ is the prediction from the surrogate model at the i -Th location and \bar{y} is the expectation of the true value of our quantity of interest under consideration. MSE stands for *Mean Square Error* and should be as close to 0 as possible. The R^2 coefficient on the other hand should be as close to 1 as possible to state that the model might be valid. The last coefficient is just the "adjusted" R^2 , to regularise the fact that R^2 tends to be artificially close to 1 when the input space dimension n_X is large. A final caveat about MSE: unlike R^2 , MSE is not "scaled" or "normalised", so if one is dealing with a model whose results are small numerically, the surrogate model might be very wrong while having a small MSE. The scaling performed when computing R^2 (dividing by the variance of the true model) is coping for this.

The criteria discussed above are only using the training database \mathcal{L} and their interpretation can be misleading in case of over-fitting for instance (see Section IV.1.2 for a discussion on this matter). To prevent this, it is possible to use another database, often called validation database¹ that will be called hereafter \mathcal{P} and whose size is n_P . The predictivity coefficient Q^2 is analogous to the R^2 one, as one can write

$$Q^2 = 1 - \frac{\sum_{i=1}^{n_P} (y_i - \hat{y}(\mathbf{x}_i))^2}{\sum_{i=1}^{n_P} (\bar{y} - y_i)^2}.$$

EQUATION IV.2: Validation database predictivity criterion definition

The data are this time coming from the validation database ($\mathbf{x}_i \in \mathcal{P}$ and $y_i = M(\mathbf{x}_i)$) so they have not been used to train the surrogate model. The closer to 1 the Q^2 coefficient is, the more predictive the model can be considered.

Finally, for some surrogate models only, it is possible to perform a certain type of cross-validation for no or very limited cost. The idea is to obtain the prediction $\hat{y}_{\sim i} = M^{\sim i}(\mathbf{x}_i)$ where $M^{\sim i}$ represents the surrogate model for which the training database would have been \mathcal{L} without the i -Th location ($\hat{y}_{\sim i}$ is therefore called the *Leave-one-out* prediction). The resulting quality criteria are then defined as

$$MSE_{LOO} = \frac{1}{n_S} \sum_{i=1}^{n_S} (y_i - \hat{y}_{\sim i})^2 \quad \text{et} \quad Q_{LOO}^2 = 1 - \frac{\sum_{i=1}^{n_S} (y_i - \hat{y}_{\sim i})^2}{\sum_{i=1}^{n_S} (y_i - \bar{y})^2}.$$

EQUATION IV.3: Leave-one-out quality criteria definition

This happens to be very practical when the data are rare and expensive to produce: it is indeed complicated to "sacrifice" good data to test the validity.

¹ but not always, as one might find also test database, also this denomination might be confusing when discussing generalisation/regularisation as a method to prevent from over-fitting (see Section IV.1.2).

IV.1.2 Adapting the fitting strategy

This part describes the possible problem that one can meet when trying to train a surrogate model. Starting back from the situation described previously where \mathcal{L} and \mathcal{P} are respectively our training and validation database. If one assumes that the following relation $y_i = f(\mathbf{x}_i) + \varepsilon_i$ can exist (basically introducing a white noise $\varepsilon \sim \mathcal{N}(0, \sigma_e^2)$), then finding a proper surrogate model would mean finding the "function" M that can generalise to location out of the training database and whose error on the validation database can be written as:

$$\begin{aligned} E[(y - M(\mathbf{x}))^2] &= E[M(\mathbf{x}) - f(\mathbf{x})]^2 + E[(M(\mathbf{x}) - E[M(\mathbf{x})])^2] + \sigma_e^2. \\ &= (\text{Bias}[M(\mathbf{x})])^2 + \text{Var}[M(\mathbf{x})] + \sigma_e^2. \end{aligned}$$

This total error is the sum of three different contributions :

- an irreducible error, σ_e^2 , which is the lowest limit expected on a validation test;
- a bias term, $\text{Bias}[M(\mathbf{x})]$, which is the difference between the average prediction of our model and the correct value, that we are trying to predict;
- a variance term, $\text{Var}[M(\mathbf{x})]$, which is the variability of our model prediction for a given data point or a value, that tells us spread of our data.

Getting the best surrogate model is then a minimisation of both the variance and bias term, even though usually these two criteria are antagonist: the more complex the surrogate model is, the smaller the bias is becoming. Unfortunately, this reduction of the bias goes with an increase of the variance as the model tends to adapt itself more to the data. This is known as the "Variance-Bias dilemma" or the "Variance-Bias trade-off". This is sketched in Figure IV.1 that depicts the evolution of the bias, the variance and their sum, as a function of the complexity of the model.

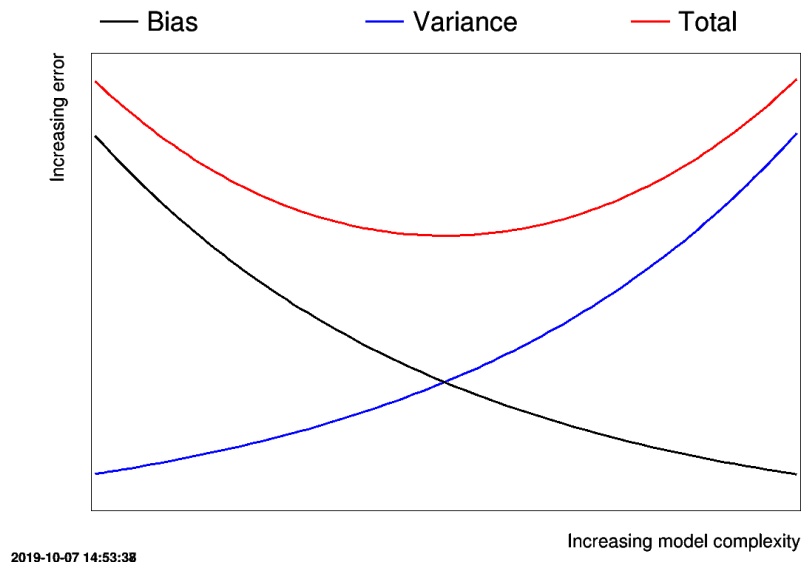


Figure IV.1: Sketch of the evolution of the bias, the variance and their sum, as a function of the complexity of the model.

There are usually three situations that one can come across when dealing with surrogate models, all depicted in Figure IV.2, where black and yellow dots respectively represent a training and a validation database. The ideal situation is the third one, in the right-hand side of the figure, where the green line passes in between all points, meaning that the

model predictions are close to all original values, disregarding the database they're coming from. This leads to a low bias and, as the variations are smooth and small through the entire range, a low variance as well. Let's now discuss the two other situations.

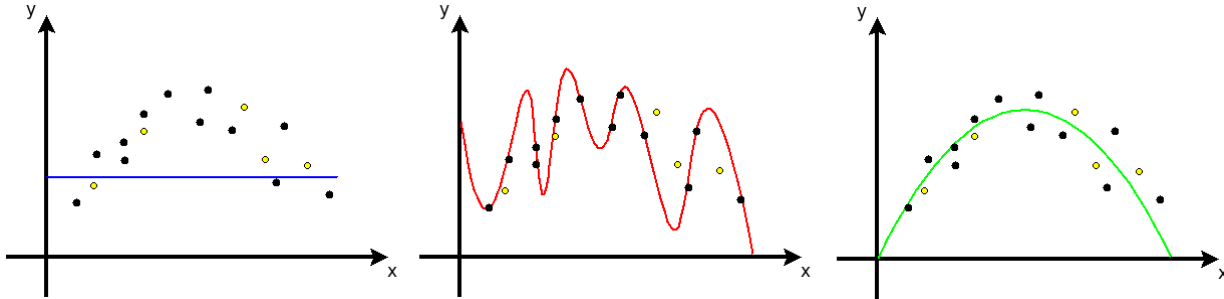


Figure IV.2: Sketches of under-trained (left), over-trained (middle) and properly trained (right) surrogate models, given that the black points show the training database, while the yellow ones show the testing database

The first situation, shown in the left part Figure IV.2, is what is called under-fitting: the model cannot capture the proper behaviour of the code and if one wants to estimate the MES, either for the training or validation database, the result in both cases will be poor. It generally arises when one assumes an over-simplified model either because of a lack of data or because of a mis-knowledge of the general trend of the problem under consideration. On the validation database, the prediction coming from the blue line will have an obvious low (null) variance and mainly a large bias (which is characteristic of under-fitting issue).

The opposite situation, represented in red in the middle of Figure IV.2, is what is called over-fitting: the model has learned (almost perfectly) the training database, and only it. It failed in capturing the proper trend of the code and if one applies this model to a validation database, the resulting prediction will be really poor. This situation corresponds to the case where the variance is becoming large. There are several strategies to avoid being in the situation but in general the first thing to check is that there is consistency between the the degrees of freedom of the model and the degree of freedom provided through the training database. One of the possible mechanism consists, for instance, in splitting the training database into a large sub-part used to train the model and a smaller one used to control how well the resulting model can predict unknown points. If one computes and represents the MSE for both sub-parts, as a function of the training steps, the resulting curves should look as those depicted in Figure IV.3. On the first hand, the MSE estimated on the training sub-part only gets better along the training, while on the other hand, the MSE computed on the control sample (called generalisation error, as it represents how well the model can be generalised to the rest of the input space) diminishes along the training error up to a plateau and then it grows again. This plateau is where one should stop the training procedure.

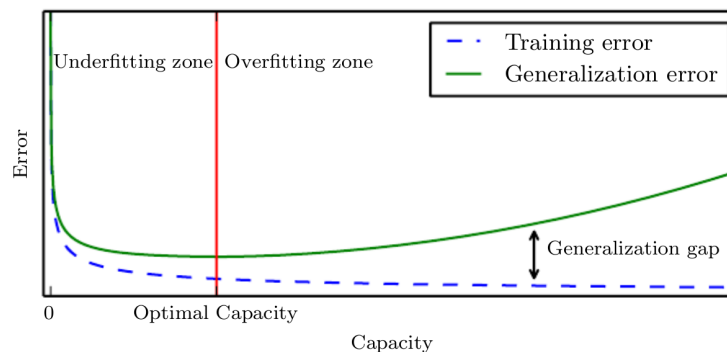


Figure IV.3: Evolution of the different kinds of error used to determine when does one start to over-train a model

IV.2 The linear regression

When using the linear regression, one assumes that there is only one output variable and at least one input variable. The data from the training database are stored here in a matrix $A(n_S, n_X)$ where n_S is the number of elements in the set and n_X is the number of input variables to be used. The idea is to write any output as $y = \sum_{i=1}^p \beta_i h_i$, where β are the regression coefficients and h_i , are the regressors: p simple functions depending on one or more input variables² that will be the new basis for the linear regression. A classical simple case is to have $p = n_X$ and $\{h_i(\mathbf{x}) = x_i\}_{i=1, \dots, n_X}$.

The regressor matrix is then constructed as $\mathbf{H}(n_S, p)$ and is filled with

$$\mathbf{H}(n_S, p) = \begin{pmatrix} h_1^1 & \dots & h_p^1 \\ h_1^2 & \dots & h_p^2 \\ \vdots & \ddots & \vdots \\ h_1^{n_S} & \dots & h_p^{n_S} \end{pmatrix}. \text{ Also } \mathbf{y} = (y^1 \ y^2 \ \dots \ y^{n_S}) \text{ and } \beta = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}$$

In the case where the number of points (n_S) is greater than the number of input variables (n_X), this estimation is just a minimisation of $\|\mathbf{y} - \mathbf{H}\beta\|^2$ which leads to the general form of the solution $\beta = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{y}$. From this, the estimated values of the output from the regression are computed as $\hat{\mathbf{y}} = \mathbf{H}\beta = \mathbf{H}(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{y} = \mathbf{P}\mathbf{y}$, if one calls $\mathbf{P} = \mathbf{H}(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T$.

As a result, a vector of parameters is computed and used to re-estimate the output parameter value. Few quality criteria are also computed, such as R^2 and the adjusted one R_{adj}^2 . There is an interesting interpretation of the R^2 criteria, in the specific case of a linear regression, coming from the previously introduced matrix \mathbf{P} , once considered as a projection matrix. It is indeed symmetrical and the following relation holds $\mathbf{P}^2 = \mathbf{P}$, so the estimation $\hat{\mathbf{y}}$ by the linear regression is a orthogonal projection of \mathbf{y} onto the subspace \mathcal{H} spanned by the column of \mathbf{H} . This is depicted in Figure IV.4 and it shows that the variance, $(y - \bar{y})^2$ can be decomposed into its component explained by the model $(\hat{y} - \bar{y})^2$ and the residual part, $(y - \hat{y})^2$. From this, the formula in Equation IV.1 can be also written

$$R^2 = 1 - \frac{\sum_{i=1}^{n_S} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n_S} (y_i - \bar{y})^2} = \frac{\sum_{i=1}^{n_S} (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n_S} (y_i - \bar{y})^2}$$

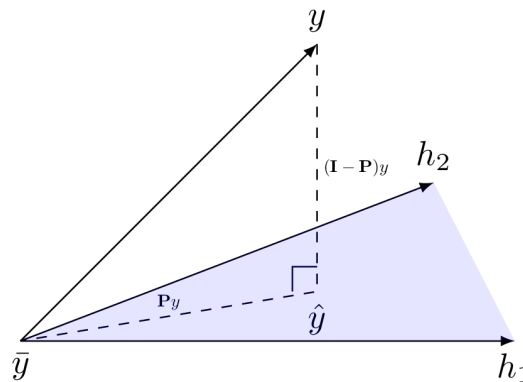


Figure IV.4: Schematic view of the projection of the original value from the code onto the subspace spanned by the column of \mathbf{H} (in blue).

²technically, one can also choose 1 as a regressor: this would bring a constant term in the regression.

For theoretical completeness, in most cases, the matrix \mathcal{H} is decomposed following a Singular Value Decomposition (SVD) such as $\mathbf{H}(n_S, p) = \mathbf{U}(n_S, n_S) \times \mathbf{S}(n_S, p) \times \mathbf{V}^T(p, p)$. In this context $\mathbf{U}(n_S, n_S)$ and $\mathbf{V}(p, p)$ are orthogonal matrices and $\mathbf{S}(n_S, p)$ is a diagonal matrix (that can also be stored as a singular values vector $S_{\text{vec}}(p)$). The diagonal matrix always exists, assuming that the number of samples is greater than the number of inputs ($n_S \geq p$). This has two main advantages the first one being that there is no matrix inversion to be performed, which implies that this procedure is more robust. The second advantage is when considering the \mathbf{P} matrix that links directly the output variable and its estimation through the *surrogate model*: it can now simply be written as $\mathbf{P} = \mathbf{U}\mathbf{U}^T$. This is highly practical once one knows that this matrix is used to compute the Leave-One-Out uncertainty, only considering its diagonal component.

IV.3 Chaos polynomial expansion

IV.3.1 Introduction

IV.3.1.1 Theoretical background

The concept of polynomial chaos development relies on the homogeneous chaos theory introduced by Wiener in 1938 [25] and further developed by Cameron and Martin in 1947 [26]. Using polynomial chaos (later referred to as *PC*) in numerical simulation has been brought back to the light by Ghanem and Spanos in 1991 [27]. The basic idea is that any square-integrable function can be written as $f(\mathbf{x}) = \sum_{\alpha} f_{\alpha} \Psi_{\alpha}(\mathbf{x})$ where $\{f_{\alpha}\}$ are the PC coefficients, $\{\Psi_{\alpha}\}$ is the orthogonal polynomial-basis. The index over which the sum is done, α , corresponds to a multi-index whose dimension is equal to the dimension of vector \mathbf{x} (i.e. n_X) and whose L1 norm ($|\alpha|_1 = \sum_{i=1}^{n_X} \alpha_i$) is the degree of the resulting polynomial. Originally written to deal with normal law, for which the orthogonal basis is Hermite polynomials, this decomposition is now generalised to few other distributions, using other polynomial orthogonal basis (the list of those available in Uranie is shown in Table IV.1). This decomposition can be helpful in many ways: it can first be used

Distribution \ Polynomial type	Legendre	Hermite	Laguerre	Jacobi
Uniform	X			
LogUniform	X			
Normal		X		
LogNormal		X		
Exponential			X	
Beta				X

Table IV.1: List of best adapted polynomial-basis to develop the corresponding stochastic law

as a *surrogate model* but it gives also access, through the value of its coefficient, to the sensitivity index (this will be first introduced in Section IV.3.1.2 and further developed in Chapter V).

IV.3.1.2 Simple example

We'll discuss here a simple example of polynomial chaos development and its implication. In the case where a system is depending on two random variables, X_U and X_N that follow respectively an uniform and normal distribution, giving rise to a single output Y . Following the remark about square-integrable functions, both inputs can be decomposed on a specific orthogonal polynomial-basis, such as $X_U = \sum_{\alpha} f_{\alpha}^U \mathcal{L}_{\alpha}$, and $X_N = \sum_{\alpha} f_{\alpha}^N \mathcal{H}_{\alpha}$, where $\{f_{\alpha}^U\}$ and $\{f_{\alpha}^N\}$ are the PC coefficients that respectively multiply the Legendre (\mathcal{L}) and Hermite (\mathcal{H}) polynomials, for the uniform and normal law and where α is the multi-index (here of dimension 1) over which the sum is done. These basis are said to be orthogonal because for any degrees k_1 and k_2 , taking the Legendre case as an example, one can write $\int_{\Omega} \mathcal{L}_{k_1} \mathcal{L}_{k_2} d\Omega = 0$, for $k_1 \neq k_2$.

It is now possible to write the output, Y , as a function of these polynomials. For the i -Th simulation,

$$Y^i = \sum_{|\alpha|_1 \leq n_X} \beta_\alpha \Psi_\alpha(X_U^i, X_N^i),$$

EQUATION IV.4: Chaos polynomial function

where α is the multi-index of dimension 2 ($\alpha \in \mathbb{N}^2$) over which the sum is performed. The Ψ polynomials are built by tensor products of the inputs basis following the previously defined degree. In the specific case of the simple example discussed here, this leads to a decomposition of the output that can be written as

$$\begin{aligned} Y^i &= \beta_{0,0} \quad (|\alpha|_1 = 0) \\ &+ \beta_{1,0} \mathcal{L}_1(X_U^i) + \beta_{0,1} \mathcal{H}_1(X_N^i) \quad (|\alpha|_1 = 1) \\ &+ \beta_{1,1} \mathcal{L}_1(X_U^i) \mathcal{H}_1(X_N^i) + \beta_{2,0} \mathcal{L}_2(X_U^i) + \beta_{0,2} \mathcal{H}_2(X_N^i) \quad (|\alpha|_1 = 2) \\ &+ \beta_{2,1} \mathcal{L}_2(X_U^i) \mathcal{H}_1(X_N^i) + \beta_{1,2} \mathcal{L}_1(X_U^i) \mathcal{H}_2(X_N^i) + \beta_{3,0} \mathcal{L}_3(X_U^i) + \beta_{0,3} \mathcal{H}_3(X_N^i) \quad (|\alpha|_1 = 3) \\ &+ \dots \end{aligned}$$

EQUATION IV.5: polynomial chaos decomposition

From this development, it becomes clear that a threshold must be chosen on the order of the polynomials used, as the number of coefficient is growing quickly, following this rule $N_{\text{coeff}} = \frac{(n_X + p)!}{n_X! p!}$, where p is the cut-off chosen on the polynomial degree. In this example, if we choose to use $p = 2$, this leads to only 6 coefficients to be measured: $\beta_{0,0}, \beta_{1,0}, \beta_{0,1}, \beta_{2,0}, \beta_{0,2}, \beta_{1,1}$. Their estimation is discussed later.

These coefficients are characterising the *surrogate model* and can be used, **when the inputs are independent**, to estimate the corresponding *Sobol's coefficients* (a deeper discussion about these coefficients and their meaning can be found in Section V.1). For the uniform and normal example, the first order coefficients are respectively given by

$$S_1^U = \frac{\beta_{1,0}^2 + \beta_{2,0}^2}{\text{Var}(Y)} \quad \text{and} \quad S_1^N = \frac{\beta_{0,1}^2 + \beta_{0,2}^2}{\text{Var}(Y)},$$

whereas the total order coefficients are respectively given by

$$S_T^U = \frac{\beta_{1,0}^2 + \beta_{2,0}^2 + \beta_{1,1}^2}{\text{Var}(Y)} \quad \text{and} \quad S_T^N = \frac{\beta_{0,1}^2 + \beta_{0,2}^2 + \beta_{1,1}^2}{\text{Var}(Y)}.$$

The complete variance of the output, can also be written as

$$\text{Var}(Y) = \sum_{|\alpha|_1 \leq 2} \beta_\alpha^2$$

Warning



One can use chaos polynomial expansion with a training database without knowing the probability laws used to generate it, as long as the polynomial coefficients estimation is done with a regression method and not an integration one (for which the integration-oriented design-of-experiments is made specifically knowing the laws, see discussion in Section IV.3.2.1 and Section IV.3.2.2).

The interpretation of the polynomial coefficients as *Sobol's coefficients*, on the other hand, is strongly relying on the hypothesis that the probability laws have been properly defined, so it becomes not suitable if the training database is made without knowing the probability laws. The explanations for this is way beyond the scope of this documentation but more information can be found in the literature (for instance in [29]).

IV.3.2 Nisp in a nutshell

The wrapper of the Nisp library, Nisp standing for *Non-Intrusive Spectral Projection*, is a tool allowing to access to Nisp functionality from the Uranie platform. The main features are detailed below.

The Nisp library [28] uses spectral methods based on polynomial chaos in order to provide a surrogate model and allow the propagation of uncertainties if they arise in the numerical models. The steps of this kind of analysis, using the Nisp methodology are represented schematically in Figure IV.5 and are introduced below:

- Specification of the uncertain parameters x_i ,
- Building stochastic variables associated ξ_i ,
- Building a design-of-experiments
- Building a polynomial chaos, either with a regression or an integration method (see Section IV.3.2.1 and Section IV.3.2.1)
- Uncertainty and sensitivity analysis

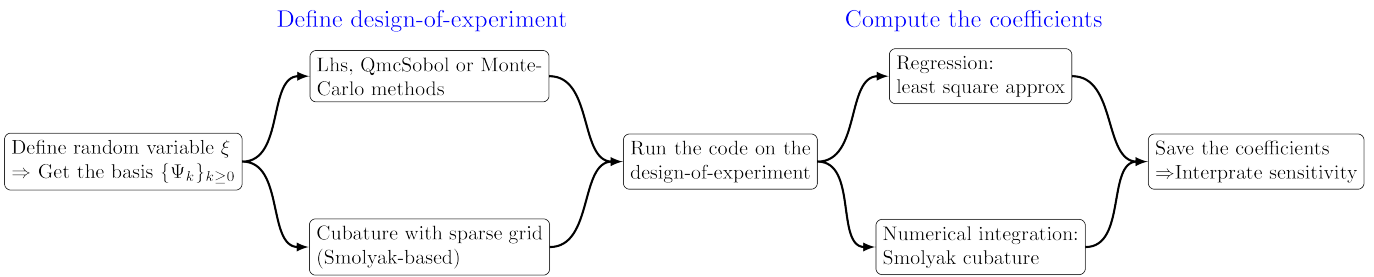


Figure IV.5: Schematic view of the Nisp methodology

IV.3.2.1 The regression method

The regression method is simply based on a least-squares approximation: once the design-of-experiments is done, the vector of output $\mathbf{y}(n_S)$ is computed with the code. The regression coefficients β are estimated considering that every computed output points can be represented following Equation IV.5. By writing the correspondence matrix $H(n_S, p)$ and the coefficient-vector β , this estimation is just a minimisation of $\|\mathbf{y} - H\beta\|^2$, where, once back to our simple example from Section IV.3.1.2 for illustration purpose,

$$\mathbf{y} = (Y^1 \ Y^2 \ \dots \ Y^{n_S}), \quad H = \begin{pmatrix} \Psi_1(X_U^1, X_N^1) & \dots & \Psi_p(X_U^1, X_N^1) \\ \Psi_1(X_U^2, X_N^2) & \dots & \Psi_p(X_U^2, X_N^2) \\ \vdots & \ddots & \vdots \\ \Psi_1(X_U^{n_S}, X_N^{n_S}) & \dots & \Psi_p(X_U^{n_S}, X_N^{n_S}) \end{pmatrix}, \quad \text{and } \beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix},$$

As already stated in Section IV.2, this leads to write the general form of the solution as $\beta = (H^T H)^{-1} H^T \mathbf{y}$ which also shows that the way the design-of-experiments is performed can be optimised depending on the case under study (and might be of the utmost importance in some rare case).

In order to perform this estimation, it is mandatory to have more points in the design-of-experiments than the number of coefficient to be estimated (in principle, following the rule $n_S \geq 1.5 \times N_{\text{coeff}}$ leads to a safe estimation).

IV.3.2.2 The integration method

The integration method relies on a more "complex" design-of-experiments. It is indeed recommended to have dedicated design-of-experiments, made with a Smolyak-based algorithms (as the ones cited in Figure IV.5). These design-of-experiments are sparse-grids and usually have a smaller number of points than the regularly-tensorised approaches. In this case, the number of samples has not to be specified by the user. Instead, the argument requested describes the level of the design-of-experiments (which is closely intricated, as the higher the level is, the larger the number of samples is). Once this is done, the calculation is performed as a numerical integration by quadrature methods, which requires a large number of computations.

In the case of Smolyak algorithm, this number can be expressed by the number of dimensions n_X and the requested level l as $N_d = 2^l \times l^{n_X-1}$ which shows an improvement with respect to the regular tensorised formula for quadrature ($\sim 2^{l \cdot n_X}$).

IV.4 The artificial neural network

The Artificial Neural Networks (ANN) in Uranie are Multi Layer Perceptron (MLP) with one or more hidden layer (containing n_H^i neurons, where i is use to identify the layer) and one or more output variable.

IV.4.1 Introduction to the formal neuron

The concept of formal neuron, has been proposed in 1943, after observing the way biological neurons are intrinsically connected [21]. This model is a simplification of the various range of functions dealt by a biological neuron, the formal one (displayed in Figure IV.6) being requested to satisfy only the two following:

- summing the weighted input values, leading to an output value called neuron's activity $a = \sum_{i=1}^{n_X} \omega_i x_i$, where $\omega_1 \dots \omega_{n_X}$ are the synaptic weights of the neuron.
- emitting a signal (whether the output level goes beyond a chosen threshold or not) $s = f(a + \theta)$ where f and θ are respectively the transfer function and the bias of the neuron.

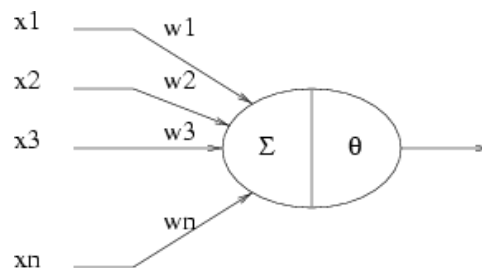


Figure IV.6: Schematic description of a formal neuron, as seen in McCulloch and Pitts [21].

One can introduce a shadow input defined as $x_0 = 1$ (or -1), which lead to consider the bias as another synaptic weight $\omega_0 = \theta$. The resulting emitted signal is written as

$$s = f\left(\sum_{i=0}^{n_X} \omega_i x_i\right)$$

There are a large variety of transfer functions possible, and an usual starting point is the sigmoid family, defined with three real parameters, c , r and k , as $f_{c,k,r}(x) = c \frac{e^{kx}-1}{e^{kx}+1} + r$. Setting these parameters to peculiar values leads to known functions as the hyperbolic tangent and the logistical function, shown in Figure IV.7 and defined as

$$f_{1,2,0}(x) = \frac{e^{2x}-1}{e^{2x}+1} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \tanh(x) \quad \text{and} \quad f_{1/2,1,1/2}(x) = \frac{1}{2} \frac{e^x - 1}{e^x + 1} + \frac{1}{2} = \frac{1}{1 + e^{-x}}$$

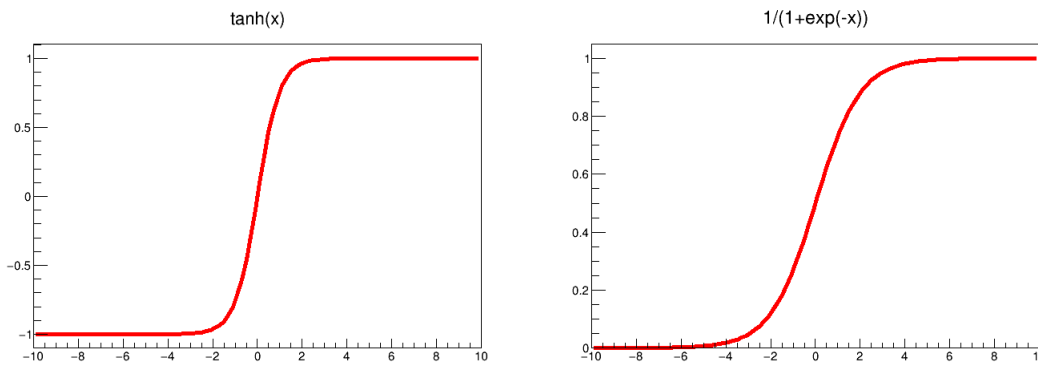


Figure IV.7: Example of transfer functions: the hyperbolic tangent (left) and the logistical one (right)

IV.4.2 The working principle

The artificial neural network conception and working flow has been first proposed in 1962 [22] and was called the perceptron. The architecture of a neural network is the description of the organisation of the formal neurons and the way they are connected together. There are two main topologies:

- complete: all the neurons are connected to the others.
- by layer: the neurons on a layer are connected to all neurons on the previous layer and the following one.

The general organisation of the neural network is detailed in three steps in the following part, and summarised in Figure IV.8. The first layer, where the vector of entries is stored, is called the input layer. The last one is called the output layer. In between, one can put from zero (leading to a Rosenblatt perceptron) to as many hidden layers as wanted. With one or more hidden layers, the neural networks are called *Multi Layer Perceptron* (MLP) and have been studied heavily since early 1990 [23, 24]. In Uranie, the architecture has to be chosen by the user (as least one hidden layer).

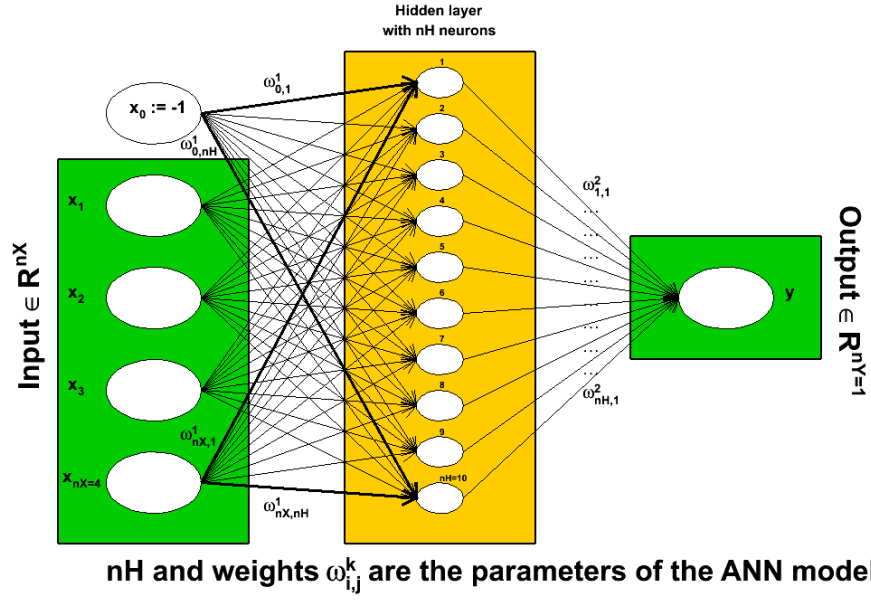


Figure IV.8: Schematic description of the working flow of an artificial neural network as used in Uranie

The first step is the definition of the problem: what are the input variables under study, how many neurons will be created in how many hidden layers, what is the chosen activation function. When choosing the architecture of an artificial neural network, one should keep in mind that the number of points used to perform the training should obviously be greater than the number of parameters to be estimated, *i.e.* the number of synaptic weights (the usual rule of thumb being having a factor 1.5 between data points and the number of coefficients). From the explanation given previously, the number of coefficients for a single hidden layer neural network is $n_H \times (n_X + 1)$ where n_H is the number of neurons chosen. This formula can be generalised to multi-layer cases, as

$$N_{\text{Coeff}} = \sum_{i=1}^{n_L} n_H^i \times (n_H^{i-1} + 1)$$

In this equation, n_H^i is the number of hidden neurons per layer, for $i \in [1, n_L]$ where n_L is the number of layer, and n_H^0 is the number of inputs ($n_H^0 = n_X$). Defining an architecture is quite tricky and depends on the problem under consideration. Going multi-layer is a way to reduce the number of coefficients to be estimated when the number of inputs is large: with 9 inputs variables, a mono-layer with 8 neurons will have 80 synaptic weights while a network with 3 layers and 3 neurons each, will have 9 neurons in total but only 54 weights to be determined.

The second step is the training of the ANN. This step is crucial and many different techniques exist to achieve it but, as this note is not supposed to be exhaustive, only the one considered in the Uranie implementation will be discussed. A learning database should be provided, composed of a set of inputs and the resulting output (the ensemble \mathcal{L} discussed in Section IV.1). From that, two mechanisms are run simultaneously:

- the learning itself. By varying all the synaptic weights contained in the parameter θ , the aim is to produce the output set $\hat{y} = f_\theta(\mathbf{x})$, that would be as close as possible to the output stored in \mathcal{L} and keep the best configuration (denoted as θ^*). The difference between the real outputs and the estimated ones are measured through a loss function which could be, in the case of regression, a quadratic loss function such as

$$L(y, \hat{y}) = \frac{1}{2} \|y - \hat{y}\|^2$$

From there, one can define the risk function $R(\theta)$ (also called cost or energy function) used to transform the optimal parameters search into a minimisation problem. The empirical risk function can indeed be written as

$$R(\theta) = \frac{1}{n_S} \sum_{i=1}^{n_S} L(y_i, f_{\theta}(\mathbf{x}_i))$$

- the regularisation. This step is made to avoid all over-fitting problem, meaning that the neural network would be trained only for the \mathcal{L} ensemble which might not be representative of the rest of the input space. To avoid this, the learning database is split into two sub-parts: one for the learning as described in the previous item, and one to prevent the over-fitting to happen. This is done by computing for every newly tested parameter set θ , the generalised error (computed as the average error over the set of points not used in the learning procedure). While it is expected that the risk function is becoming smaller when the number of optimisation step is getting higher, the generalised error is also becoming smaller at first, but then it should stabilise and even get worse. This flattening or worsening is used to stop the optimisation.

This procedure is stochastic: the splitting of the \mathcal{L} ensemble is done using a random generator, so does the initialisation of the synaptic weights for all the formal neurons.

Finally, the constructed neural network can be (and should be) exported: the weight initialisation, but also the way the split is performed between the test and training basis, are randomised leading to different results every time one redo the training procedure.

IV.5 The kriging method

First developed for geostatistic needs, the kriging method, named after D. Krige and also called Gaussian Process method (denoted GP hereafter) is another way to construct a *surrogate model*. It recently became popular thanks to a series of interesting features:

- it provides a prediction along with its uncertainty, which can then be used to plan the simulations and therefore improve the predictions of the *surrogate model*
- it relies on relatively simple mathematical principle
- some of its hyper-parameters can be estimated in a Bayesian fashion to take into account *a priori* knowledge.

Kriging is a family of interpolation methods developed in the 1970s for the mining industry [30]. It uses information about the "spatial" correlation between observations to make predictions with a confidence interval at new locations. In order to produce the prediction model, the main task is to produce a spatial correlation model. This is done by choosing a correlation function and search for its optimal set of parameters, based on a specific criterion.

The *gpLib* library [31] provides tools to achieve this task. Based on the gaussian process properties of the kriging [32], the library proposes various optimisation criteria and parameter calculation methods to find the parameters of the correlation function and build the prediction model.

The present chapter describes the integration of the *gpLib* inside Uranie, from a methodological point of view. For a more practical point of view, see the *gpLib* tutorial [33].

IV.5.1 Theoretical introduction

The modelisation relies on the assumption that the deterministic output $y(\mathbf{x})$ can be written as a realisation of a gaussian process $Y(\mathbf{x})$ that can be decomposed as $Y(\mathbf{x}) = m(\mathbf{x}) + Z(\mathbf{x})$ where $m(\mathbf{x})$ is the deterministic part that

describes the expectation of the process and $Z(\mathbf{x})$ is the stochastic part that allows the interpolation. This method can also take into account the uncertainty coming from the measurements. In this case, the previously-written $Y(\mathbf{x})$ is referred to as $Y_{\text{Real}}(\mathbf{x})$ and the gaussian process is then decomposed into $Y_{\text{Obs}}(\mathbf{x}) = m(\mathbf{x}) + Z(\mathbf{x}) + \varepsilon(\mathbf{x})$, where $\varepsilon(\mathbf{x})$ is the uncertainty introduced by the measurement.

The first step is to construct the model from the n_S known measurements, that will be hereafter called the training site. To do so, a parametric correlation function has to be chosen amongst a list of provided one (discussed in Section IV.5.1.1); a deterministic trend can also be imposed to bring more information on the behaviour of the output expectation. These steps define the list of hyper-parameters to be estimated, which is done in Uranie through an optimisation loop. The training site and the estimated hyper-parameters constitute the kriging model that can then be used to predict the value of a new sets of points.

It is possible, at this stage, even before applying the kriging model to a new set of points, to make a verification of the covariance function at hand. This is done in Uranie using a *Leave-One-Out* (LOO) technique. This method consists in the prediction of a value for y_i using the rest of the known values in the training site, i.e. $y_1, \dots, y_{i-1}, y_{i+1}, \dots, y_{n_S}$ for $i = 1, \dots, n_S$. From there, it is possible to use the LOO prediction $\hat{y}_{\sim i}$ and the expectation \bar{y} to estimate both the MSE_{LOO} and Q_{LOO}^2 (see Section IV.1.1 for completeness). The first criterion should be close to 0 while, if the covariance function is correctly specified, the second one should be close to 1. Another possible test to check whether the model seems reasonable consists in using the predictive variance vector $(\sigma_{\hat{y}_{\sim i}}^2)_{i=1, \dots, n_S}$ to look at the distribution of the ratio $(y_i - \hat{y}_{\sim i}) / \sqrt{\sigma_{\hat{y}_{\sim i}}^2}$ for every point in the training site. A good modelling should result in a standard normal distribution, so one can find in the literature (see [42] for instance) quality criteria proposal such as having at least 99.7% of the points comply with

$$\frac{y_i - \hat{y}_{\sim i}}{\sqrt{\sigma_{\hat{y}_{\sim i}}^2}} \in [-3, 3].$$

The next section will introduce the correlation functions implemented, that can be used to learn from the data and set the value of the corresponding hyper-parameters. This is a training step which should lead to the construction of the covariance matrix K of the stochastic part $Z(\mathbf{x})$ introduced previously. From there, the predictions to new points can be performed, as discussed in Section IV.5.1.2.

IV.5.1.1 Correlation functions implemented

To end this introduction it might be useful to introduce one of the most used correlation function (at least a very-general one): the Matern function. It uses the Gamma function Γ and the modified Bessel function of order ν called hereafter K_ν . This ν parameter describes the regularity (or smoothness) of the trajectory (the larger it is, the smoother the function will be) which should be greater than 0.5. In one dimension, with δx the distance, this function can be written as

$$c(\delta x) = \sigma^2 \frac{1}{\Gamma(\nu) 2^{\nu-1}} \left(2\sqrt{\nu} \frac{\delta x}{l} \right)^\nu K_\nu \left(2\sqrt{\nu} \frac{\delta x}{l} \right).$$

EQUATION IV.6: General Matern function

In this function, l is the correlation length parameter, which has to be positive. The larger l is, the more Y is correlated between two fixed locations x_1 and x_2 and hence, the more the trajectories of Y vary slowly with respect to x . As discussed in Section IV.5.1 this is a crucial part to specify a GP and there are many possible functions implemented in Uranie. In the following list, l are the correlation lengths and ν are the regularity parameters. For the rest of this discussion, the variance parameter σ^2 will be glossed over as it will be determined in all case and it is global (one value for a problem disregarding the number of input variable). The impact of these parameters, variance, correlation

length and smoothness are displayed respectively in Figure IV.9, Figure IV.10 and Figure IV.11 (these plots are taken from [32]).

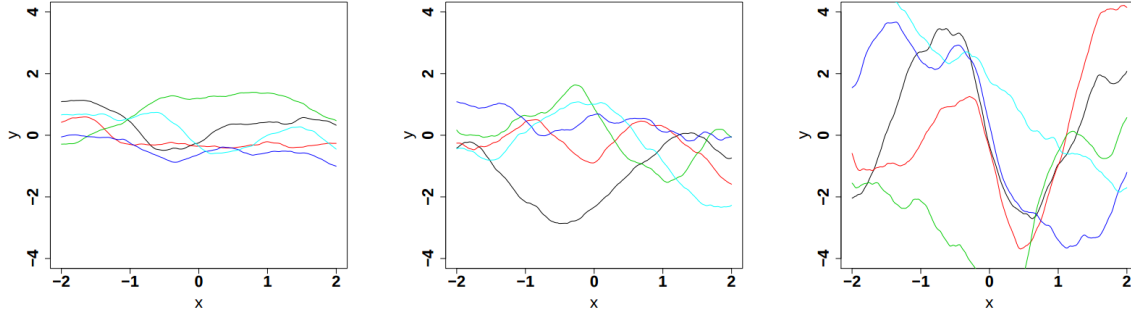


Figure IV.9: Influence of the variance parameter in the Matern function once fix at 0.5, 1 and 2 (from left to right). The correlation length is set to 1 while the smoothness is set to $3/2$.

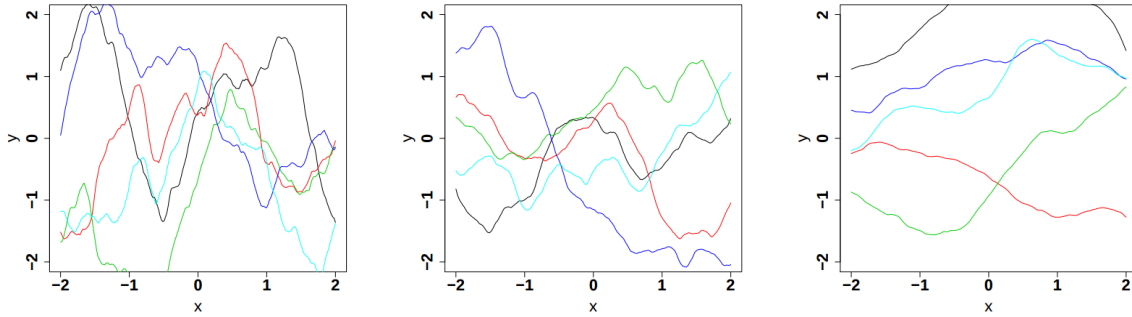


Figure IV.10: Influence of the correlation length parameter in the Matern function once fix at 0.5, 1 and 2 (from left to right). The variance is set to 1 while the smoothness is set to $3/2$.

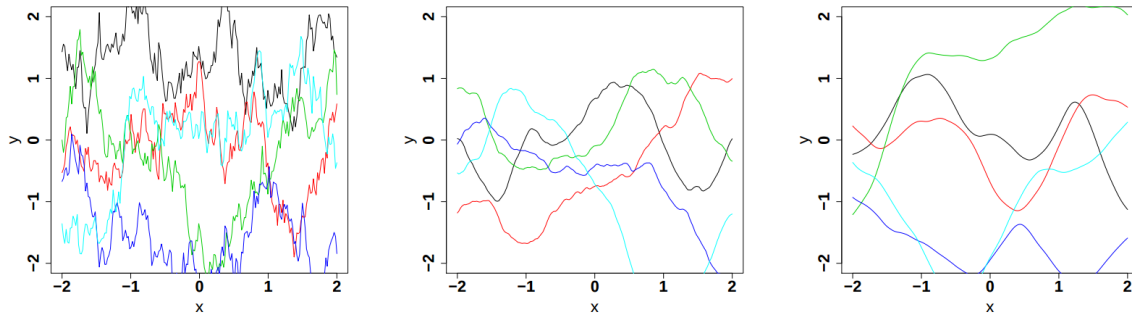


Figure IV.11: Influence of the smoothness parameter in the Matern function once fix at 0.5, 1.5 and 2.5 (from left to right). Both the variance and the correlation length are set to 1.

- Gauss: it is defined with one parameter per dimension, as $c(\delta\mathbf{x}) = \exp \left[-\sum_{k=1}^{n_X} \left(\frac{\delta x_k}{l_k} \right)^2 \right]$.
- Isogauss: it is defined with one parameter only, as $c(\delta\mathbf{x}) = \exp \left[-\frac{|\delta\mathbf{x}|^2}{l^2} \right]$.

- Exponential: it is defined with two parameters per dimension, as $c(\delta \mathbf{x}) = \exp \left[-\sum_{k=1}^{n_X} \left(\frac{|\delta x_k|}{l_k} \right)^{p_k} \right]$, where p are the power parameters. If $p = 2$, the function is equivalent to the Gaussian correlation function.
- MaternI: the most general form, it is defined with two parameters per dimension, as

$$c(\delta \mathbf{x}) = \prod_{k=1}^{n_X} \frac{1}{\Gamma(v_k) 2^{v_k-1}} \left(2\sqrt{v_k} \frac{\delta x_k}{l_k} \right)^{v_k} K_{v_k} \left(2\sqrt{v_k} \frac{\delta x_k}{l_k} \right)$$

- MaternII: it is defined as maternI, with only one common smoothness (leading to $n_X + 1$ parameters).
- MaternIII: it first compute a distance as $\delta = \sqrt{\sum_{k=1}^{n_X} \left(\frac{\delta x_k}{l_k} \right)^2}$ and then use Equation IV.6 with $\delta x = \delta$ (leading to $n_X + 1$ parameters).
- Matern1/2: it is equivalent to maternIII, when $\nu = 1/2$
- Matern3/2: it is equivalent to maternIII, when $\nu = 3/2$
- Matern5/2: it is equivalent to maternIII, when $\nu = 5/2$
- Matern7/2: it is equivalent to maternIII, when $\nu = 7/2$

The choice has to be made on a case-by-case basis, knowing the behaviour of the various inputs along with the way the output evolves. A gaussian function is infinitely derivable, so it is expected to work particularly well for cases where the output has a smooth trend, whereas the exponential, when considering small powers, *i.e.* $p = 0.5$ for instance, could better describe a more erratic output behaviour. The Matern function can easily go from one of this performance to the other by changing the smoothness. Figure IV.12 presents the evolution of the different covariance functions implemented in Uranie.

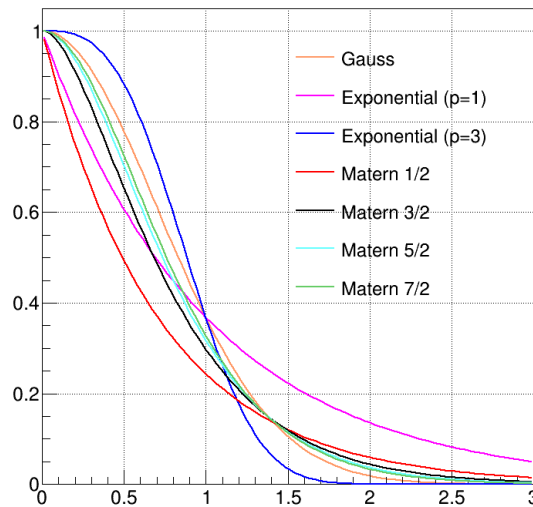


Figure IV.12: Evolution of the different covariance functions implemented in Uranie.

IV.5.1.2 From the training to the prediction

The predictions are done after the estimations of the hyper-parameters (σ^2, θ) of the covariance process, along with the errors if this is requested and the trend parameters as-well. As a reminder, the probabilistic model is depicted as

$$Y_{obs}(\mathbf{x}_i) = \mathbf{h}(\mathbf{x}_i)^T \boldsymbol{\beta} + Z(\mathbf{x}_i), \quad i = 1, 2, \dots, n_S$$

where $\boldsymbol{\beta} \sim \mathcal{N}(\boldsymbol{\beta}_{prior}, \mathbf{Q}_{prior})$ and Z is a gaussian process independent from $\boldsymbol{\beta}$, possibly including uncertainty measurements. As already stated, K is the covariance matrix of the stochastic part $Z(\mathbf{x}_i), i = 1, 2, \dots, n_S$ and $Y_{obs}^{n_S}$ the gaussian vector defined as $Y_{obs}(\mathbf{x}_i), i = 1, 2, \dots, n_S$. For the sake of simplicity, we will discuss only deterministic trend here where $\boldsymbol{\beta}$ are supposed constant and are estimated from the regressor matrix H , constructed from the $\mathbf{h}(\mathbf{x}_i), i = 1, 2, \dots, n_S$, as already discussed in Section IV.2. For a more general discussion on a bayesian approach to define the trend, see [32].

Every prediction of the gaussian process at a new location \mathbf{x}_{new} can be calculated from the conditional gaussian law $Y_{real}(\mathbf{x}_{new})|Y_{obs}^{n_S}$ which can be easily obtained from the joined law $(Y_{real}(\mathbf{x}_{new}), Y_{obs}^{n_S})$ using the gaussian conditioning theorem³. Starting from the simple case where one wants to get the best estimate $\hat{y}(\mathbf{x}_{new})$ and its conditional variance $\hat{\sigma}^2(\mathbf{x}_{new})$ for a single new location \mathbf{x}_{new} . Both quantities can be expressed from the previous matrix, using $\mathbf{h}(\mathbf{x}_{new})$ the regressor vector estimated at this new location and $\mathbf{r}(\mathbf{x}_{new})$ the vector (of size n_S) of covariance computed between this new location and the training set. The results are provided by the following equations:

$$\begin{aligned} \hat{y}(\mathbf{x}_{new}) &= \mathbf{h}(\mathbf{x}_{new})^T \hat{\boldsymbol{\beta}} + \mathbf{r}(\mathbf{x}_{new})^T K^{-1} (\mathbf{y}_{obs} - H \hat{\boldsymbol{\beta}}) \\ \hat{\sigma}^2(\mathbf{x}_{new}) &= \sigma^2 - \mathbf{r}(\mathbf{x}_{new})^T K^{-1} \mathbf{r}(\mathbf{x}_{new}) \\ &\quad + (\mathbf{h}(\mathbf{x}_{new})^T - \mathbf{r}(\mathbf{x}_{new})^T K^{-1} H) (H^T K^{-1} H)^{-1} (\mathbf{h}(\mathbf{x}_{new}) - H^T K^{-1} \mathbf{r}(\mathbf{x}_{new})) \end{aligned}$$

EQUATION IV.7: Best estimate and its conditional variance for a single new location.

In the case where multiple locations have to be estimated and one wants to keep track of their possible correlation, a more complex formula is written, starting from Equation IV.7. The regressor vector is now written as H_{new} , a regressor matrix gathering all new locations estimation, R is a $n_{new} \times n_S$ matrix that gathers covariance computed between all new locations and the training set and finally K_{new} is introduced as the covariance matrix between all the new input locations (with a given size $n_{new} \times n_{new}$). The results are then provided by the following equations:

$$\begin{aligned} \hat{\mathbf{y}}(X_{new}) &= H_{new} \hat{\boldsymbol{\beta}} + R K^{-1} (\mathbf{y}_{obs} - H \hat{\boldsymbol{\beta}}) \\ \hat{\Gamma}(X_{new}) &= K_{new} - R K^{-1} R^T + (H_{new} - R K^{-1} H) (H^T K^{-1} H)^{-1} (H_{new}^T - H^T K^{-1} R^T) \end{aligned}$$

EQUATION IV.8: Best estimates and their covariance matrix, for a set of new locations.

The main interest of the second procedure is to provided the complete covariance matrix of the estimation so if one wants to investigate the residuals, when dealing with a validation database, the possible correlation between location can be taken into account (through a whitening procedure for instance, that is partly introduced in Section III.2.2) to

³ For $\begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{pmatrix}$ a gaussian vector, following the left-hand side relation below, meaning that \mathbf{y}_1 and $\bar{\mathbf{y}}_1$ are of size n_1 and \mathbf{y}_2 and $\bar{\mathbf{y}}_2$ are of size n_2 and the global covariance matrix is made out of a R_{11} matrix of size $n_1 \times n_1$, a R_{22} matrix of size $n_2 \times n_2$ and a R_{12} matrix of size $n_1 \times n_2$ defining also $R_{21} = R_{12}^T$. Under the hypothesis that R_{22} can be inverted, the law of \mathbf{y}_1 conditionally to \mathbf{y}_2 is gaussian and can be written as the right-hand side equation below.

$$\begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{pmatrix} \sim \mathcal{N} \left(\begin{pmatrix} \bar{\mathbf{y}}_1 \\ \bar{\mathbf{y}}_2 \end{pmatrix}, \begin{pmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{pmatrix} \right), \text{ if } R_{22} \text{ can be inverted, } (\mathbf{y}_1 | \mathbf{y}_2) \sim \mathcal{N}(\bar{\mathbf{y}}_1 + R_{12} R_{22}^{-1} (\mathbf{y}_2 - \bar{\mathbf{y}}_2), R_{11} - R_{12} R_{22}^{-1} R_{21})$$

have proper residuals distribution. There are different ways to that, see [18], the ones used here are either based on Cholesky decomposition or eigen-value decomposition.

Finally, one interesting thing to notice is the fact that the prediction estimation needs both the inverse of the covariance matrix K^{-1} and the regressor matrix H . This means that with a large training database, this would mean keep this possibly two large matrices for every single new location.

IV.5.1.3 A simple example

We illustrate the kriging method on a very simple model, *i.e.* an uni-dimensional function, and display the resulting kriging model prediction. We use a training site with 6 points along with a test basis that is made of about 100 points. The covariance function used is a Gaussian one (left) and a Matern one (right), precisely a Matern3/2, presented in Figure IV.13. In this figure, one can see the training sites (the six black points), the real values of the testing site (the blue crosses), the predicted value from the kriging model (the red line) and the uncertainty band on this prediction (the red-shaded band). Both the MSE and Q^2_{LOO} from LOO are also indicated showing that in this particular case, the Gaussian choice is better than the Matern one.

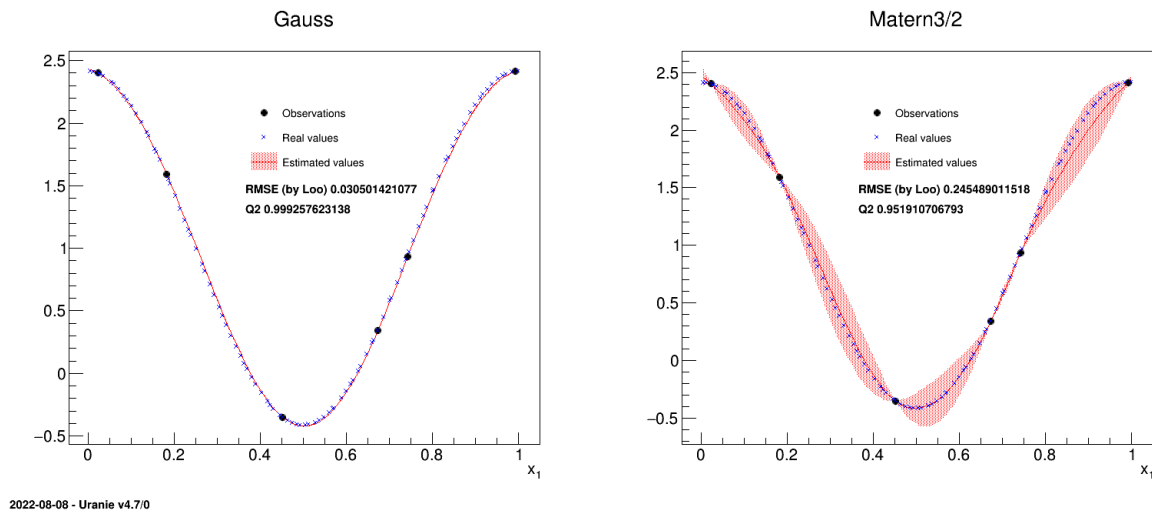


Figure IV.13: Example of kriging method applied on a simple uni-dimensional function, with a training site of six points, and tested on a basis of about hundred points, with either a gaussian correlation function (left) or a matern3/2 one (right).

IV.5.2 Running a kriging

The kriging procedure in Uranie can be schematised in five steps, depicted in Figure IV.14. Here is a brief description of the steps:

- get a training site. Either produced by a design-of-experiments from a model definition, or taken from anywhere else, it is mandatory to get this basis (the larger, the better).
- set the parameter's values. It can be set by hands, but it is highly recommended to proceed through an optimisation, to get the best possible parameters.
- build the kriging method.
- test the obtained kriging model. This is done by running the kriging model over a new basis .

Get training basis

Set parameter's value

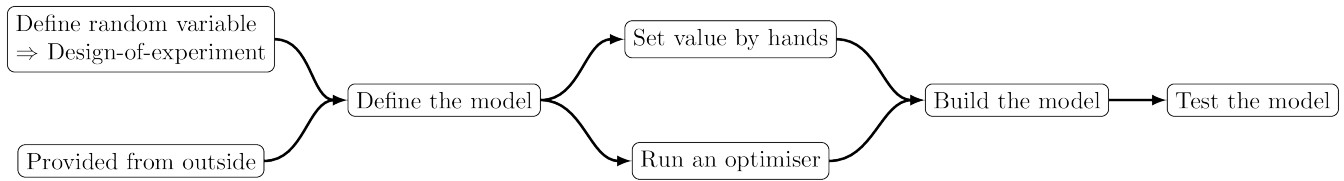


Figure IV.14: Schematic description of the kriging procedure as done within Uranie

Chapter V

Sensitivity analysis

V.1 Brief reminder of theoretical aspects

In this section, we will briefly remind the different ways to measure the sensitivity of an output to the inputs of the model. A theoretical introduction will give a glimpse of different techniques and formalism, going from the simplest case to the more complex one. It should remain at a very basic level, only to introduce notions since more details can be found in many references [34, 35, 36].

The list of methods available in Uranie will also be briefly discussed, as most of these procedures, local and global ones, are further discussed in the following sections (both implementation and cost in terms of number of assessments).

V.1.1 Theoretical aspects

V.1.1.1 The linearity case

In this section, the tested hypothesis is to know whether the output of our considered model, Y , can be written as a linear approximation, as follows

$$Y^k = \beta_0 + \sum_{i=1}^{n_X} \beta_i \times X_i^k + \varepsilon^k = \hat{Y}^k + \varepsilon^k.$$

EQUATION V.1: Linear model definition

In this equation, β_i are the linear regression coefficients, n_X is the dimension of the input variables, k is the index of the considered event in the complete set of data (of size n_S). In the second equality, \hat{Y} is the estimation and ε is the regression residual of the k-Th output when using the linear model. In order to study this case, few numerical expressions can be computed:

- The linear correlation coefficient: also named *Pearson* coefficient, it is computed as

$$\rho_i = \rho(X_i, Y) = \frac{\text{Cov}(X_i, Y)}{\sqrt{\text{Var}(X_i) \text{Var}(Y)}} \quad \forall i = 1, \dots, n_X$$

- The standard regression coefficient: also named *SRC* coefficient, it is computed as

$$SRC_i = SRC(X_i, Y) = \beta_i \sqrt{\frac{\text{Var}(X_i)}{\text{Var}(Y)}} \quad \forall i = 1, \dots, n_X.$$

In the case where the various inputs are independent, it is possible to link two of these coefficients by the following relation: $SRC_i = \rho_i$. Assuming a pure linear model, meaning that Equation V.1 can be written $Y = \hat{Y}$ (or equivalently $\varepsilon = 0$), there is then a closing relation, as follows:

$$\sum_{i=1}^{n_X} SRC_i^2 = 1$$

- The partial correlation coefficient: also named *PCC* coefficient. It is defined to quantify the unique sensitivity of the output to X_i that cannot be explained in terms of the relations of these variables with any other variables (*i.e.* considering that the other variables are constant). It is so defined as

$$PCC_i = \rho_{Y, X_i, \mathbf{X}_{\sim i}} = \frac{\rho_{Y, X_i} - \rho_{Y, \mathbf{X}_{\sim i}} \rho_{X_i, \mathbf{X}_{\sim i}}}{\sqrt{1 - \rho_{Y, \mathbf{X}_{\sim i}}^2} \sqrt{1 - \rho_{X_i, \mathbf{X}_{\sim i}}^2}}$$

where $\mathbf{X}_{\sim i}$ is the vector of input where the i -Th component has been taken out.

The SRC and PCC coefficients are not equal one to another but they can both be used to sort the inputs according to their impact on the output, giving the same ranking even though their values would be different. The validity of the assumption made to consider the model as linear can also be tested by computing "quality criteria". There are few of them available in Uranie (R^2 , R_{adj}^2 and Q^2) which have already been discussed in Section IV.1.1.

V.1.1.2 The monotone case

The linearity is a very strong hypothesis, which is rarely correct when dealing with real problems. In order to circumvent this hypothesis, it is possible to use the ranks (instead of the values) in order to test only the monotony of the output with respect to the inputs. Each simulation (whose index k goes from 1 to n_S) is ranked according to a variable, 1 being attributed to the simulation with the lowest value, while n_S is allocated to the largest one (or the other way around, this should not change the results). It is then possible, using these ranks instead of the values, to redefine all the previously-discussed expressions:

- The correlation coefficient: using the ranks, it is defined like the *Pearson* coefficient and is called *Spearman* coefficient

$$\rho_i^S = \rho^S(X_i, Y) = \rho(R_{X_i}, R_Y) \forall i = 1, \dots, n_X$$

- The standardised regression rank coefficient: using the rank, it is defined like the SRC and is called the *SRRC* coefficient

$$SRRC_i = SRRC(X_i, Y) = SRC(R_{X_i}, R_Y) \forall i = 1, \dots, n_X$$

- The partial rank correlation coefficient: using the rank, it is defined like the PCC and is called the *PRCC* coefficient

$$PRCC_i = PRCC(X_i, Y) = PCC(R_{X_i}, R_Y) \forall i = 1, \dots, n_X$$

The quality criteria discussed previously in the context of values study, can as well be computed with the ranks. These quality criteria (defined in Section IV.1.1) remain indeed valid to judge the validity of the monotony hypothesis once the same replacement is performed.

V.1.1.3 No hypothesis on the model

When no hypothesis is made on the relation between the inputs and the outputs, one can try a more general approach. If we can consider that the inputs are independent one to another, it is possible to study how the output variance diminish when fixing X_i to a certain value x_i^* . This variance denoted by

$$\text{Var}(Y|X_i = x_i^*)$$

is called the conditional variance and depends on the chosen value of X_i . In order to study this dependence, one should consider $\text{Var}(Y|X_i)$, the conditional variance over all possible x_i^* value. It is a random variable and, as such, one can define the expectation of this quantity as $E(\text{Var}(Y|X_i))$. The value of this newly defined random variable ($\text{Var}(Y|X_i)$) is as small as the impact of X_i on the output variance is large.

From there it is possible to use the theorem of the total variance which states, under the assumption of having X_i and Y two jointly distributed (discrete or continuous) random variables, that $\text{Var}(Y) = \text{Var}(E(Y|X_i)) + E(\text{Var}(Y|X_i))$. It becomes clear that the variance of the conditional expectation can be a good estimator of the sensitivity of the output to the specific input X_i . It is then more common and practical to refer to a normalised index in order to define this sensitivity, which is done by writing

$$S_i = \frac{\text{Var}(E(Y|X_i))}{\text{Var}(Y)}.$$

EQUATION V.2: First order sensitivity index

This normalised index is often called the *first order sensitivity index* and is, in the specific case of a purely linear theory, equal to the corresponding SRC coefficient.

If the previously defined index is the *first order*, then it means that there are higher-order indices that one might be able to compute. This index which does indeed describe the impact of the input X_i on the output, does not take into account the possible interaction between inputs. It can then be completed with the crossed impact of this particular input with any other X_j (for $i \neq j$ and $j \in [1, n_X]$). This *second order index* S_{ij} does not take into account the crossed impact of inputs i and j with another one, named for instance k (for $i \neq j \neq k$ and $k \in [1, n_X]$)... This shows the necessity to consider the interaction between all the inputs, even though they are not correlated, leading to a set of $2^{n_X} - 1$ indices to compute. A complete estimation of all these coefficients is possible and would lead to a perfect break down of the output variance, which has been proposed by many authors in the literature and is referred to with many names, such as functional decomposition, ANOVA method (*ANalysis Of VAriance*), HDMR (*High-Dimensional Model Representation*), Sobol's decomposition, Hoeffding's decomposition... This decomposition assumes that **the input factors are independent**.

In order to simplify this, it is possible to estimate, for an input X_i , the *total order sensitivity index* S_{T_i} defined as the sum over all the sensitivity indices involving the input variable under study [37]:

$$S_{T_i} = \sum_{k \in \#i} S_k = 1 - S_{\bar{i}},$$

EQUATION V.3: Total order sensitivity index

where $\#i$ and \bar{i} represents respectively the group of indices that contains and that does not contain the i index. There are few ways to check but also interpret the value of the Sobol sensitivity indices:

- $\sum S_i \leq 1$: should always be true.

- $\sum S_i = 1 = \sum S_{T_i}$: the model is purely additive, or in other words, there are no interaction between the inputs and $S_i = S_{T_i} \forall i = 1, \dots, n_X$.
- $1 - \sum S_i$ is an indicator of the presence of interactions.
- $S_{T_i} - S_i$ is a direct estimate of the interaction of the i-Th input with all the other factors.

In Uranie, there are several methods implemented to get both the first order and the total order index of sensitivity and they will be discussed in the upcoming sections.

V.1.1.4 Limitation of the Sobol indices

The decomposition discussed in Section V.1.1.3 and the resulting interpretation in terms of first and total order Sobol indices both suffer from theoretical limitation that are discussed within this section. There are different methods and indices that are said to overcome part of all of the listed issues. This part introduces the limitation and gives an insight of the way to overcome them.

There are few main concerns when dealing with sensitivity indices within the Sobol framework:

- there are two indices for every input variables.
 - On the one hand, first order indices might be obtained but because of their definition their value can be small or even null even though their real impact should not be discarded.
 - On the other hand, total order indices are very useful to complete the picture provided by the first order ones but they are usually quite costly to obtain and not so many methods can allow to reach them.
- the input variables must be independent.
 - It means input variables must be independant one to another.
 - It implies, to remain simple, that the inputs and outputs should remain scalars and if not, they should be no correlation between their elements.

Concerning the first observation, the sub-item is right disreagarding the situation: because of its definition based on the evolution of the mean of the conditional output, an input can be discarded even though it has a strong impact as long as the average impact is constant. As soon as one gets the total index (if possible), it completes the picture but having two indices can still be confusing (to understand their meaning). A way to overcome this would be to get another metric providing only one index per variable. The litterature is snowed under with new index definitions in order to overcome these limitations: based on different research field and on various mathematical techniques, one can find such names as HSIC [76], Shapley's values [72], Johnson's relative weights [48]...

For the second observation, changing the strategy is also a solution. One possibility is to gather the dependent inputs into a group and only considerer and compare the resulting groups. There will be no hierarchy information within every groups, but this technique will work only if all variables are not dependent. If not, then, rather than relying on a variance decomposition-based technique, one should use other indices once again to overcome this issue. Some methods are discussed later on, in dedicated part, while the rest of this section will briefly introduce the Shapley's values.

The Shapley analysis has originated from the game theory, when considering the case of a coalitional game, *i.e.* a couple (p, c) where

- p is the number of players;
- c is a function from $[1 : p] \rightarrow \mathbb{R}$ so that $c(\emptyset) = 0$ and $\forall A \subset B, c(A) \leq c(B)$. It is called the characteristic function.

The function c has the following meaning: if u is a coalition of players, then $c(u)$, called the worth of coalition u , describes the total expected sum of payoffs the members of u can obtain through cooperation. The Shapley's method is one way to distribute the total gains to the players, assuming that they all collaborate. It is a "fair" distribution in the sense that it is the only distribution with certain desirable properties listed below. According to the Shapley value, the amount that player i will receive, in a coalitional game (p, c) , is

$$\varphi_i := \frac{1}{p} \sum_{u \subset -i} \binom{p-1}{|u|}^{-1} (c(u \cup \{i\}) - c(u))$$

The main properties of the Shapley values are the following ones:

- efficiency: $\sum_{i=1}^p \varphi_i = c([1 : p])$
- symmetry: if i and j are two equivalent players, meaning $c(u \cup \{i\}) = c(u \cup \{j\})$, $\forall u \subset [1 : p] \setminus \{i, j\}$, then $\varphi_i = \varphi_j$
- additivity: when two coalition games (p, c) and (p, d) are combined, it defines a new game $(p, c + d)$ where $(c + d)(u) = c(u) + d(u)$, then $\varphi_i(c + d) = \varphi_i(c) + \varphi_i(d)$.
- nullity: $\varphi_i = 0$ for a null player. A player i is null if $c(u \cup \{i\}) = c(u)$, $\forall u \subset [1 : p] \setminus \{i\}$

The concept of Shapley value has been extensively used in finance for some times but has only recently been brought up in the uncertainty community, one can indeed define the Shapley value for a given input i , as done in Owen [72]:

$$\eta_i := \frac{1}{p \text{Var}(Y)} \sum_{u \subset -i} \binom{p-1}{|u|}^{-1} (V_{u \cup \{i\}} - V_u),$$

where $-i$ is the set $[1 : p] \setminus \{i\}$. Based on this definition, Shapley values have been exhibited as proper sensitivity indices in [46] when the inputs are dependent. There is indeed only one value for each input variable, this value always lies in $[0, 1]$ and their sum equals to one, once all input variables are considered (even with correlation).

In the case where $X \sim \mathcal{N}(\mu, \Gamma_{xx})$, assuming $\mu = 0$, for the sake of simplicity, without genericity loss, one can rewrite the sensitivity indices, as they can be calculated explicitly. The Sobol indices, for instance, can be expressed with expectations of conditional variances [71], as done below:

$$S_u := \frac{(-1)^{|u|}}{\text{Var}(Y)} \sum_{v \subset u} (-1)^{|v|+1} \mathbb{E}(\text{Var}(Y|X_u)), \forall u \neq \emptyset,$$

which leads to a new expression for the i -Th Shapley value:

$$\eta_i := \frac{1}{p \text{Var}(Y)} \sum_{u \subset -i} \binom{p-1}{|u|}^{-1} (\mathbb{E}(\text{Var}(Y|X_u)) - \mathbb{E}(\text{Var}(Y|X_{u \cup \{i\}})))$$

Using the Gaussian framework, one can express the conditional variance as shown here:

$$\text{Var}(Y|X_u) = \text{Var}(\beta_{-u}^T X_{-u} | X_u) = \beta_{-u}^T (\Gamma_{-u, -u} - \Gamma_{-u, u} \Gamma_{u, u}^{-1} \Gamma_{u, -u}) \beta_{-u}$$

This expression is constant for a given subset u , so it is equal to its expectation which provide a way to compute all Shapley values.

V.1.2 List of available methods

Methods for Sensitivity Analysis (SA) are split into two types:

- local: variations around a nominal value,
- global: variations in all the domain.

1. **Finite differences** (local method):

It consists in estimating the partial derivatives around a nominal value for each input parameters (see Section V.2).

2. **Values Regression** method (linearity):

It performs a sensitivity analysis based on the coefficients of a normalised linear regression (see Section V.3).

3. **Ranks Regression** method (monotony):

Here, the analysis is based on the coefficients of a normalised rank regression (see Section V.3).

4. **Morris'** screening method:

It consists in ordering the input variables according to their influence on the output variables. This method should be used for input ranking. Despite the low computational cost encountered, the obtained information is insufficient to get a proper quantitative estimation of the impact of the input variable on the output under consideration (see Section V.4).

5. **Sobol** method:

This method produces numerical values for the Sobol indices. However, it requires a high numerical cost as numerous code assessments are needed (see Section V.5).

It is based on the so-called *Saltelli & Tarantola* method, to compute the first and total order indices, using different algorithms.

6. **FAST** method:

It computes Sobol's first order indices from Fourier coefficients, using a sample on a periodic curve with different frequencies for each input variables (see Section V.6).

7. **RBD** method:

It computes Sobol's first order indices from Fourier coefficients, using a sample on a periodic curve with a unique frequency (see Section V.6).

8. **Johnson's relative weights** method:

It computes the relative weights which are aimed to be a good approximation of the Shapley's values, but whose main advantage is to be a lot quicker to estimate. This method is limited to linear cases (see Section V.7).

V.2 The finite differences method

V.2.1 General presentation of finite difference sensitivity indices

The finite differences method is among the simplest one. The resulting sensitivity index of an input variable X_i with respect to an output $Y = f(\{X_i\}_{i \in [1, n_X]})$ is an estimation of the derivative of f versus X_i , $\delta f / \delta X_i$, around a nominal

value X_i^{nom} . In this implementation of the method, the estimation is obtained by applying an OAT design-of-experiments (*One-At-a-Time*) to the studied model. For each input's nominal value, we define a range ΔX_i . The resulting estimate of the partial derivative around the nominal value is then given by

$$\frac{\partial f}{\partial X_i} = \frac{f(X_i^{\text{nom}} + \Delta X_i) - f(X_i^{\text{nom}} - \Delta X_i)}{2 \times \Delta X_i}$$

V.3 The regression method

V.3.1 General presentation of regression's coefficients

The estimation is done by estimating the correlation matrix of the output under study with the different inputs, leading to a matrix $A(n_X + 1, n_X + 1)$ which can be based on values (for SRC and PCC) or on ranks (for SRRC and PRCC). Once this matrix is estimated, it is inverted and the requested coefficients are estimated using the following relations (established in Ref [38] and given here only for illustration purpose):

- $\left(\frac{A^{-1}(n_X + 1, i)}{A^{-1}(n_X + 1, n_X + 1)} \right)^2$ for standard regression coefficients
- $\left(\frac{A^{-1}(n_X + 1, i)}{\sqrt{A^{-1}(n_X + 1, n_X + 1) \times A^{-1}(i, i)}} \right)^2$ for partial regression coefficients

where i and $n_X + 1$ being respectively the number of the input and output under study in the correlation matrix. An important factor for the quality of the regression coefficients is the quality of the model which can be asserted with the value of the R^2 and the R_{adj}^2 factors defined in Equation IV.1 and whose computation is performed as followed:

- $R^2 = 1 - \frac{1}{A^{-1}(n_X + 1, n_X + 1)}$
- $R_{\text{adj}}^2 = 1 - \frac{(1 - R^2) \times n_S}{(n_S - n_X)}$

It can be considered that R^2 and R_{adj}^2 must be superior to 0.7-0.8 in order to use the regression coefficients. However, these values are not guaranteed-threshold, one should be careful not to only rely on them to state that the underlying hypothesis is correct or not.

V.3.2 Getting a confidence-interval estimation

When considering SRC, one can rely on the equality introduced in Section V.1.1.1: $\text{SRC}_i = \rho_i$, where ρ_i is the Pearson coefficient between the output and the i -Th input. This is interesting, as uncertainty on the estimation of a correlation coefficient can be computed from Fisher's z-transformation [39] under certain hypothesis. Given a certain sample of size N , the empirical estimation $\hat{\rho}_N$ of a true correlation ρ between two normal-distributed variables (independent and identically distributed) can be transformed into \hat{Z}_N following this recipe:

$$\hat{Z}_N = \frac{1}{2} \ln\left(\frac{1 + \hat{\rho}_N}{1 - \hat{\rho}_N}\right) = \text{arctanh}(\hat{\rho}_N)$$

The nice property of this newly-defined variable is that it follows asymptotically a normal distribution of mean $\mu_{\hat{Z}_N} = \frac{1}{2} \ln\left(\frac{1 + \rho}{1 - \rho}\right)$ and standard deviation $\sigma_{\hat{Z}_N} = \frac{1}{\sqrt{N-3}}$. It is particularly appealing to notice that the standard deviation is

independent of the correlation value itself, and only depends on the number of points provided in the initial sample. From there, in order to get a 95% confidence level on the correlation coefficient, one can start from this equation

$$\hat{Z}_N - \frac{1.96}{\sqrt{N-3}} \leq \frac{1}{2} \ln\left(\frac{1+\rho}{1-\rho}\right) \leq \hat{Z}_N + \frac{1.96}{\sqrt{N-3}}$$

and invert it to get the 95% confidence interval on the correlation coefficient itself, defined as

$$\tanh\left(\hat{Z}_N - \frac{1.96}{\sqrt{N-3}}\right) \leq \rho \leq \tanh\left(\hat{Z}_N + \frac{1.96}{\sqrt{N-3}}\right) \quad (\text{V.1})$$

This procedure would be fine if the quantity of interest was the correlation coefficient, but in our case, we're interested in the square value of this coefficient. To extrapolate the confidence interval on the correlation coefficient into a proper confidence interval on its squared value, we draw a large sample (of size N_{Gen}) of $\{\hat{Z}_N^i\}_{i \in [1, N_{Gen}]}$, following the expected asymptotic behaviour of the Fisher's z-transformed variable but using the estimated value $\hat{\rho}_N$ instead of the true one ρ : $\mathcal{N}\left(\frac{1}{2} \ln\left(\frac{1+\hat{\rho}_N}{1-\hat{\rho}_N}\right), \frac{1}{\sqrt{N-3}}\right)$. This set of points is transformed into new coefficients $\{\hat{\rho}_N^i\} = \{\tanh(\hat{Z}_N^i)\}_{i \in [1, N_{Gen}]}$ which are squared in order to get a set of N_{Gen} squared-correlation coefficients. From this set, the 2.5% and 97.5% quantile are estimated leading to a resulting 95% confidence interval on the squared-value of the estimated correlation coefficient $\hat{\rho}_N^2$ (and thus on its Sobol interpretation in the linear case).

This procedure has been tested using an linear analytic model (for which it is possible to estimate the expected SRC coefficients) with normal-distributed independent and identically distributed inputs variables. Ten thousand design-of-experiments were generated and the theoretical Sobol indices were included in the estimated confidence interval in 95% of the cases. Running the same protocole with uniform distributions instead of normal ones, the theoretical Sobol indices were in the estimated confidence interval between 95% and 98% of the cases. This illustrate the fact that the resulting confidence interval can be considered exact only if the hypothesis stated above are respected. If not, it anyways provides an interesting insight on the way the estimation converges, without being a quantifiable range.

Finally, the procedure described above relies on the Pearson coefficient, to get an estimation of a confidence interval for Sobol indices in linear model. The exact same procedure can be followed using the Spearman correlation coefficient, which leads then to an estimation of a confidence interval for Sobol indices in monotonic model.

V.4 The Morris screening method

V.4.1 Principle of the Morris' method

The Morris method [14] is an effective screening procedure that robustifies a bit the OAT protocol (*One-factor-At-a-Time*). Instead of varying every input parameters only once (leading then to a minimum of $n_X + 1$ assessments of the code/function, with an OAT technique), the Morris method repeats this OAT principle r times (practically, it is between 5 and 10 times, each time being called a *trajectory* or a *replica*), with a randomly chosen starting point (in the input parameters space). In order to do so, it computes Elementary effects (later on called *EE*), defined as

$$EE_i^t = EE_i(\mathbf{X}^t) = \frac{y(X_1^t, \dots, X_i^t + \Delta_i^t, \dots, X_{n_X}^t) - y(X_1^t, \dots, X_i^t, \dots, X_{n_X}^t)}{\Delta_i^t},$$

where Δ^t is the chosen variation in the trajectory t . This variation can be set by the user, but the default (recommended, because it is supposed to be optimal [34]) value is $\Delta = \frac{p}{2(p-1)}$, knowing the evolution range of the considered input and the chosen level p that describes in how many interval, the range should be split. The resulting cost (in terms of assessment number) is then $r(n_X + 1)$. This method is schematised in Figure V.1 for a problem with three inputs. The hyper-volume is normalised and transformed into an unit hyper-cube. The resulting volume is discretised with the requested level and two trajectories are drawn for different values of the elementary variation.

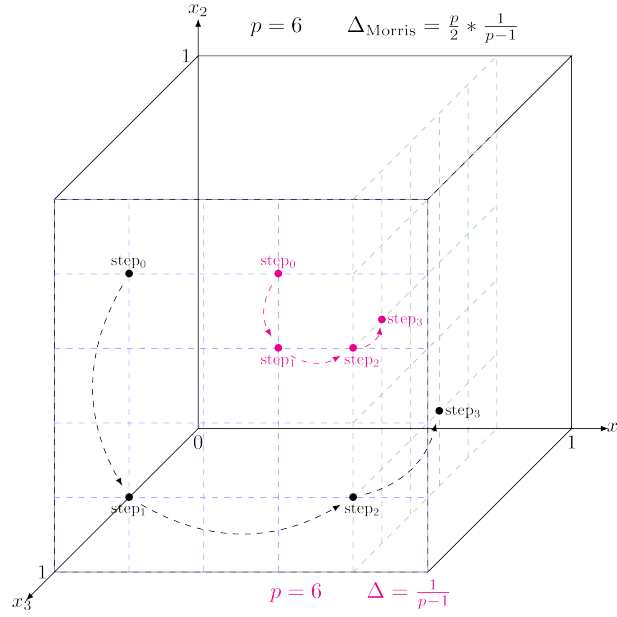


Figure V.1: Schematic view of two trajectories drawn randomly in the discretised hyper-volume (with $p=6$) for two different values of the elementary variation (the optimal one in black and the smallest one in pink, as detailed on the figure itself).

With the repetition of this procedure r times, it is possible to compute basic statistics on the elementary effects computed for every input parameter, as

$$\mu_i = \frac{1}{r} \sum_{t=1}^r EE_i^t, \mu_i^* = \frac{1}{r} \sum_{t=1}^r |EE_i^t| \text{ and } \sigma_i^2 = \frac{1}{r-1} \sum_{t=1}^r (EE_i^t - \mu_i)^2.$$

The variable μ_i and σ_i represents respectively the mean and standard deviation of the elementary effects of the i -Th input parameters. In the case where the model is not monotonic some EE_i^t may cancel each other out, resulting in a low μ_i value even for an important factor. For that reason, a revised version called μ_i^* has been created and defined as the mean of the absolute values of the EE_i^t [35].

The results are usually visualised in the (μ^*, σ) plane, allowing to sort its inputs in the following categories:

- factors that have negligible effects on the output: both μ^* and σ are small.
- factors that have linear effects, without interaction with other inputs: μ^* is large (all variations have an impact) but σ is small (the impact is the same independently of the starting point).
- factors that have non-linear effects and/or interaction with other inputs: both μ^* and σ are large.



Warning

The optimal value of $\Delta = \frac{p}{2(p-1)}$ given previously might be a dangerous choice in very few cases. When the evolution of the output as a function of one input is $T/2$ -periodic (when T is the total range of the input under consideration), the "optimal" elementary variation will lead to insensitive trajectories. In this precise cases, one might want to change elementary variation.

V.5 The Sobol method

V.5.1 Sobol's sensitivity indices

The Sobol method is a Monte-Carlo based estimation that provides the first and total order sensitivity indices at the cost of requiring a total of $n_S(n_X + 2)$ code assessments. In order to produce these results, a design-of-experiments must be produced, whose size (n_S) has to be precised by the user. Instead of generating one design-of-experiments with the requested properties, the program generates actually twice as many data points, split in two different matrices that one could call $M(n_S, n_X)$ and $N(n_S, n_X)$ (both matrices are different and independent random samplings). The method is detailed in the following paragraph, separately for the first and total order indices, but a schematic view can be found at the end in Equation V.4.

The first step is to compute the first order sensitivity index, whose definition has been given in Equation V.2. This estimation is based on the measurement of the numerator, $\text{Var}(E(Y|X_i))$, which can be written $E(E(Y|X_i)^2) - E(E(Y|X_i))^2$ from the definition of variance. Since the second part of previous formula is equivalent to the output expectation, the needed inputs to get the first order indices are: $E(E(Y|X_i)^2)$, $\text{Var}(Y)$ and $E(Y)$. The method discussed in the following paragraphs and illustrated by Equation V.4 is called the *pick-and-freeze* method. The matrix M is passed to the code and n_S assessment are done to get a vector of outputs (this is shown by the first line of Equation V.4). From this, the output properties can easily be estimated, leaving the first term only to be measured. In order to estimate it for the i -Th input variable, one needs to have two different random sampling with the i -Th component (to satisfy the conditional character of the expectation to be squared). This is where the second matrix N is used: its i -Th column is replaced by the M 's one (*peek*), creating a new N_i matrix. The last term to be estimated can be computed as

$$E(E(Y|X_i)^2) = \frac{1}{n_S} \sum_{k=1}^{n_S} y_k^M y_k^{N_i}$$

(the mathematical development leading to this formula can be found in many references). This step, whose total cost is $n_S(n_X + 1)$ code assessments, is shown by the second line and the right-part of the third line in Equation V.4 (following the arrows).

Finally the total order indices are computed starting from the second part of Equation V.3. This second part can also be written as

$$S_{T_i} = 1 - \frac{\text{Var}(E(Y|X_{\bar{i}}))}{\text{Var}(Y)}$$

which looks very much alike Equation V.2 used to compute the first order but instead of a condition on having i known (*frozen*), it is the exact opposite: the condition is to freeze everything but i . This is easily doable as this is the only difference between the N and N_i matrices. Following the same recipe as for the first order, only the $E(E(Y|X_{\bar{i}})^2)$ has to be computed with the following formula:

$$E(E(Y|X_{\bar{i}})^2) = \frac{1}{n_S} \sum_{k=1}^{n_S} y_k^N y_k^{N_i}.$$

The compulsory step is so to pass the N matrix to the code, leading to n_S code assessments, as shown in Equation V.4 by the left part of the third line.

$$\begin{array}{ccc}
M = \begin{pmatrix} m_{1,1} & \dots & m_{1,i} & \dots & m_{1,n_X} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ m_{j,1} & \dots & m_{j,i} & \dots & m_{j,n_X} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ m_{n_S,1} & \dots & m_{n_S,i} & \dots & m_{n_S,n_X} \end{pmatrix} & \xrightarrow{-n_S \text{ assessments}} & \begin{pmatrix} f(m_{1,1}, \dots, m_{1,i}, \dots, m_{1,n_X}) \\ \vdots \\ f(m_{j,1}, \dots, m_{j,i}, \dots, m_{j,n_X}) \\ \vdots \\ f(m_{n_S,1}, \dots, m_{n_S,i}, \dots, m_{n_S,n_X}) \end{pmatrix} \\
\\
N = \begin{pmatrix} n_{1,1} & \dots & n_{1,i} & \dots & n_{1,n_X} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ n_{j,1} & \dots & n_{j,i} & \dots & n_{j,n_X} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ n_{n_S,1} & \dots & n_{n_S,i} & \dots & n_{n_S,n_X} \end{pmatrix} & \xrightarrow{M's i^{th} \text{ input instead of } N's} & N_i = \begin{pmatrix} n_{1,1} & \dots & n_{1,i-1} & \mathbf{m_{1,i}} & n_{1,i+1} & \dots & n_{1,n_X} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ n_{j,1} & \dots & n_{j,i-1} & \mathbf{m_{j,i}} & n_{j,i+1} & \dots & n_{j,n_X} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ n_{n_S,1} & \dots & n_{n_S,i-1} & \mathbf{m_{n_S,i}} & n_{n_S,i+1} & \dots & n_{n_S,n_X} \end{pmatrix} \\
\downarrow n_S \text{ assessments (for } S_{T_i} \text{ estimation)} & & \downarrow n_S \times n_X \text{ assessments (for } i \in [1, n_X]) \\
\begin{pmatrix} f(n_{1,1}, \dots, n_{1,i}, \dots, n_{1,n_X}) \\ \vdots \\ f(n_{j,1}, \dots, n_{j,i}, \dots, n_{j,n_X}) \\ \vdots \\ f(n_{n_S,1}, \dots, n_{n_S,i}, \dots, n_{n_S,n_X}) \end{pmatrix} & & \begin{pmatrix} f(n_{1,1}, \dots, n_{1,i-1}, \mathbf{m_{1,i}}, n_{1,i+1}, \dots, n_{1,n_X}) \\ \vdots \\ f(n_{j,1}, \dots, n_{j,i-1}, \mathbf{m_{j,i}}, n_{j,i+1}, \dots, n_{j,n_X}) \\ \vdots \\ f(n_{n_S,1}, \dots, n_{n_S,i-1}, \mathbf{m_{n_S,i}}, n_{n_S,i+1}, \dots, n_{n_S,n_X}) \end{pmatrix}
\end{array}$$

EQUATION V.4: Schematic view of the Sobol method

This method is, in Uranie, said to be "à la Saltelli" in contrast with the other implementation (previously used as default) which is said to be "à la Sobol". The difference between the two being the number of assessment used to get a certain precision: for the same results, the implementation "à la Sobol" was requesting $n_S(2n_X + 2)$ and was offering more numerical results as five algorithms were used. The new implementation "à la Saltelli" requests only $n_S(n_X + 2)$ estimation but only three algorithms are run. This is summarised as follow where the bold name is the default stored in `--first--` and `--total--`:

a la Saltelli First order: Saltelli02 [40], Sumo10 [41], **Martinez11** [44]

Total order: Homma96 [37], Sumo10 [41], **Martinez11** [44]

a la Sobol First order: **Sobol93** [45], Saltelli02 [40], Jansen99 [73], Sumo10 [41], Martinez11 [44]

Total order: Homma96 [37], Saltelli02 [40], **Jansen99** [73], Sumo10 [41], Martinez11 [44]



Tip

The Martinez11 algorithm is the recommended one, as it provides an estimation of the 95% confidence interval for every coefficient determined.

V.6 Fourier-based methods

V.6.1 Introducing the method

V.6.1.1 The FAST method

The *Fourier Amplitude Sensitivity Test* (FAST) [74, 75] is a procedure that provides a way to estimate the expected value and variance of the output variable of a model, along with the contribution of the input factors to this variance. An advantage of it, is that the evaluation of sensitivity can be carried out independently for each factor using just a set of runs because all the terms in a Fourier expansion are mutually orthogonal. The main idea behind this procedure is to transform the n_X -dimensional integration into a single-dimension one, by using the transformation

$$X_i = G_i(\sin(\omega_i \times s)),$$

where ideally, $\{\omega_i\}$ is a set of angular frequencies said to be incommensurate (meaning that no frequency can be obtained by linear combination of the other ones when using integer coefficients) and G_i is a transformation function chosen in order to ensure that the variable is sampled accordingly with the probability density function of X_i (meaning that they are all uniformly distributed in their respective volume definition). Given these conditions, the parametric variable s will evolve in $[-\infty, \infty]$ and the vector $(X_1(s), \dots, X_{n_X}(s))$ traces out a curve that fills the entire n_X -dimensional research volume. Practical considerations dictate that an integer rather than an incommensurate set of frequencies must be used, with few consequences: the resulting parametric curve is not longer a space-filling one, the fundamental of each input (the chosen frequency for this input) will have harmonics that interfere with one another and the parametric curve becomes periodic with a 2π -period.

When both G_i and ω_i are properly chosen, one can approximate the following relations:

$$E(Y) = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(s) ds \text{ and } \text{Var}(Y) = \frac{1}{2\pi} \int_{-\pi}^{\pi} f^2(s) ds - E^2(Y) \approx 2 \sum_{k=1}^{\infty} (A_k^2 + B_k^2),$$

EQUATION V.5: Expectation and variance of output in Fourier space

where $f(s) = f(G_1(\sin(\omega_1 s)), \dots, G_{n_X}(\sin(\omega_{n_X} s)))$ and A_k and B_k are the Fourier coefficients, defined as

$$A_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(s) \cos(ks) ds \text{ and } B_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(s) \sin(ks) ds.$$

EQUATION V.6: Fourier coefficient definition

The first order coefficient is then obtained by estimating the variance for a fundamental ω_i and its harmonics. This can be done by using the second half of Equation V.5 running over p instead of k and replacing in the index by $p \cdot \omega_i$. The important point to notice, for a real computation, is the limitation of the sum that, in the previous equation, runs up to infinity. A truncation is done by imposing a cut-off with a factor M called the **interference factor** (whose default value in Uranie is set to 6). Knowing that the exact same replacements can be done to obtain the corresponding Fourier coefficients in Equation V.6, the contribution to the output variance of a certain frequency, *i.e.* the first order sensitivity index, can be expressed as

$$S_i = \frac{\sum_{p=1}^M (A_{p \cdot \omega_i}^2 + B_{p \cdot \omega_i}^2)}{\sum_{i=1}^{n_X} \sum_{p=1}^M (A_{p \cdot \omega_i}^2 + B_{p \cdot \omega_i}^2)}.$$

Finally, the sample size n_S used to measure these coefficients should respect the relation $n_S \geq 2M \max(\{\omega_i\}) + 1$.

V.6.1.2 The RBD method

The *Random Balance Design* (RBD) [77] method selects n_S design points over a curve in the input space. The input space is explored here using the same frequency ω . However the curve is not space-filling, therefore, we take random permutations of the coordinates of such points, to generate a set of scrambled points that cover the input space. The model is then evaluated at each design point. Subsequently, the model outputs are re-ordered such that the design points are in increasing order with respect to factor X_i . The Fourier spectrum is calculated on the model output at the frequency ω and at its higher harmonics $\{\omega, 2\omega, \dots, M\omega\}$ and yields the estimate of the sensitivity index of factor X_i . The model outputs are re-ordered with respect to the other factors to obtain all the other sensitivity indices.

In practice the RBD approach selected design points can be written as:

$$X_i(s_{ij}) = G_i(\sin(\omega \cdot s_{ij})) \quad \forall i = 1, \dots, n_X \text{ and } \forall j = 1, \dots, n_S$$

where $\omega \leq (n_S - 1)/2M$ and $\{s_{i1}, \dots, s_{in_S}\}$ denotes the i -Th random permutation of the n_S points. The values of the model output $Y(s_j)$, for $j = 1, \dots, n_S$ are computed and then are reordered ($Y^R(s_j)$) in order to get the corresponding values of $X_i(s_{ij})$ ranked in increasing order. The sensitivity of Y to X_i is determined by the harmonic content of Y^R , which is quantified by its Fourier spectrum:

$$F(\omega) = \frac{1}{\pi} \sum_{j=1}^{n_S} Y^R(s_j) \exp(-\text{Im } n_X \omega s_j)$$

evaluated at ω equal to 1 and its higher harmonics (2, 3, ..., up to M equal to 6 in our case), leading to

$$\hat{V}_i = \text{Var}[E(Y|X_i)] = \sum_{l=1}^M F(\omega)|_{\omega=l}$$

This relation is used to estimate all the \hat{V}_i , for $i = 1, \dots, n_X$, by re-ordering the output to rank the i -Th input in an increasing order, which provides a complete estimation of the variance. Thanks to the use of permutations, the total cost is of the order of n_S assessments instead of the order of $n_S \times n_X$ for the FAST one.

V.6.2 Implementation of methods

In the implementation done within Uranie there are several modifiable parameters that can be considered before starting an analysis using the FAST method:

- The transformation function G_i chosen among the following list:

- Cukier: $X_i = \bar{X}_i \exp(\bar{v}_i \sin(\omega_i s))$
- SaltelliA: $X_i = 0.5 + \frac{1}{\pi} \arcsin(\sin(\omega_i s))$
- SaltelliB: $X_i = 0.5 + \frac{1}{\pi} \arcsin(\sin(\omega_i s + \phi_i))$

In this list, \bar{X}_i is the nominal value of the factor X_i , \bar{v}_i denotes the endpoints that define the estimated range of uncertainty of X_i , ϕ_i is a random phase shift taken value in $[0, 2\pi]$ and s evolves in $[-\pi/2, \pi/2]$.

- The interference factor: M can be changed as well.
- The frequencies: by providing a vector, it is possible to set a default at the frequencies' value used instead of having them determined by a specific algorithm to avoid, as best as possible, the interference.

The only common parameter changeable for both methods (and directly in the construction) is the number of samples.

V.7 The Johnson relative weight

This section introduces indices whose purpose is mainly to obtain good estimators of the Shapley's values defined in Section V.1.1.4. The underlying assumption is to state that the model can be considered linear so that the results can be considered as proper estimation of the Shapley indices (with or without correlation between the input variables).

V.7.1 Introducing the method

The idea here is very similar to the standard regression coefficients introduced in Section V.1.1.1, as one will use orthogonal transformation to represent our data, with dependent inputs. The method has been introduced by Johnson in [48] and its principle can be split into three steps:

1. transform the dependent input variables X through a linear transformation into Z so that $Z^T Z = 1_{p,p}$;
2. compute sensitivity index of Y with respect to Z ;
3. reconstruct the sensitivity index of Y with respect to the component of X .

Practically, the method proposed by [48] relies on the singular value decomposition of \mathbf{X} , written as $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ for which \mathbf{U} , contains the eigenvectors of $\mathbf{X}\mathbf{X}^T$, \mathbf{V} contains the eigenvectors of $\mathbf{X}^T \mathbf{X}$ and $\mathbf{\Sigma}$ is a diagonal matrix containing the singular values of \mathbf{X} . From there, the best-fitting orthogonal approximation of \mathbf{X} can be obtained (c.f. [47] for the demonstration) as

$$\mathbf{Z} = \mathbf{U}\mathbf{V}^T.$$

The second steps consists in regressing \mathbf{Y} onto \mathbf{Z} , which is obtained by

$$\begin{aligned}\beta^* &= (\mathbf{Z}^T \mathbf{Z})^{-1} \mathbf{Z}^T \mathbf{Y} = (\mathbf{V}\mathbf{U}^T \mathbf{U}\mathbf{V}^T)^{-1} \mathbf{V}\mathbf{U}^T \mathbf{Y} = 1_{p,p}^{-1} \mathbf{V}\mathbf{U}^T \mathbf{Y} \\ &= \mathbf{V}\mathbf{U}^T \mathbf{Y}.\end{aligned}$$

The squared elements of β^* represent the proportion of predictable variance in Y accounted for by the $(Z_i)_{i=1,\dots,p}$, but in the case where two or more original variables are highly correlated, the Z variables are not a close representation of the X ones. To take this into account, Johnson proposed to regress \mathbf{X} onto \mathbf{Z} , leading to other weights defined as:

$$\begin{aligned}\Lambda^* &= (\mathbf{Z}^T \mathbf{Z})^{-1} \mathbf{Z}^T \mathbf{X} = (\mathbf{V}\mathbf{U}^T \mathbf{U}\mathbf{V}^T)^{-1} \mathbf{V}\mathbf{U}^T \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = 1_{p,p}^{-1} \mathbf{V}\mathbf{\Sigma}\mathbf{V}^T \\ &= \mathbf{V}\mathbf{\Sigma}\mathbf{V}^T\end{aligned}$$

From there, the variance of Y explained by X_i can be written as Δ_i^2 and estimated from the following formula

$$\Delta_i^2 = \sum_{j=1}^p \lambda_{ij}^{*2} \beta_j^{*2},$$

where $\lambda_{ij}^* = (\Lambda^*)_{ij}, \forall i, j \in [1, p]^2$ and bearing in mind that

$$\text{Var}(Y) = \sum_{i=1}^p \Delta_i^2 = \sum_{i=1}^p \beta_i^{*2}.$$

The resulting sensitivity indices are written

$$\Delta_i^* = \frac{\Delta_i^2}{\text{Var}(Y)}, \quad \forall 1 \leq i \leq p.$$

Chapter VI

Dealing with optimisation issues

VI.1 Introduction

An optimisation is a complex problem, because each study has its own peculiarities and it often requires to grope one's way forward, before finding an interesting solution. Most commonly, when dealing with optimisation, there are:

- one or more objectives that one wants to minimise (or maximise).
- decision variables that have a clear influence on the objectives.
- possibly some constraints either on the decision variables, on combination of some of them, or on objectives (defining the search domain)

Knowing this, it is also compulsory to choose an optimisation algorithm, which is a crucial part of the optimisation procedure. It is possible to divide these algorithms into two different categories:

- local ones: they allow mono-criterion optimisation, with or without constraints. They are generally computationally efficient, but can not be used in parallel and tend to be trapped in local optima.
- global ones: they allow multi-objective optimisation, with or without constraints. They are suitable for problems with many local optima, but are expensive computationally. However, they are easily parallelisable.

In the rest of this section, we will discuss optimisation procedure either in single-criterion or multi-objective case. Independently of the chosen configuration the problem depends on input variables which will be gathered in a vector \mathbf{x} , called *optimisation variable* or *decision variable*. Each of these variables can be constrained, *i.e.* following simple functions (as inequality) or more complicated ones, that can be combination of several variables. The most obvious constraints are the boundaries: they define the minimum and maximum values authorised for all the inputs, defining then the *research space*.

VI.1.1 Single criterion case

In the case of a single criterion problem, the optimisation procedure is equivalent to the minimisation of a function $f(\mathbf{x})$ which is called the *cost function* (but also *objective function* depending on the literature). The optimisation leads to the determination of a minimum (that can be called *optimum*) that can either be global (there is no \mathbf{x}' in the research volume such as $f(\mathbf{x}') < f(\mathbf{x}_{\min})$) or local (same relation as before, but only in the vicinity of \mathbf{x}_{\min}).

VI.1.2 The pareto concept in a nutshell

As already discussed above, the optimisation is usually a minimisation problem of one or more objectives. We will discuss here the multi-objective case as it leads to the definition of the Pareto front and the Pareto set. The optimisation procedure can then be expressed as the minimisation of this function:

$$F(\mathbf{x}) = (f_1(\mathbf{x}); f_2(\mathbf{x}); \dots; f_n(\mathbf{x}))$$

where n is the number of objectives imposed and F is, here, the complete cost function. Unlike the single-criterion case, there might be no such thing as an overall optimum since it is usually not possible to quantify a relation between the objectives (to state which one is more important than the other). In the case where two solutions (x_1 and x_2) are possible, one can say that x_1 dominates x_2 if the former does as good as the latter for all the objectives but at least one, where it does better. The most common thing is to look for a group of solutions that are said to be *not dominated*, meaning that they dominate the rest of the solutions (that do not belong to this group) but each and every one of them can not dominate the others. No-one can state whether one solution in this group is better than any other in the group (unless an external constraint or preference is imposed, usually with hindsight). This definition goes along with the concept of trade off. The group of not-dominated solutions is then called the *Pareto set* and its graphical representation in the objective space is called the *Pareto front*. Finally, in the objective space, one can represent the *ideal point* and *Nadir point* which are respectively the minimum and maximum of all objective when considered separately in the Pareto front.

Figure VI.1 shows a very simple example of a pure analytic model with two objectives (the blue and red curves on the left panel) depending only on one variable. In this simple case, the Pareto set is shown in pink, as the area in between both criterion's minimum. Now looking in the criterion's space (right panel of Figure VI.1), all the solutions are shown in black and the corresponding Pareto front is, once more, depicted in pink. Both the ideal and Nadir points are represented for illustration purpose.

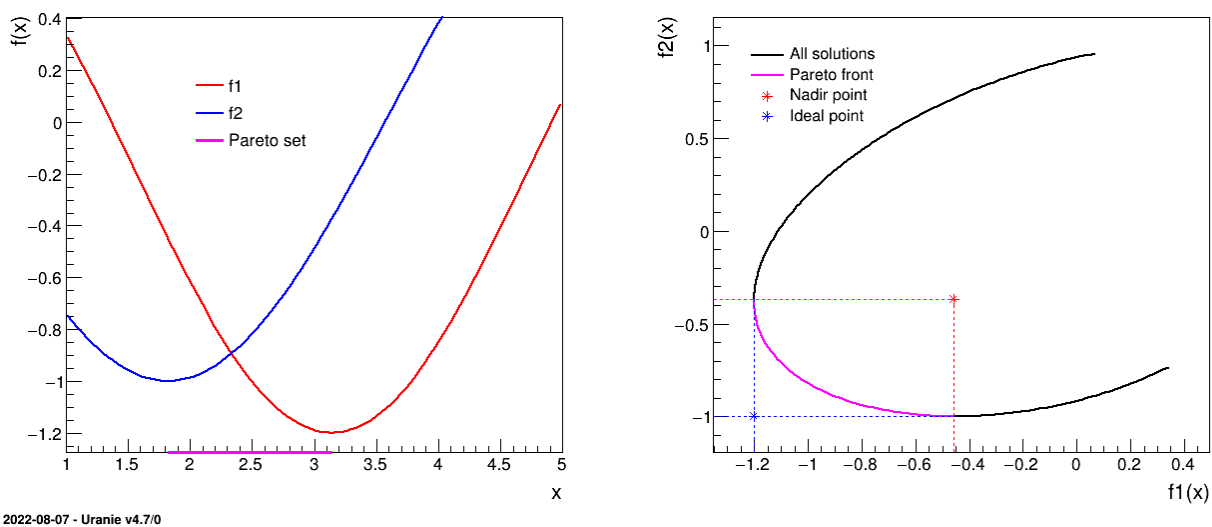


Figure VI.1: Naive example of an imaginary optimisation case relying on two objectives that only depend on a single input variable.

VI.2 Multicriteria optimisation

VI.2.1 Hitchhiker's guide to genetic algorithms

A genetic algorithm is based on the way the genes are organised in a living being and the way they might evolve through time and through the future generations.

VI.2.1.1 A brief recap of genetic: needed vocabulary (simplify model)

Genetic is the study of **genes** which are the places where all information concerning a being are encoded. These genes are gathered in some kind of containers named **chromosomes**. Depending on the being under consideration, the number of chromosomes will change (for instance, the human being has 23 types of chromosomes). In nature, the living beings can be split into two categories: those with only one chromosome per type (called **haploid** beings) and those with two chromosomes per type (called **diploid** beings). For the former, one precise characteristic is completely described by the corresponding gene in the chromosome. On the other hand, for the latter, the same characteristic is encoded by the two versions of the gene, called **alleles**, located on both chromosomes.

When considering a diploid being, another notion may arise: for every gene, it is possible to have twice the same allele or two different alleles on both chromosomes. When talking about this specific gene, the first case is called **homozygote** while the second is called **heterozygote**. The latter case will then leads to define the concept of **dominance**, whose existence is possible in two different forms: the **true dominance** when an investigated characteristic is only driven by the dominant allele, or the **shared dominance** when the resulting characteristic is an admixture of both alleles. In the former case, the characteristic will be defined by one allele (called the **dominant** one while the other is being called the **recessive** one).

Finally, evolution of the genotype through generations can come from two processes: the **crossing** one (when two beings are combining themselves to create a new one) and the **mutation** one (that happened spontaneously changing slightly one or several genes). The complete description of the genes in a being under study, is called **genotype**, while the complete description of its characteristics is called **phenotype**.

VI.2.1.2 Uranie implementation of the genetic algorithm

The genetic algorithm implemented in Uranie is a diploid algorithm with real-coding and true dominance. In other words, it means that every gene has two values (allele) which are represented by a double number, and the phenotype for this gene is driven either by one allele or the other (but not an admixture of both version). The implementation is done by plugging the Vizir package [78] which is developed by Gilles Arnaud. The concept of homozygote or heterozygote (that should apply at the gene level) is applied here at the being level, implying that if a candidate is flagged as homozygote, it means that all genes on a certain chromosome are rigorously identical on the second one.

Concretely the implementation is done as follows: a chromosome is a fixed-size vector of double (the size being the number of gene, *i.e.* the number of criteria). At the initialisation, the value for the gene are chosen in the research space of the corresponding criteria, following a random drawing (like a SRS one). The next step, the pairing by two to create a candidate (as the considered beings are diploid), is complex and very much alike the crossing step to create children from selected candidates. Based on the Figure VI.2, this procedure is described below, split in several steps: starting from two parents (selected candidate of the N-Th generation) the crossing recipe to get N+1-Th generation candidate is

Draw one chromosome per parent In Uranie the chromosome is not taken as it is (copied) from the N-Th generation, but instead a new chromosome is created, taking randomly allele from the first or second chromosome of the corresponding parent. Once this is done twice, we obtain two chromosomes that should be associated to

form a being. This step is present in Figure VI.2 both in the red and blue parts, whose difference is explained in the next step.

Create homozygote being In Uranie the concept of homozygote or heterozygote (that should apply to gene, c.f. Section VI.2.1.1) is applied to the considered being. There is a certain probability H_o for the being to be homozygote (the default value in Uranie being 50%). To do that, the two gathered chromosomes are scanned, and for every genes a linear sum is done between both alleles using a real coefficient randomly drawn in $[0 - \varepsilon, 1 + \varepsilon]$ (this ε can help a bit exploring the research space). The results is stored in a new vector (chromosome) which is duplicated to get a new being with two identical chromosome that we call an homozygote candidate (red part in Figure VI.2). On the contrary, if the *zygote* probability is in $(1 - H_o)$, the candidate will be called heterozygote (blue part in Figure VI.2).

Determine the phenotype The phenotype is determined by the dominant or recessive character of alleles, which is shown by the green part in Figure VI.2. The first and second steps discussed here to draw one chromosome per parent and create or not a homozygote being, do not take this information into account at all. The finalisation of the candidate consists then in doing a new random drawing for every gene, in order to determine which allele will be the dominant one. Once this is settled, the dominant one is put in the first vector (chromosome) of the candidate, for the sake of simplicity when testing the candidate. This is why in the parents and the resulting children, disregarding the candidate is homozygote and heterozygote, a capital D is found below the first chromosome (for dominant) while a capital R is written below the second one (for recessive).

Mutate genes To keep on exploring the research space, a random drawing is performed in order to know whether the resulting candidate should mutate. This step is also applied both to homozygote and heterozygote candidate, with a probability M whose default value in Uranie is 1%. If selected, a gene is chosen in both chromosomes of the candidate (not necessarily the same gene, as this choice is as well coming from random drawing) and a new randomly-drawn value is stored instead of the one already there.

Test and validate the children The newly produced candidates are then tested.

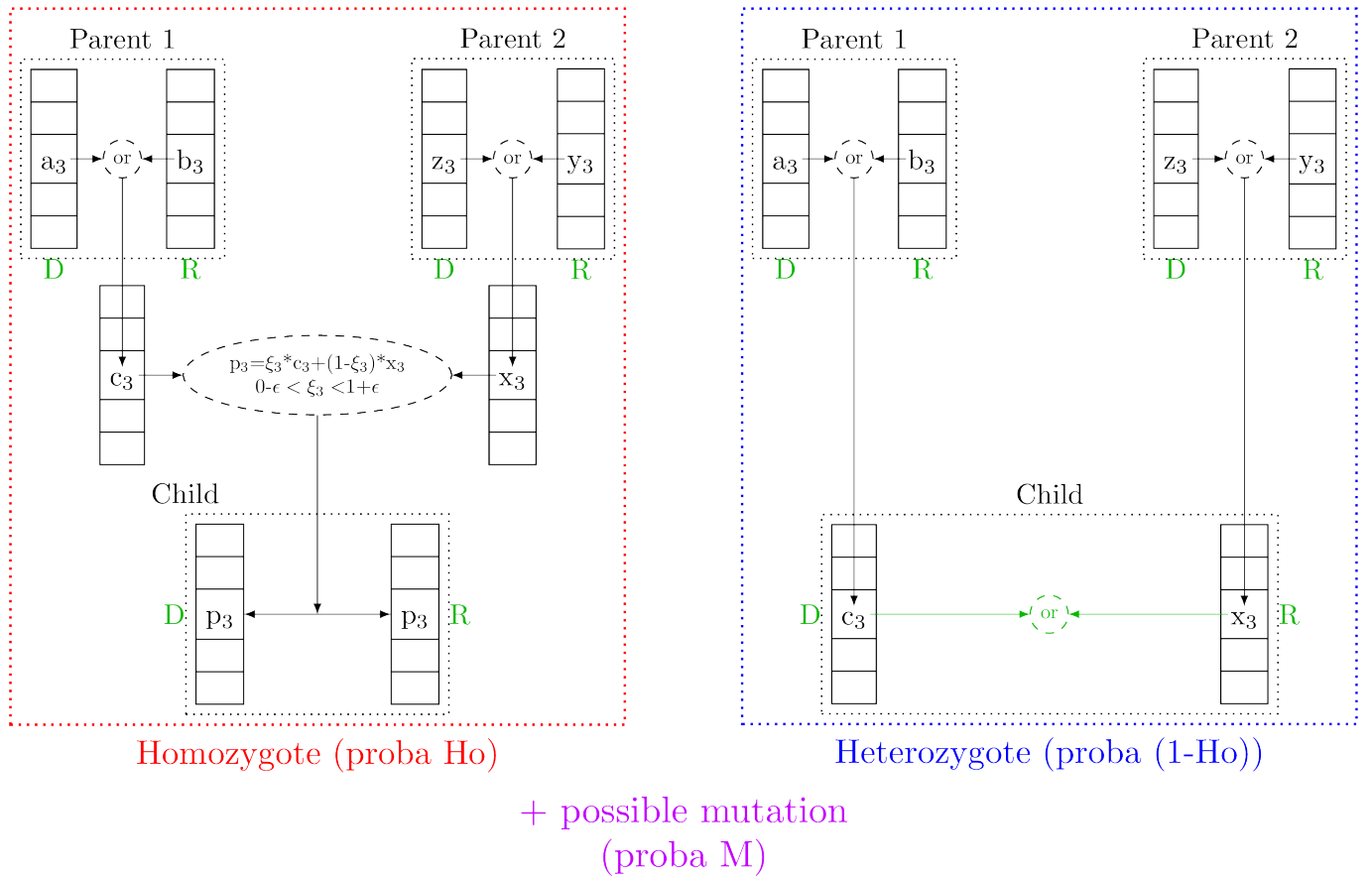


Figure VI.2: Description of the children production process in the Uranie implementation of the genetic algorithm

VI.2.2 General discussion on multi and many criteria problem.

This section introduces the discussion from [1] which is a CEA technical note. The idea is to provide access to dedicated references and provide the vocabulary, if the reader cannot either access the catalog of CEA notes or read french (otherwise Ref. [1] is the one to consult, the rest of this section can be discarded).

VI.2.2.1 Evolutionary algorithm principle.

Unlike classical optimisation algorithm whose research is basically sequential from a given starting point, an evolutionary algorithm starts from a random sample (a population) which evolves thanks to:

- selection operators: how to choose the parents, *i.e.* the best element that should be the starting point to new elements (*parental selection*) and how to remove the less interesting one (*environmental selection*).
- variation operators: how to produce a new elements from existing ones (*crossing*), how to randomly alterate some characters (*mutation*) and how to correct some characters to make a new element viable (*reparation*, optional one).

When dealing with multi-objective problems, the notion of Pareto set and front is quickly arising as one wants to obtain a pareto set, *i.e.* an ensemble of non-dominated solutions. The solutions should ideally obey different properties:

- **convergence:** be as close as possible to the "real" pareto front usually called the Pareto-optimal front;

- **coverage**: coverage the widest range possible;
- **density**: elements of the set should be reparted as evenly as possible on the obtained coverage.

Discussions about these notions can be found for instance in [10] and in [9].

VI.2.2.2 Issues with more than one objective.

Is is simple to spot that 2 things can separatly bring a poor convergence of a finite pareto set:

- the size of the requested population;
- the number of objectives to be minimised.

If the only criteria to stop an evolutionary algorithm is the fact that the ensemble of solution are non-dominated one to another, then clearly having a large number of criteria will lead to have non-comparable solutions almost out-of-box (at the initialisation level). Even with only two criteria, but a small number of requested element in the final population, putting the solution away one to another to improve the coverage will automatically reduce the convergence as the competition between elements will diminish. Even the crossing could become problematic: given a very large Pareto front, crossing very different elements might lead far away new resulting elements, meaning an inefficiency in the generation steps.

Figure VI.3 shows a dummy example of this, using our classical hollow bar example introduce in the user manual, asking only 20 elements in the final population. The classical genetic algorithm (blue dots) stops with the non-dominance criteria, so to obtain this only 100 estimations are needed in total. If one compares the way the solutions are distributed in the criteria space (the Pareto Front), the coverage, density and convergence are worse than the curve shown by the red dots. These dots are the 20 solutions proposed by the MOEAD implentation (see next section below) that decomposes the space before starting the initialisation and is not only relying on dominance to perform the different operation stated above.

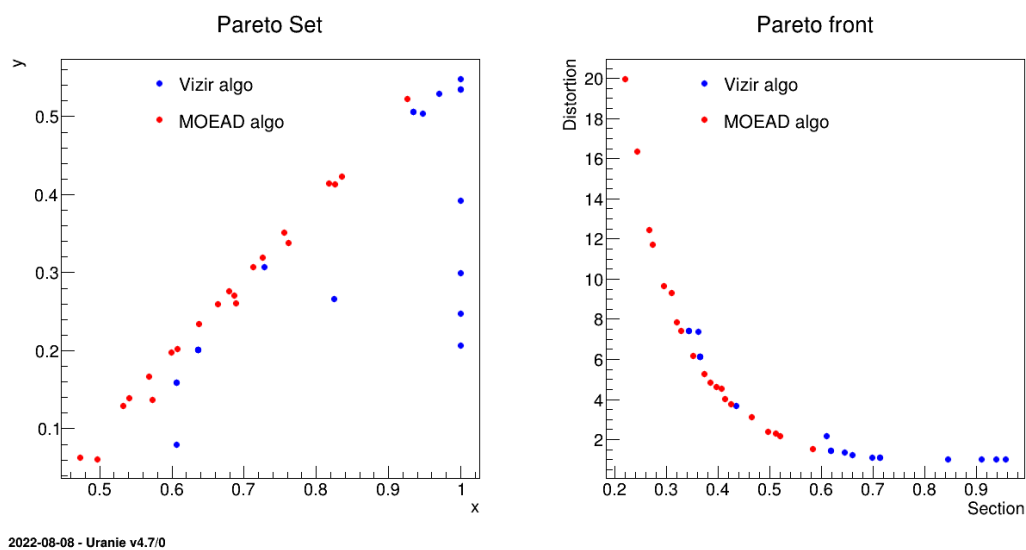


Figure VI.3: Comparison of two Pareto sets (left) and fronts (right) from vizir (blue) and MOEAD (ref) when the hollow bar case is studied with very low number of points, *i.e.* about 20 (simulating higher dimensions).

VI.2.2.3 Available implementations and references.

This section provides a simple list of algorithms and their references, that can deal with many-objective optimisation. They have been splitted in different categories, among which

- those who rely on distance or isolation notion to sort out solution which have the same rank (once pareto-dominance ranking is done);
- those who try to generalise pareto ranking to make it more discriminant;
- those who no longer use pareto rank but try to set up quality index instead;
- those who split the domain first, in order to get a better coverage at first of the ensemble of solutions.

In Uranie one would find different implementation of many-objective algorithms that enrich the possibility and capacity of the usual evolutionary algorithm, among which:

- a knee-point one [8]
- an IBEA one (*Indicator Based Evolutionary Algorithm*) [5]
- an MOEAD one (*Multi Objective Evolutionary Algorithm based on Decomposition*) [4]

Chapter VII

The Calibration module

VII.1 Brief reminder of theoretical aspects

This section presents different calibration methods that are provided to help get a correct estimation of the parameters of a model with respect to data (either from experiment or from simulation). The methods implemented in Uranie are going from the point estimation to more advanced Bayesian techniques and they mainly differ from the hypothesis that can be used.

In general, a calibration procedure will request an input datasets meaning an existing set of elements (either resulting from simulations or experiments). This ensemble (of size n) can be written as

$$\mathcal{D} = \{(\mathbf{x}^i, \mathbf{y}^i), i = 1, \dots, n\}$$

where \mathbf{x}^i is the i -Th input vector which can be written as $\mathbf{x}^i = (x_1^i \dots x_{n_x}^i)$ while \mathbf{y}^i is the i -Th output vector which can be written as $\mathbf{y}^i = (y_1^i \dots y_{n_y}^i)$. These data will be compared to model predictions, the model being a mathematical function $\mathbf{f}_\theta : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_y}$. From now on and unless otherwise specified (for distance definition for instance, see Section VII.1.1) the dimension of the output is set to 1 ($n_y = 1$) which means that the reference observations and the predictions of the model are scalars (the observation will then be written y and the prediction of the model $f_\theta(\mathbf{x})$).

On top of the input vector, already introduced previously, the model depends also on a parameter vector $\theta \in \Theta \subset \mathbb{R}^p$ which is constant but unknown. The model is deterministic, meaning that $f_\theta(\mathbf{x})$ is constant once both \mathbf{x} and θ are fixed. In the rest of this documentation, a given set of parameter value θ is called a **configuration**.

The standard hypothesis for probabilistic calibration is that the observations differ from the the predictions of the model by a certain amount which is supposed to be a random variable as

$$\varepsilon = y - f_\theta(\mathbf{x}) \tag{VII.1}$$

where ε is a random variable whose expectation is equal to 0 and which is called *residue*. This variable represents the deviation between the model prediction and the observation under investigation. It might arise from two possible origins which are not mutually exclusive:

- experimental: affecting the observations. For a given observation, it could be written $\varepsilon_{\text{obs}} = y_{\text{real}} - y$
- modelling: the chosen model f_θ is intrinsically not correct. This contribution could be written $\varepsilon_{\text{model}} = f_\theta^* - f_\theta$

As the ultimate goal is to have $y_{\text{real}} - f_\theta^* = 0$, injecting back the two contributions discussed above, this translates back to equation VII.1, only breaking down:

$$y - f_\theta = \varepsilon_{\text{obs}} + \varepsilon_{\text{model}}.$$

The rest of this section introduces two important discussions that will be referenced throughout this module:

- the distance between observations and the predictions of the models, in Section VII.1.1;
- the theoretical background and hypotheses (linear assumption, concept of *prior* and *posterior* distributions, the Bayes formulation...) in Section VII.1.2.

The former is simply the way to obtain statistic over the n samples of the reference observations when comparing them to a set of parameters and how these statistics are computed when the $n_Y \neq 1$

On top of this description, there are several predefined calibration procedures proposed in the Uranie platform:

- The minimisation, discussed in Section VII.2
- The linear Bayesian estimation, discussed in Section VII.3
- The ABC approaches, discussed in Section VII.4
- The Markov-chain Monte-Carlo sampling, discussed in Section VII.5

VII.1.1 The distance used to compare observations and model predictions

There are many ways to quantify the agreement of the observations (our references) with the predictions of the model given a provided vector of parameter θ . As a reminder, this step has to be run every time a new vector of parameter θ is under investigation which means that the code (or function) should be run n times for each new parameter vector.

Starting from the formalism introduced above, many different distance functions can be computed. Given the fact that the number of variable n_Y used to perform the calibration can be different than 1, one might also need variable weight $\{\omega_j\}_{j \in [1, n_Y]}$ that might be used to ponderate the contribution of every variable with respect to the others. Given this, here is a non-exhaustive list of distance functions:

- L1 distance function (sometimes called Manhattan distance): $d(\mathbf{y}, \mathbf{f}_\theta(\mathbf{x})) = \sum_{j=1}^{n_Y} \omega_j \times \left(\sum_{i=1}^n |\mathbf{y}_i^j - \mathbf{f}_\theta(\mathbf{x})_i^j| \right)$
- Least square distance function: $d(\mathbf{y}, \mathbf{f}_\theta(\mathbf{x})) = \sum_{j=1}^{n_Y} \omega_j \times \sqrt{\sum_{i=1}^n (\mathbf{y}_i^j - \mathbf{f}_\theta(\mathbf{x})_i^j)^2}$
- Relative least square distance function: $d(\mathbf{y}, \mathbf{f}_\theta(\mathbf{x})) = \sum_{j=1}^{n_Y} \omega_j \times \sqrt{\sum_{i=1}^n \left(\frac{\mathbf{y}_i^j - \mathbf{f}_\theta(\mathbf{x})_i^j}{\mathbf{y}_i^j} \right)^2}$
- Weighted least square distance function: $d(\mathbf{y}, \mathbf{f}_\theta(\mathbf{x})) = \sum_{j=1}^{n_Y} \omega_j \times \sqrt{\sum_{i=1}^n \psi_i \times (\mathbf{y}_i^j - \mathbf{f}_\theta(\mathbf{x})_i^j)^2}$ where $\{\psi_i\}_{i \in [1, n]}$ are weights used to ponderate each and every observations with respect to the others.
- Mahalanobis distance function: $d(\mathbf{y}, \mathbf{f}_\theta(\mathbf{x})) = \sum_{j=1}^{n_Y} \omega_j \times \sqrt{(\mathbf{y}^j - \mathbf{f}_\theta(\mathbf{x})^j)^T \Sigma^{-1} (\mathbf{y}^j - \mathbf{f}_\theta(\mathbf{x})^j)}$ where Σ is the covariance matrix of the observations.

These definitions are not orthogonal, indeed if $\{\psi_i\}_{i \in [1, n]} = \alpha, \alpha \in \mathbb{R}$, then the least-square function is equivalent to the weighted least-square one. This situation is concrete as it can correspond to the case where the least-square estimation is weighted with an uncertainty affecting the observations, assuming the uncertainty is constant throughout the data (meaning $\alpha = \sigma^{-2}$). This is called the **Homoscedasticity** hypothesis and it is important for the linear case, as discussed later-on.

One can also compare the relative and weighted least-square, if $\alpha = \mathbb{R}$ and $\{\psi_i = (\alpha \% \times y_i)^{-1}\}_{i \in [1,n]}$ these two forms become equivalent (the relative least-square is useful when uncertainty on observations is multiplicative). Finally if one assumes that the covariance matrix of the observations is the identity (meaning $\Sigma = \mathbf{1}$), the Mahalanobis distance is equivalent to the least-square distance.



Warning It is important to stress something here : it might seem natural to think that the lower the distance is, the closest to the real values our parameters are. Bearing this in mind would mean thinking that "having a null distance" is the ultimate target of calibration, which is actually dangerous. As for the general discussion in Chapter IV, the risk could be to overfit the set of parameters by "learning" just the set of observations at our disposal as the "truth", not considering that the residue (introduced in VII.1) might be here to introduce observation uncertainties. In this case, knowing the value of the uncertainty on the observations, the ultimate target of the calibration might be to get the best agreement of observations and model predictions within the uncertainty models, which can be translated into a distribution of the reduced-residue (that would be something like $\{(y^i - f_{\theta}^i)/\sigma_{\varepsilon_i}\}_{i \in [1,n]}$ in a scalar cases) behaving like a centred reduced Gaussian distribution.

VII.1.2 Discussing assumptions and theoretical background

VII.1.2.1 Calibration in the context of VVUQ principle

VVUQ is a known acronym standing for "Verification, Validation and Uncertainty Quantification". Within this framework, the calibration procedure of a model, also called sometimes "Inverse problem" [49] or "data assimilation" [50] depending on the hypotheses and the context, is an important step of uncertainty quantification. This step should not be confused with validation, even if both procedures are based on comparison between reference data and model predictions, their definition is recalled here [51]

validation: process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model.

calibration: process of improving the agreement of a code calculation or set of code calculations with respect to a chosen set of benchmarks through the adjustment of parameters implemented in the code.

The underlying question to validation is "What is the confidence level that can be granted to the model given the difference seen between the predictions and physical reality ?" while the underlying question of calibration is "Given the chosen model, what parameter's value minimise the difference between a set of observations and its predictions, under the chosen statistical hypotheses ?".

It can happen sometimes that a calibration problem allows an infinity of equivalent solutions [52], which is possible for instance when the chosen model f_{θ} depends explicitly on an operation of two parameters. The simplest example would be to have a model f_{θ} depending only on two parameters through the difference $\theta_1 - \theta_2$. In this peculiar case, every couple of parameters (θ_1, θ_2) that would lead to the same difference $\theta_1 - \theta_2$ would provide the exact same model prediction, which means that it is impossible to disentangle these solutions. This issue, also known as identifiability of the parameters is crucial as one needs to think at the way our chosen is parameterised [53].

Defining a calibration analysis consists in several important steps:

- Precise the ensemble of observations that will be used as reference;
- Precise the model that is supposed to fairly describe the real world;
- Define the parameters to be analysed (either by defining the *a priori* laws or at least by setting a range). This step is the moment where caution has to be taken on the identifiability issue.

- Choose the method used to calibrate the parameters.
- Choose the distance function used to quantify the distance between the observations and the predictions of the model.

VII.1.2.2 Interest in the least square measurement

The least-square distance function introduced in Section VII.1.1 is very classically used when considering calibration issue. This is true whether one is considering calibration within a statistical approach or not (see the discussion on uncertainty sources in Section VII.1). The importance of the least-square approach can be understood by adding an extra hypothesis on the residue defined previously. If one considers that the residue is normally distributed, it implies that one can write

$$\varepsilon_i = \mathcal{N}(0, \sigma_{\varepsilon_i}^2) \text{ for } i = 1, \dots, n,$$

where σ_{ε_i} can quantify both sources of uncertainty and whose values are supposed known. The formula above can be used to transform equation VII.1 into (setting $n_Y = 1$ for simplicity):

$$y_i \sim Y_i | \theta := \mathcal{N}(f_\theta(\mathbf{x}_i), \sigma_{\varepsilon_i}^2) \quad (\text{VII.2})$$

This particular case is very interesting, as from equation VII.2 it becomes possible to write down the probability of the observation set \mathcal{D} as the product of all its component probability which can be summarised as such:

$$L(\mathbf{y} | \theta) = \prod_{i=1}^n \ell(y_i | \theta) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma_{\varepsilon_i}} e^{-\frac{1}{2} \left(\frac{y_i - f_\theta(\mathbf{x}_i)}{\sigma_{\varepsilon_i}} \right)^2} \quad (\text{VII.3})$$

A logical approach is to consider that the fact the datasets \mathcal{D} has been observed, means that the probability of this collection of observations is highly probable. The probability defined in equation VII.3 can then be maximised by varying θ in order to get its most probable values. This is called the Maximum Likelihood Estimation (MLE) and maximising the likelihood is equivalent to minimise the logarithm of the likelihood which can be written as:

$$\log L(\mathbf{y} | \theta) = -\frac{n}{2} \log 2\pi \sigma_{\varepsilon_i}^2 - \frac{1}{2} \sum_{i=1}^n \left(\frac{y_i - f_\theta(\mathbf{x}_i)}{\sigma_{\varepsilon_i}} \right)^2.$$

EQUATION VII.1: Log-likelihood formula for a normally-distributed residue without homoscedasticity hypothesis

The first part of the right-hand side is independent of θ which means that minimising the log-likelihood is basically focusing on the second part of the right-hand side which is basically the weighted least-square distance with the weights set to $\{\psi_i = \sigma_{\varepsilon_i}^{-2}\}_{i \in [1, n]}$.

Finally on the way to get an estimation of the parameters in this case, it depends on the underlying hypotheses of the model, and this discussion is postponed to another section (this is discussed in Section VII.2). More details on least-square concepts can be found in many references, such as [54, 55].

VII.1.2.3 Introduction to Bayesian approach

The probability for an event to happen can be seen as the limit of its occurrence rate or as the quantification of a personal judgement or opinion as for its realisation. This is a difference in interpretation that usually split the frequentist and the Bayesian. For a simple illustration one can flip a coin : the probability of getting head, denoted $\mathbb{P}[\text{head}]$ is either the average result of a very large number of experiments (this definition being very factual but whose value depends

highly on the size of the set of experiments) or the intimate conviction that the coin is well-balanced or not (which is basically an *a priori* opinion that might be based on observations, or not).

Lets call (W, Z) a random vector with a joined probability density $f_{(W,Z)}(w, z)$ and marginal densities written as $f_W(w)$ and $f_Z(z)$. From there, the Bayes rules states that:

$$f_{W|Z}(w|z) = \frac{f_{Z|W}(z|w) \times f_W(w)}{f_Z(z)} \quad (\text{VII.4})$$

where $f_{W|Z}(w|z)$ (respectively $f_{Z|W}(z|w)$) is the conditional probability density of W knowing that z has been realised (and vice-versa respectively). These laws are called conditional laws.

Getting back to our formalism introduced previously, using equation VII.4 implies that the probability density of the random variable θ given our observations, which is called *posterior* distribution, can be expressed as

$$\pi_{post}(\theta|\mathbf{y}) = \frac{L(\mathbf{y}|\theta)\pi_{prior}(\theta)}{\pi(\mathbf{y})} \propto L(\mathbf{y}|\theta)\pi_{prior}(\theta). \quad (\text{VII.5})$$

In this equation, $L(\mathbf{y}|\theta)$ represents the conditional probability of the observations knowing the values of θ , $\pi_{prior}(\theta)$ is the *a priori* probability density of θ , often referred to as **prior**, $\pi(\mathbf{y})$ is the marginal likelihood of the observations, which is constant in our scope (as it does not depend on the values of θ but only on its prior, as $\pi(\mathbf{y}) = \int_{\Theta} L(\mathbf{y}|\theta)\pi_{prior}(\theta)d\theta$, it consists only in a normalisation factor).

The *prior* law is said to be proper when one can integrate it, and *improper* otherwise. It is conventional to simplify the notations, by writing $\pi(\theta|\mathbf{y})$ instead of $\pi_{post}(\theta|\mathbf{y})$ and also $\pi(\theta)$ instead of $\pi_{prior}(\theta)$. The choice of the prior is a crucial step when defining the calibration procedure and it must rely on physical constraints of the problem, expert judgement and any other relevant information. If one of these are available or reliable, it is still possible to use non-informative priors for which the calibration will only use the data as inputs. One can find more discussions on non-informative prior here [56, 57].

VII.2 Using minimisation techniques

The theory behind this method may seem very basic as it consists mainly in a point estimation of a correct configuration that could be performed without any underlying uncertainty modelling. This vision is over-simplistic, since numerical optimisation problems are not simple ones, even for mono-criterion optimisation, because of, for instance:

- the regularity and possible local minimum of the cost function under consideration;
- the size of the input spaces and the possible constraints that can be applied on each and every single input.

This is not even starting discussing the intricate problems that might arise when the calibration is facing identifiability problem. An insight might be seen using for instance evolutionary algorithms. For more methodological aspects on these issues, please refer to Chapter VI.

VII.3 Analytical linear Bayesian estimation

This method consists mainly in the analytical formulation of the posterior distribution when the hypotheses on the prior are well set: the problem can be considered linear and the prior distributions are normally distributed (or flat, this aspect being precised at the end of this section).

In the specific case of a linear model, one can write then $f_{\theta}(\mathbf{x}) = h^T(\mathbf{x})\theta$ where $h(\mathbf{x})$ is the regressor vector. This way of writing the model can include an "hidden virtual" $\theta_0 = 1$ whose purpose is to integrate a constant term into the

regression (to describe a pedestal). Using the statistical approach introduced in Section VII.1, one can also define the covariance matrix of the residue which will be written hereafter as $\Sigma = \text{diag}(\sigma_{\varepsilon_1}, \dots, \sigma_{\varepsilon_n})$

From there, one can construct the *conception matrix* $H = [h(\mathbf{x}_1), \dots, h(\mathbf{x}_n)]^T \in M_{n,p}(\mathbb{R})$ whose columns are defining the sub-space onto which the model is projected. With a normal prior, which follows the form $\theta \sim \mathcal{N}(m_\theta, \Sigma_\theta)$ the posterior is expected to be normal as well, meaning that it can be written $\pi(\theta|\mathbf{y}) \sim \mathcal{N}(m_\theta^{post}, \Sigma_\theta^{post})$ where its parameters are expressed as

$$m_\theta^{post} = \left(\Sigma_\theta^{-1} + H^T \Sigma^{-1} H \right)^{-1} \left(m_\theta^T \Sigma_\theta^{-1} + \mathbf{y}^T \Sigma^{-1} H \right)^T \quad (\text{VII.6})$$

and

$$\Sigma_\theta^{post} = \left(\Sigma_\theta^{-1} + H^T \Sigma^{-1} H \right)^{-1}. \quad (\text{VII.7})$$

Actually, one can also use, as already introduced in Section VII.1.2.3, non-informative **prior** such as the Jeffrey's prior: it is an improper flat prior ($\pi(\mathbf{y}) \propto 1$) [57], whose posterior distribution (in the linear case) is also Gaussian. For this prior, the parameters of the posterior are equivalent to the parameters of the posterior from a Gaussian prior, given in VII.6 and VII.7 only removing all reference to Σ_θ , as shown here:

$$m_\theta^{post} = \left(H^T \Sigma^{-1} H \right)^{-1} H^T \Sigma^{-1} \mathbf{y} \text{ and } \Sigma_\theta^{post} = \left(H^T \Sigma^{-1} H \right)^{-1}. \quad (\text{VII.8})$$

This final form is the results expected and obtained when only considering linear regression of the weighted least-squares approach [58].

VII.3.1 Prediction values

Once both the posterior parameter values and covariances are estimated, it is possible to get a prediction for a set of data not used to get the estimation. The central value of the prediction is easy to get, as for any other methods shown in this documentation, since one knows the model and can use the newly estimated posterior central values of the parameters.

What's new is the fact that a variance can be estimated as well for the predicted central value using the posterior covariance matrix of the parameters, Σ_θ^{post} , already introduced in VII.7. This variance is the variance for every new estimated points coming from the uncertainty of the parameters, and it is contained in the covariance matrix Σ_θ^{pred} whose dimension is (q, q) , where q is the size sample under consideration. To get the estimation one needs the new conception matrix $H_{pred} = [h(\mathbf{x}_1), \dots, h(\mathbf{x}_q)]^T \in M_{q,p}(\mathbb{R})$ which leads to

$$\Sigma_\theta^{pred} = \left(H_{pred} \Sigma_\theta^{post} H_{pred}^T \right). \quad (\text{VII.9})$$

VII.4 The Approximation Bayesian Computation techniques (ABC)

This sections is discussing methods gathered below the ABC acronym, which stands for *Approximation Bayesian Computation*. The idea behind these methods is to perform Bayesian inference without having to explicitly evaluate the model likelihood function, which is why these methods are also referred to as **likelihood-free** algorithms [59].

As a reminder of what's discussed in further details in Section VII.1.2.3, the principle of the Bayesian approach is recap in the equation $\pi_{post}(\theta|\mathbf{y}) = \frac{L(\mathbf{y}|\theta)\pi_{prior}(\theta)}{\pi(\mathbf{y})} \propto L(\mathbf{y}|\theta)\pi_{prior}(\theta)$. where $L(\mathbf{y}|\theta)$ represents the conditional probability of the observations knowing the values of θ , $\pi_{prior}(\theta)$ is the *a priori* probability density of θ (the **prior**) and $\pi(\mathbf{y})$ is the marginal likelihood of the observations, which is constant here. It does, indeed, not depend on the values of θ but only on its prior, as $\pi(\mathbf{y}) = \int_{\Theta} L(\mathbf{y}|\theta)\pi_{prior}(\theta)d\theta$ which makes this a normalisation factor.

VII.4.1 Rejection ABC algorithm

The rejection ABC is the simplest possible version of the ABC approach. Its prehistory genesis was stated in the eighties [60] and it is possible to see a nice introduction of the rejection algorithm as originally applied to a problem with a finite countable set \mathcal{V} of values in [61]. In this specific case, it only consisted in two random drawing: the parameter's value according to their prior then the model prediction according to the parameter's value just drawn. If the results was the element of the reference data set, the configuration was kept.

Things are getting a bit more tricky when considering continuous sample spaces as there is no such thing as strict equality when considering stochastic behaviour (without even discussing the numerical issues that have to arise at some points). This implies the need for two important concepts

- a distance measure on the output space, denoted $\rho(.,.)$;
- a tolerance determining the accuracy of the algorithm and denoted δ .

Unlike the simpler discrete case quickly introduced above where the aim is to have strict equality between the predictions and the reference data, here the accepted configurations would be those fulfilling the following condition

$$\rho(\mathbf{z}, \mathbf{y}) \leq \delta$$

where θ_T is the configuration under study draw following the prior $\pi(\theta)$ and \mathbf{z} is the model predictions estimated from f_{θ_T} once run on the reference datasets. This was firstly used in the late nineties, as can be seen in [62].



Warning A peculiar attention has to be taken to the way the model is defined : one should recall that the uncertainty model is defined on the residue, as stated in VII.1 and that residue is usually considered normally-distributed VII.2. Disregarding the origin of this residue, as discussed in Section VII.1, if the model one is providing is deterministic, the calibration will focus on a single realisation of the observation without uncertainty consideration. In this case, the model prediction must be modified to include a noise representative of the residue hypotheses [63].

This methodology shows that accepted configurations are not really taken out of the true posterior distribution $\pi(\theta|\mathbf{y})$ but they're coming from an approximation of it that can be written $\pi(\theta|\rho(\mathbf{z}, \mathbf{y}) \leq \delta)$. Two interesting asymptotic regime can be emphasised:

- when $\delta \rightarrow 0$: the algorithm is exact and is leading to the real $\pi(\theta|\mathbf{y})$;
- when $\delta \rightarrow \infty$: this algorithm does not use information from the reference datasets and gives back the original prior $\pi(\theta)$ instead.

There are many different version of this kind of algorithm, among which one could find an extra step using summary statistic $S(.)$ to project both \mathbf{z} and \mathbf{y} onto a lower dimensional space. In this version, the configurations kept are drawn from $\pi(\theta|\rho(S(\mathbf{z}), S(\mathbf{y})) \leq \delta)$.

Finally another possible way to select the best representative sub-sample might be by using a percentile of the analysed and computed set of configurations. Although, mainly recommended for high-dimension case (meaning when n is become large), this solution might works as long as one keep an eye on the residue distribution provided by the *a posteriori* estimated parameters. Indeed, if no threshold is chosen but a percentile is used, the requested number of configurations will always been brought at the end, but the only way to check whether the uncertainty hypotheses were correct is to look at how close the predictions have become for the full reference datasets.

VII.5 The Markov-chain approach

Unlike the Monte-Carlo methods already discussed in Chapter III to obtain design-of-experiments and which usually provides independent samples (which means that the successive observations are statistically independent unless correlation is purposely injected), the Monte-Carlo techniques describe here are called "Markov-chain" and they provide dependent samples as the estimation of the i -Th iteration only depends of the value of the previous one, the $(i-1)$ Th.

VII.5.1 Markov-chain principle

An usual approach to explain the Markov-chain theory on a continuous space is to start with a transition kernel $P(x, A)$ where $x \in \mathbb{R}^p$ and $A \in \mathcal{B}$, where \mathcal{B} is the Borel σ -algebra on \mathbb{R}^p [64]. This transition kernel is a conditional distribution function that represents the probability of moving from x to a point in the set A . It is interesting to notice two properties: $P(x, \mathbb{R}^p) = 1$ and $P(x, \{x\})$ is not necessarily zero, meaning that a transition might be possible from x to x . For a single estimation, from a given starting point x_0 , this can be summarised as $\mathbb{P}(x_1 \in A | x_0) = P(x_0, A)$. The Markov-chain is defined as a sequence using this transition kernel a certain number of time, leading to the k -Th estimation ($k \geq 1$) $\mathbb{P}(x_k \in A | x_0) = P^k(x_0, A)$ where P^k denotes the k -Th iteration of the kernel P [65].

The important property of a Markov-chain is the invariant distribution, π^* , which is the only distribution satisfying the following relation

$$\pi^*(dy) = \int_{\mathbb{R}^p} P(x, dy) \pi(x) dx \quad (\text{VII.10})$$

where π is the density with respect to the Lebesgue measure of π^* (meaning $\pi^*(dy) = \pi(y)dy$). This invariant distribution is an equilibrium distribution for the chain that is the target of the sequence of transition, as

$$\lim_{k \rightarrow \infty} P^k(x, A) = \pi^*(A)$$

The Monte-Carlo Markov-chain approach (hereafter called MCMC) is the following one: the invariant distribution is considered known, as it is the one from which one wishes to sample, while the transition kernel is unknown and to be determined. This might seem to be "the proverbial needle in a haystack" but the idea is to be able to write the target kernel through a transition kernel probability $p(x, y)$ (describing the move from x to y) as

$$P(x, dy) = p(x, y)dy + r(x)\delta_x(dy) \quad (\text{VII.11})$$

where $\delta_x(dy) = 1$ and 0 otherwise, while $r(x) = 1 - \int_{\mathbb{R}^p} p(x, y)dy$ is the probability that the chain remains at its current location. If the transition part of this function, $p(\cdot, \cdot)$, satisfies the *reversibility condition* (also called *time reversibility*, *detailed balance*, *microscopic reversibility*...)

$$\pi(x)p(x, y) = \pi(y)p(y, x) \quad (\text{VII.12})$$

then $\pi(\cdot)$ is the invariant density of $P(x, \cdot)$ [65].

VII.5.2 The Metropolis-Hasting algorithm

In the Metropolis-Hasting approach, the *candidate-generating* density is traditionally denoted $q(x, y)$. If this density satisfies the *reversibility condition* in equation VII.12 for all x and y the search is over (but this is very unlikely). What's more probable is to find something like $\pi(x)q(x, y) > \pi(y)q(y, x)$ that states that moving from x to y is happening too often (or the other way too scarcely).

The proposed way to correct this is to introduce a probability $\alpha(x, y) < 1$ where this $\alpha(x, y)$ is called the *probability of move* that is injected in the *reversibility condition* to help fulfil it. Without getting in too much details (see [67] which nicely discusses this), the probability of move is usually set to

$$\alpha(x, y) = \min \left[\frac{\pi(y)q(y, x)}{\pi(x)q(x, y)}, 1 \right], \quad \text{if } \pi(x)q(x, y) > 0$$

$$= 1, \quad \text{otherwise}$$

If the chain is currently at a point $x_k = x$, then it generates a value y accepted as x_{k+1} with the probability $\alpha(x, y)$. If rejected the chain remains at the current location and another drawing is performed from there.

With this, one can define the off-diagonal density of the Metropolis-Hasting kernel as function, $p(x, y) = q(x, y)\alpha(x, y)$ if $x \neq y$ and 0 otherwise and with thanks to equation VII.11, one has the invariant distribution for P [65].

Warning Two important things to notice here



- the obtained sample is obviously not independent as the $k+1$ -Th location is taken out from the k -Th one.
 - the very first drawn locations are usually rejected as part of the **burn-in** (also called **warm-up**) process. As discussed above, the algorithm needs a certain number of iteration to converge through the invariant distribution.
-

VII.5.2.1 The random walk choice

The idea here is to choose a family of candidate-generating densities that follows $q(x, y) = q_1(x - y)$ where $q_1(\cdot)$ is a multivariate density [66], a classical choice being q_1 set as a multivariate normal density. The candidate is indeed drawn as current value plus a noise, which is the origin of the random walk name.

Once the newly selected configuration-candidate is chosen, let's call it , θ_T the comparison with respect to the latest configuration kept, called here θ_k , is done through the ratio of likelihood, which allows to get rid of any constant factors and should look like this once transformed to its log form:

$$\log \left(\frac{L(\mathbf{y}|\theta_T)}{L(\mathbf{y}|\theta_k)} \right) = \frac{1}{2} \sum_{i=1}^n \left(\frac{y_i - f_{\theta_k}(\mathbf{x}_i)}{\sigma_{\epsilon_i}} \right)^2 - \frac{1}{2} \sum_{i=1}^n \left(\frac{y_i - f_{\theta_T}(\mathbf{x}_i)}{\sigma_{\epsilon_i}} \right)^2.$$

This result is then compared to the logarithm of a random uniform drawing between 0 and 1 to decide whether one should keep this configuration (as usually done in Monte-Carlo approach, see [67]).

There are few more properties for this kind of algorithm such as the acceptance rate, that might be tuned or used as validity check according to the dimension of our parameter space for instance [68, 69] or the lag definition, sometimes used to thin the resulting sample (whose usage is not always recommended as discussed in [70]). These subjects being very close to the implementation choices, they are not discussed here.

Chapter VIII

The Uncertainty modeler module

VIII.1 Introduction

We present in this section the quantitative comparison of an already existing sample to a given probability density function ("PDF") and the CIRCE method used to determine the uncertainty of non measurable physical model's parameters .

VIII.2 Tests based on the *Empirical Distribution Function* ("*EDF tests*")

This part is introducing comparison tests, sometimes called "goodness of fit" tests, which are used as test hypothesis. This idea is to check, when considering a certain variable, whether it is following a predefined law among the list of implemented ones: normal, lognormal and uniform ones. To do so, there are three different tests implemented in Uranie. If one calls $F_n(x)$ the *Empirical Distribution Function* of the law $F(x)$ (*i.e.* the distribution that we'd like to test) and $F_0(x)$ the reference law one wants to compare to, then, for n the number of data in the EDF, these tests are defined as:

Kolmogorov-Smirnov (D) [79]

$$D = \sup |F_0(X_i) - F_n(X_i)|_{i=1, \dots, n}$$

Anderson-Darling (A^2) [80]

$$A^2 = n \int \frac{|F_0(x) - F_n(x)|^2}{F_0(x)(1 - F_0(x))} dF_0(x) = -n - \frac{1}{n} \sum_{i=1}^n (2i - 1) \times [\log(F_n(X_i)) + \log(F_n(X_{n+1-i}))]$$

Cramer-VonMises (W^2) [81]

$$W^2 = n \int |F_0(x) - F_n(x)|^2 dF_0(x) = \frac{1}{12n} \sum_{i=1}^n \left(F_0(X_i) - \frac{2i-1}{2n} \right)^2$$

In these three formulas, the $(X_i)_{i=1, \dots, n}$ set represents the ordered data of the random variable x which comes usually as the CDF distribution for convenience.

VIII.3 The Circe method

The Circe method is a statistical approach which is applied as an alternative to the expert judgement, used to determine the uncertainty of physical model's parameters. These uncertainties can be tricky to estimate as some of these parameters might not be directly measurable. However, it might be possible to use SET (separate-effect tests) experiments, which are sensitive to the physical model, to derive an estimation of these uncertainties.

VIII.3.1 Main principle of the CIRCE method

As already stated previously, CIRCE (which stands for "Calcul des Incertitudes Relatives aux Correlations Elementaires") is a statistical method in which the uncertainties are defined through random variables, mean values and standard deviations [82, 83]. Usually, if one considers P_i the non-observed parameters (i being the number of these parameters, limited here to a certain number q), one can write the following equation

$$P_i = p_i \times P_i^{\text{nom}}$$

The physical parameter is then expressed as a function of a nominal value (P_i^{nom}) and a multiplier coefficient p_i . A relation can be constructed between these multipliers and the parameters considered by CIRCE, as

$$p_i = 1 + \alpha_i \text{ or } p_i = e^{\alpha_i}$$

EQUATION VIII.1: Relation between multipliers and CIRCE parameters.

The nominal value of α_i is set to 0 (leading to a nominal value of the influential physical model equal to 1). The other inputs needed by the method are the observed data (or responses) that will be hereafter called R_j^{exp} (j being a realisation of the SET experiment), and the corresponding code result R_j^{code} . CIRCE combines the difference between the experimental and the code results ($R_j^{\text{exp}} - R_j^{\text{code}}$) with the derivatives of each code response with respect to each parameter $\frac{\partial R_j^{\text{code}}}{\partial \alpha_i}$. It is also possible to take into account the experimental uncertainties of the response, called hereafter δR_j^{exp} . This procedure should lead to the estimation, for every α_i parameter, of its mean value b_i (for bias) and its standard deviation, σ_i .

In order to perform this estimation, there are two main hypothesis done by the CIRCE method:

- the linearity between the code response and each parameter α_i . This hypothesis is clearly visible since first-order derivatives are used for the estimation ($\frac{\partial R_j^{\text{code}}}{\partial \alpha_i}$). It is further discussed in Section VIII.3.1.1.
- the normality of the α_i parameters. An hypothesis on the PDF of CIRCE parameters is indeed compulsory, leading to the hypothesis of normality or lognormality of the p_i multiplier if respectively the additive or exponential change of variable is used in Equation VIII.1. This is further discussed in Section VIII.3.1.2.

VIII.3.1.1 The linearity hypothesis

For every response, the quantity of interest is $R_j^{\text{exp}} - R_j^{\text{code}}$ which can as well be written as, if one notes R_j^{real} the real value of the R_j response, $R_j^{\text{exp}} - R_j^{\text{code}} = (R_j^{\text{exp}} - R_j^{\text{real}}) + (R_j^{\text{real}} - R_j^{\text{code}})$. It is the sum of two independent random variables:

- $(R_j^{\text{exp}} - R_j^{\text{real}})$: the experimental uncertainty which obeys a centered normal law of known standard deviation σ^{exp} .

- $(R_j^{\text{real}} - R_j^{\text{code}})$: which is obtain from a first order development as

$$R_j^{\text{real}} - R_j^{\text{code}} = \sum_{i=1}^q \frac{\partial R_j^{\text{code}}}{\partial \alpha_i} (\alpha_{j,i} - \alpha_i^{\text{nom}})$$

In this definition, $\alpha_{j,i}$ is the unknown value to be given to the i-Th parameter so that $R_j^{\text{code}}(\alpha_{j,1}, \dots, \alpha_{j,q}) = R_j^{\text{real}}(\alpha_{j,i}$ being different for every response) and α_i^{nom} is the nominal value of this i-Th parameter (generally 0).

VIII.3.1.2 The normality hypothesis

If one gathers all the information about the system described up to now, the problem can be summarised as

$$R_j^{\text{exp}} - R_j^{\text{code}} = (R_j^{\text{exp}} - R_j^{\text{real}}) + (R_j^{\text{real}} - R_j^{\text{code}}) = e_j + \sum_{i=1}^q \frac{\partial R_j^{\text{code}}}{\partial \alpha_i} \times \alpha_{j,i}$$

In this expression, on can discuss the different contributions:

- $R_j^{\text{exp}} - R_j^{\text{code}}$ and $\frac{\partial R_j^{\text{code}}}{\partial \alpha_i}$ are known.
- e_j is a realisation of $\mathcal{N}(0, (\sigma^{\text{exp}})^2)$ where σ^{exp} is also known.
- The $\alpha_{j,i}$ are unknown. The only available information can be extracted through their statistical features: their bias b_i and their standard deviation σ_i .

There will be several solutions possible for the vector of α , leading to a needed choice among them. The criterion chosen to do so is the maximum of likelihood, which obliges to make an hypothesis on the form of the law followed by the α_i parameters. The normal hypothesis is then chosen.

Chapter IX

References

- [1] G. Arnaud. Méthodes pour l'optimisation many-objective par algorithme évolutionnaire. Technical report, CEA, STMF/LGLS/RT/17-005/A, 2017.
- [2] Eric W. Weisstein. Likelihood. <https://mathworld.wolfram.com/Likelihood.html>.
- [3] W. Appel. *Probabilité pour les non probabilistes*. H & K, Paris, 2013.
- [4] Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007.
- [5] Eckart Zitzler and Simon Künzli. Indicator-based selection in multiobjective search. In *International Conference on Parallel Problem Solving from Nature*, pages 832–842. Springer, 2004.
- [6] K. B. Petersen and M. S. Pedersen. The matrix cookbook, nov 2012. Version 20121115.
- [7] T.W. Simpson, J.D. Poplinski, P. N. Koch, and J.K. Allen. Metamodels for computer-based engineering design: Survey and recommendations. *Engineering with Computers*, 17(2):129–150, Jul 2001.
- [8] Xingyi Zhang, Ye Tian, and Yaochu Jin. A knee point-driven evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 19(6):761–776, 2015.
- [9] Eckart Zitzler. *Evolutionary algorithms for multiobjective optimization: Methods and applications*, volume 63. Citeseer, 1999.
- [10] Eckart Zitzler, Lothar Thiele, and Johannes Bader. On set-based multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 14(1):58–79, 2010.
- [11] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1):55–61, February 2000.
- [12] J. C. Helton and F. J. Davis. Illustration of sampling-based methods for uncertainty and sensitivity analysis. *Risk Analysis*, 22(3):591–622, 2002.
- [13] R. L. Iman and W. J. Conover. A distribution-free approach to inducing rank correlation among input variables. *Communications in Statistics - Simulation and Computation*, 11(3):311–334, 1982.
- [14] D. Morris and J. Mitchell. Exploratory designs for computational experiments. *Journal of Statistical Planning and Inference*, 43:381–402, 1995.
- [15] G. Damblin, M. Couplet, and B. Iooss. Numerical studies of space filling designs: optimization of Latin hypercube samples and subprojection properties. *Journal of simulation*, 7:276–289, 2013.

- [16] L. Pronzato and W. Muller. Design of computer experiments: space filling and beyond. *Statistics and Computing*, 22(3):681–701, 2012.
- [17] I.M Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86 – 112, 1967.
- [18] Leonardo S Bastos and Anthony O'Hagan. Diagnostics for gaussian process emulators. *Technometrics*, 51(4):425–438, 2009.
- [19] J. H. Halton. Algorithm 247: Radical-inverse quasi-random point sequence. *Commun. ACM*, 7(12):701–702, December 1964.
- [20] K. Petras. Fast calculation of coefficients in the smolyak algorithm. *Numerical Algorithms*, 26(2):93–109, 2001.
- [21] W.S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.
- [22] F. Rosenblatt. *Principles of neurodynamics: perceptrons and the theory of brain mechanisms*. Report (Cornell Aeronautical Laboratory). Spartan Books, 1962.
- [23] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366, July 1989.
- [24] A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, May 1993.
- [25] N. Wiener. The homogeneous chaos. *American Journal of Mathematics*, 60(4):897–936, 1938.
- [26] R. H. Cameron and W. T. Martin. The orthogonal development of non-linear functionals in series of fourier-hermite functionals. *Annals of Mathematics*, 48(2):385–392, 1947.
- [27] Roger G. Ghanem and Pol D. Spanos. *Stochastic Finite Elements: A Spectral Approach*. Springer-Verlag New York, Inc., New York, NY, USA, 1991.
- [28] M. Baudin and J.M. Martinez. Polynômes de chaos sous Scilab via la librairie NISP. In *42èmes Journées de Statistique*, Marseille, France, France, 2010.
- [29] Olivier Roustant, Fabrice Gamboa, and Bertrand Iooss. Sensitivity analysis and generalized chaos expansions. lower bounds for sobol indices. *arXiv preprint arXiv:1906.09883*, 2019.
- [30] G. Matheron. La théorie des variables régionalisées, et ses applications. *Fascicule 5 in Les Cahiers du Centre de Morphologie Mathématique de Fontainebleau*, 1970.
- [31] J.M. Martinez, A. Marrel, N. Gilardi, and F. Bachoc. Krigeage par processus gaussiens. Librairie gpLib. Technical report, CEA DEN/DANS/DM2S/STMF/LGLS/RT/12-026/A, 2012.
- [32] F. Bachoc. *Estimation paramétrique de la fonction de covariance dans le modèle de Krigeage par processus Gaussiens : application à la quantification des incertitudes en simulation numérique*. PhD thesis, Mathématiques appliquées, Paris 7, 2013. Thèse de doctorat dirigée par J. Garnier.
- [33] J.M. Martinez. Tutorial du krigeage dans uranie. Technical report, CEA DEN/DANS/DM2S/STMF/LGLS/NT/13-014/A, 2015.
- [34] A Saltelli, S. Tarantola, F. Campolongo, and M. Ratto. *Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models*. Wiley, New York, 2004.

- [35] A Saltelli, S. Tarantola, F. Campolongo, M. Ratto, T. Andres, J. Cariboni, D. Gatelli, and M. Saisana. *Global Sensitivity Analysis: The Primer*. Wiley, New York, 2008.
- [36] A Saltelli, K. Chan, and E.M. Scott. *Sensitivity Analysis*. Wiley, New York, 2008.
- [37] T. Homma and A. Saltelli. Importance measures in global sensitivity analysis of nonlinear models. *Reliability Engineering and System Safety*, 52:1–17, 1996.
- [38] R.L. Iman, M.J. Shortencarier, and J.D. Johnson. *FORTTRAN 77 program and users guide for the calculation of partial correlation and standardized regression coefficients*. Sandia National Laboratories, Jun 1985.
- [39] R. A. Fisher. On the probable error of a coefficient of correlation deduced from a small sample. *Metron*, 1:3–32, 1921.
- [40] A. Saltelli. Making best use of model evaluations to compute sensitivity indices. *Computer Physics Communications*, 145:280–297, 2002.
- [41] H. Monod, C. Naud, and D. Makowski. *Uncertainty and sensitivity analysis for crop models*. In D. Wallach, D. Makowski, and J. W. Jones, editors, 2006.
- [42] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.
- [43] Ian Jolliffe. *Principal component analysis*. Springer, 2011.
- [44] J.M. Martinez. Analyse de sensibilité globale par décomposition de la variance. Technical report, GdR Ondes et Mascot Num, institut Henri Poincaré, 2011.
- [45] I.M. Sobol'. Sensitivity indices for nonlinear mathematical models. *Mathematical Modelling and Computational Experiment 1*, 1993.
- [46] Bertrand Iooss and Clémentine Prieur. Shapley effects for sensitivity analysis with dependent inputs: comparisons with sobol' indices, numerical estimation and applications. *arXiv preprint arXiv:1707.01334*, 2017.
- [47] Richard M Johnson. The minimal transformation to orthonormality. *Psychometrika*, 31(1):61–66, 1966.
- [48] Jeff W Johnson. A heuristic method for estimating the relative weight of predictor variables in multiple regression. *Multivariate behavioral research*, 35(1):1–19, 2000.
- [49] Albert Tarantola. *Inverse problem theory*. SIAM, 2005.
- [50] Mark Asch, Marc Bocquet, and Maëlle Nodet. *Data assimilation. Methods, algorithms and applications*. SIAM, 2016.
- [51] Timothy G Trucano, Laura Painton Swiler, Takera Igusa, William L Oberkampf, and Martin Pilch. Calibration, validation, and sensitivity analysis: What's what. *Reliability Engineering & System Safety*, 91(10-11):1331–1357, 2006.
- [52] Christian Hansen. *Rank-Deficient and Discrete Ill-Posed Problems*. SIAM, 1996.
- [53] Eric Walter and Luc Pronzato. Identification of parametric models. *Communications and control engineering*, 8, 1997.
- [54] Ake Börck. *Numerical Methods for Least Squares Problems*. Society for Industrial Applied Mathematics, 1996.
- [55] V. Pereyra P. C. Hansen and G. Scherer. *Least Squares Data Fitting with Applications*. Johns Hopkins University Press, 2013.

- [56] Harold Jeffreys. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 186(1007):453–461, 1946.
- [57] Christèle Bioche. *Approximation de lois impropres et applications*. PhD thesis, 2015.
- [58] N. H. Bingham and John M. Fry. *Regression. Linear Models in Statistics*. Springer, 2010.
- [59] Richard David Wilkinson. Approximate bayesian computation (abc) gives exact results under the assumption of model error. *Statistical applications in genetics and molecular biology*, 12(2):129–141, 2013.
- [60] Donald B Rubin. Bayesianly justifiable and relevant frequency calculations for the applies statistician. *The Annals of Statistics*, pages 1151–1172, 1984.
- [61] Jean-Michel Marin, Pierre Pudlo, Christian P Robert, and Robin J Ryder. Approximate bayesian computational methods. *Statistics and Computing*, 22(6):1167–1180, 2012.
- [62] Jonathan K Pritchard, Mark T Seielstad, Anna Perez-Lezaun, and Marcus W Feldman. Population growth of human y chromosomes: a study of y chromosome microsatellites. *Molecular biology and evolution*, 16(12):1791–1798, 1999.
- [63] Elske van der Vaart, Dennis Prangle, and Richard M Sibly. Taking error into account when fitting models using approximate bayesian computation. *Ecological applications*, 28(2):267–274, 2018.
- [64] Jon Wakefield. *Bayesian and frequentist regression methods*. Springer Science & Business Media, 2013.
- [65] Luke Tierney. Markov chains for exploring posterior distributions. *the Annals of Statistics*, pages 1701–1728, 1994.
- [66] Peter Müller. *A generic approach to posterior integration and Gibbs sampling*. Purdue University, Department of Statistics, 1991.
- [67] Siddhartha Chib and Edward Greenberg. Understanding the metropolis-hastings algorithm. *The american statistician*, 49(4):327–335, 1995.
- [68] Andrew Gelman, Gareth O Roberts, Walter R Gilks, et al. Efficient metropolis jumping rules. *Bayesian statistics*, 5(599-608):42, 1996.
- [69] Gareth O Roberts, Andrew Gelman, Walter R Gilks, et al. Weak convergence and optimal scaling of random walk metropolis algorithms. *The annals of applied probability*, 7(1):110–120, 1997.
- [70] William A Link and Mitchell J Eaton. On thinning of chains in mcmc. *Methods in ecology and evolution*, 3(1):112–115, 2012.
- [71] Baptiste Broto, François Bachoc, Marine Depecker, and Jean-Marc Martinez. Sensitivity indices for independent groups of variables. *Mathematics and Computers in Simulation*, 163:19–31, 2019.
- [72] Art B Owen. Sobol’indices and shapley value. *SIAM/ASA Journal on Uncertainty Quantification*, 2(1):245–251, 2014.
- [73] M.J.W. Jansen. Analysis of variance designs for model output. *Computer Physics Communications*, 117, 1999.
- [74] G.J. McRae, J.W. Tilden, and J.H. Seinfeld. Global sensitivity analysis: a computational implementation of the fourier amplitude sensitivity test (fast). *Computers & Chemical Engineering*, 6(1):15 – 25, 1982.
- [75] A. Saltelli and R. Bolado. An alternative way to compute fourier amplitude sensitivity test (fast). *Computational Statistics & Data Analysis*, 26(4):445 – 460, 1998.

-
- [76] Sébastien Da Veiga. Global sensitivity analysis with dependence measures. *Journal of Statistical Computation and Simulation*, 85(7):1283–1305, 2015.
- [77] S. Tarantola, D. Gatelli, and T.A. Mara. Random balance designs for the estimation of first order global sensitivity indices. *Reliability Engineering & System Safety*, 91(6):717 – 727, 2006.
- [78] G. Arnaud. Manuel d'utilisation de Vizir distribué v2.0. Technical report, CEA, SFME/LGLS/RT/10-001/A, 2010.
- [79] A. N. Kolmogorov. Sulla Determinazione Empirica di una Legge di Distribuzione. *Giornale dell'Istituto Italiano degli Attuari*, 4:83–91, 1933.
- [80] T. W. Anderson and D. A. Darling. Asymptotic theory of certain goodness of fit criteria based on stochastic processes. *Ann. Math. Statist.*, 23(2):193–212, 06 1952.
- [81] T. W. Anderson. On the distribution of the two-sample cramer-von mises criterion. *Ann. Math. Statist.*, 33(3):1148–1159, 09 1962.
- [82] A. De Crécy. Circe: A methodology to quantify the uncertainty of the physical models of a code. Technical report, CEA DEN/DANS/DM2S/STMF/LGLS/RT/12-013/A, 2012.
- [83] A. De Crécy and P. Bazin. Determination of the uncertainties of the constitutive relationship of the CATHARE 2 code. *M&C 2001*, 2001.