Unlocking Large Scale Uncertainty Quantification with In Transit Iterative Statistics

Alejandro Ribés, Théophile Terraz, Yvan Fournier, Bertrand Iooss, Bruno Raffin

Unlocking Large Scale Uncertainty Quantification with In Transit Iterative Statistics

Alejandro Ribés, Théophile Terraz, Yvan Fournier, Bertrand Iooss, and Bruno Raffin

Abstract Multi-run numerical simulations using supercomputers are increasingly used by physicists and engineers for dealing with input data and model uncertainties. Most of the time, the input parameters of a simulation are modeled as random variables, then simulations are run a (possibly large) number of times with input parameters varied according to a specific design of experiments. Uncertainty quantification for numerical simulations is a hard computational problem, currently bounded by the large size of the produced results. This book chapter is about using in situ techniques to enable large scale uncertainty quantification studies. We provide a comprehensive description of Melissa, a file avoiding, adaptive, fault-tolerant, and elastic framework that computes in transit statistical quantities of interest. Melissa currently implements the on-the-fly computation of the statistics necessary for the realization of large scale uncertainty quantification studies: moment-based statistics (mean, standard deviation, higher orders), quantiles, Sobol' indices, and threshold exceedance.

1 Introduction

A numerical simulation is a calculation that is run on a computer following a program that implements a mathematical model for a physical system. Nowadays, engineers and scientists often use numerical simulation as a tool, and its use in industries or scientific laboratories is very broad. From the mathematical point of view, discretized differential equations are numerically solved, often using a mesh as spatial support. Popular methods include Finite Difference, Finite Volumes, Finite

Alejandro Ribés, Yvan Fournier and Bertrand Iooss
EDF R&D, France, e-mail: {alejandro.ribes,yvan.fournier,bertrand.ioss}@edf.fr

Théophile Terraz, Bruno Raffin
Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, 38000 Grenoble, France, e-mail: {theophile.terraz,bruno.raffin}@inria.fr

Elements, or particle-based methods. From the computer science point of view, a numerical simulation consists of a workflow of actions. First, the engineer or scientist prepares the mesh or other spatial discretization such as particles, and she/he defines the initial and boundary conditions. Second, the calculations are run in a computer and generate results that are written to files. Finally, the results are analyzed.

Chapter 1 of this book already presented an introduction to in situ techniques for computational science. We would like to insist on the fact that engineers and scientists are so used to the classical worklow presented above, that they currently do not realize that a major change is occurring in current computer systems. The sizes of the simulations are strongly increasing, and writing files is becoming cumbersome and time-consuming. This bottleneck, in numerous cases, limits the size of the simulations. When executing multiple simulation runs, this problem becomes even more critical.

Multiple simulation runs (sometimes several thousands) are required to compute sound statistics in the context of uncertainty quantification [44]. Taking uncertainties into account when dealing with complex numerical simulations is necessary to assess their robustness, as well as to answer tighter regulatory processes (security, safety, environmental control, health impacts, etc.). Many attempts at treating uncertainty in industrial applications (e.g. automotive and aerospace engine design, nuclear safety, agronomy, renewable energy production) have involved different mathematical approaches from many scientific domains as in metrology, structural reliability, variational analysis, design of experiments, machine learning and global sensitivity analysis [11]. As an example, in aeronautic and nuclear industries, uncertainty quantification approaches are applied on numerical models simulating non destructive testing procedures, in order to evaluate probability of detection curves [27]. Such curves allow the operators to evaluate the performance of their non destructive tests for the detection of harmful defects of the inspected structure, which is particularly important for the system safety.

Current practice consists of performing multiple executions of a classical workflow. All the necessary instances with different sets of input parameters are run, and the results are stored to disk, often called ensemble data, to later read them back from disk to compute statistics. In this context, we are confronted with two important problems. First, the amount of storage needed may quickly become overwhelming, with the associated long read time that makes statistic computing time-consuming. To avoid this pitfall, scientists reduce their study size by running low-resolution simulations or down-sampling output data in space and time. Second, humans should be able to somehow navigate through the complexity of these large simulation results. Down-sampling output data in space and time, extracting probes, or concentrating on specific features of the ensemble are usually performed to reduce complexity; this introduces strong dependences on a priori ideas related to the behavior of the ensemble.

Novel approaches are required. In situ and in transit processing emerged as a solution to perform data analysis starting as soon as the results are available in the memory of the simulation. The goal is to reduce the data to store to disk and to avoid the time penalty to write and then read back the raw data set as required by

the classical post hoc analysis approach. In recent works, we proposed the Melissa framework for the on-line data aggregation of high-resolution ensemble runs [46, 40]. As soon as each available simulation provides the results produced to a set of staging nodes, these nodes process them to update the statistics on a first-come-first-served basis thanks to one-pass algorithms. This in transit processing mode enables us to fully avoid storage of intermediate data on disks. Furthermore, this new approach allows the computation of *ubiquitous statistics*: we compute multidimensional and time-varying statistics, i.e. everywhere in space and time; instead of providing a down-sampled subset, for a limited sample of probes or concentrating on specific features of the ensemble, as usually done.

In the context of this book, we would like to remark that this chapter does not present any in situ visualization system but an example of how in situ techniques can be used for the statistical analysis of large quantities of data; which is defined in Chapter 1 as "use cases beyond exploratory analysis". As a matter of fact, uncertainty quantification for numerical simulations is a hard computational problem, currently bounded by the large size of the produced results. This chapter is about using in situ techniques to unleash large scale uncertainty quantification (UQ) studies.

In the following Section 2 presents the general methodology for dealing with UQ studies; Section 3 introduces the iterative statistics necessary to perform on-line UQ; Section 4 briefly describes the MELISSA platform which implements the statistics introduced in Section 3; Section 5 uses a fluid mechanics example to illustrate the realization of a large scale UQ study; finally, a short conclusion and a bibliography section end up the chapter.

2 Uncertainty Management Methodology

2.1 Introduction

A general framework has been proposed in order to deal with various uncertainties that arise in numerical simulations [11, 3]. The uncertainty management generic methodology is schematized in Figure 1. Based on a probabilistic modeling of the model input variables, it consists of the following steps:

- Step A: specify the random inputs X, the deterministic inputs d, the numerical model G (analytical, complex computer code or experimental process), the variable of interest (model output) Y and the quantity of interest on the output (central dispersion, its distribution, probability to exceed a threshold, . . .). The fundamental relation writes:

$$Y = G(X, d) = G(X),$$

  with $X = (X_1, \ldots, X_p) \in \mathbb{R}^p$.

- Step B: quantify the sources of uncertainty. This step consists in modeling the joint probability density function (pdf) of the random input vector by direct methods (e.g. statistical fitting, expert judgment).
- Step B': quantify the sources of uncertainty by indirect methods using some real observations of the model outputs. The calibration process aims to estimate the values or the pdf of the inputs while the validation process aims to model the bias between the model and the real system.
- Step C: propagate uncertainties to estimate the quantity of interest. With respect to this quantity, the computational resources and the CPU time cost of a single model run, various methods will be applied as linear-based analytical formula, geometrical approximations, Monte Carlo sampling strategies, metamodel-based techniques.
- Step C': analyze the sensitivity of the quantity of interest to the inputs in order to identify the most influential inputs (useful if one wants to reduce the output uncertainty) and to rank the uncertainty sources.

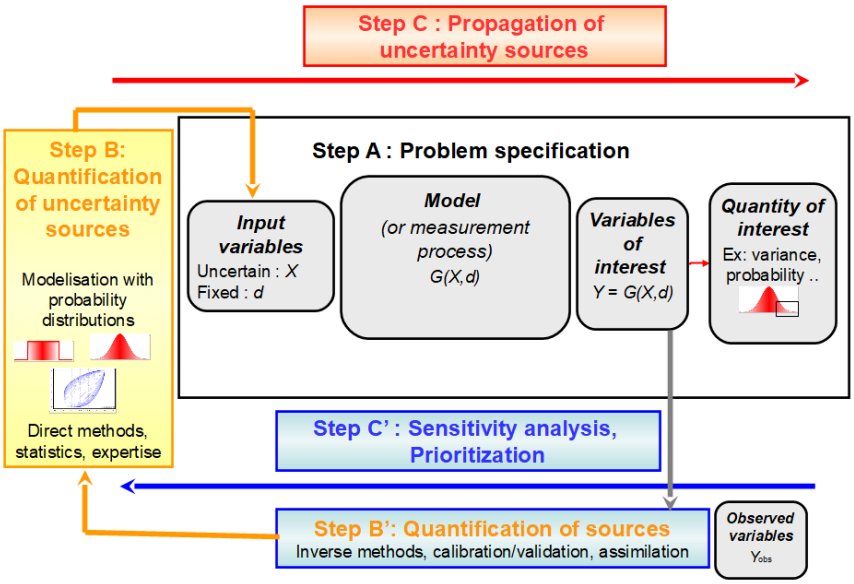


Figure 1 The methodology of uncertainty management in numerical simulation.

2.2 Quantiles of Simulation Outputs

Quantiles are important order statistics for outlier detection or computation of non parametric prediction and tolerance intervals. Then, in the context of uncertainty quantification analysis of computer models, quantile estimation is one of the key steps. Low or high-order quantiles are often required, especially in industrial safety studies [11, 33, 23].

Standard approaches deal with the problem of quantile estimation of scalar outputs [16, 17, 8]. Let us consider a N-sample $(Y_1, \ldots, Y_N)$ of independent and identically distributed random variables from an unknown distribution $f_Y(y)$. We look for an estimator $\hat{q}_\alpha$ of the α-quantile q_α defined by:

$$\mathbb{P}(Y \leq q_\alpha) = \alpha \,, \tag{1}$$

which is sometimes written as

$$q_\alpha = \inf\{y | \mathbb{P}(Y \leq y) \geq \alpha\} \,. \tag{2}$$

However, simulation models most often return spatial fields varying over time. For example, recent studies have considered quantiles of one-dimensional functional outputs (temporal curves) [37, 39, 34, 41], that demonstrates users' interest in computing these functional quantiles.

2.3 Sensitivity Analysis via Sobol' Indices

Sensitivity studies are an important application of uncertainty quantification in numerical simulation [44]. The objective of such studies can be broadly viewed as quantifying the relative contributions of individual input parameters to a simulation model, and determining how variations in parameters affect the outcomes of the simulations. In this context, multi-run studies treat simulations as black boxes that produce outputs when a set of parameters is fixed (Figure 2). Global sensitivity analysis is an ensemble of techniques that deal with a probabilistic representation of the input parameters [43, 22] to consider their overall variation range.

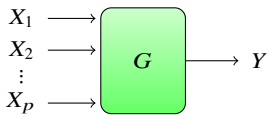


Figure 2 A simple solver G taking p input parameters X_1 to X_p, and computing a scalar output Y.

Variance-based sensitivity measures, also called Sobol' indices [45], are popular among methods for global sensitivity analysis because they can deal with nonlinear

responses. They decompose the variance of the output, Y, of the simulation into fractions, which can be attributed to random input parameters or sets of random inputs.

For example, with only two input parameters X_1 and X_2 and one output Y (Figure 2), Sobol' indices might show that 60% of the output variance is caused by the variance in the first parameter, 30% by the variance in the second, and 10% due to interactions between both. These percentages are directly interpreted as measures of sensitivity. If we consider p input parameters, the Sobol' indices can identify parameters that do not influence or influence very slightly the output, leading to model reduction or simplification. Sobol' indices can also measure the effect of interactions in non-additive systems.

Mathematically, the first and second order Sobol' indices [45] are defined by:

$$S_i = \frac{\mathrm{Var}(\mathrm{E}[Y|X_i])}{\mathrm{Var}(Y)}, \quad S_{ij} = \frac{\mathrm{Var}(\mathrm{E}[Y|X_iX_j])}{\mathrm{Var}(Y)} - S_i - S_j, \tag{3}$$

where $X_1, \ldots, X_p$ are p independent random variables. In Eq. (3), S_i represents the first order sensitivity index of X_i while S_{ij} represents the effect of the interaction between X_i and X_j. Higher-order interaction indices ($S_{ijk}, \ldots, S_{1\ldots p}$) can be similarly defined. The total Sobol' indices express the overall sensitivity of an input variable X_i:

$$ST_i = S_i + \sum_{j \neq i} S_{ij} + \sum_{j \neq i, k \neq i, j<k} S_{ijk} + \ldots + S_{1\ldots p} \ . \tag{4}$$

The previous formula applies for a scalar output Y. Some authors have proposed the generalization of the concept of Sobol' indices for multidimensional and functional data [14] by synthesizing all the sensitivity information of the multidimensional output in a single sensitivity value. Few authors have considered the estimation of Sobol' indices at each output cell (see [32] for an overview on this subject) and this estimation has always been applied to small models. Applications of these techniques on environmental assessment can be found for example in [18, 30] for spatial outputs and in [31, 29] for spatio-temporal outputs. All these works have shown that obtaining temporal/spatial/spatio-temporal sensitivity maps leads to powerful information for the analysts. Indeed, the parameter effects are localized in time or space, and can be easily examined in relation with the studied physical phenomena.

3 In Transit Statistics

Computing statistics from N samples classically requires $O(N)$ memory space to store these samples. But if the statistics can be *computed in one-pass* (also called iterative, on-line or even parallel [35]), i.e. if the current value can be updated as soon as a new sample is available, the memory requirement goes down to $O(1)$ space. With this approach, not only simulation results do not need to be saved, but they can

be consumed in any order, loosening synchronization constraints on the simulation executions.

There also exist multi-pass algorithms for the computation of statistics, where P passes are necessary. In this case, the on-line processing system would need to have access to the same data P times thus forcing the data to be stored till the last pass is finished. This is not feasible for large scale use cases. These approaches, along with the classical ones requiring $O(N)$ memory space, are avoided in on-line applications.

3.1 Moment-Based Statistics: Mean, Std, Higher Orders

One-pass variance algorithms were proposed in [48, 9, 12]. Numerically stable, one-pass formulas for arbitrary centered statistical moments and co-moments are presented in [5, 35]. Reference Pébay et al. [35] also contains update formulas for higher order moments (skewness, kurtosis and more). These works set the base for a module of parallel statistics in the VTK scientific visualization toolkit [36]. In this context, the one-pass algorithms enables to compute partial results in parallel before performing a reduction to get the final result. These iterative statistics were used for computing large scale parallel statistics for a single simulation run either from raw data files [7], compressed data files [25] or in situ [6]. More recently Lampitella et al. [26] proposed a general update formula for the computation of arbitrary-order, weighted, multivariate central moments.

In Melissa, we iteratively compute the moments of a random variable Y as

$$\mu_{(k),\mathcal{S}}(Y) = \mu_{(k),\mathcal{S}} = \mu_{(k),\mathcal{S}_1} + \frac{1}{n}\left(y^k - \mu_{(k),\mathcal{S}_1}\right) \tag{5}$$

for $k = 1, 2, 3$ and 4, where $\mathcal{S} = \mathcal{S}_1 \cup \{y\}$ and $n = \text{card}(\mathcal{S})$. Then from these moments, we compute the mean, variance, skewness and kurtosis:

$$\begin{aligned}
\text{Mean} &= \mu_{(1)}\,, \\
\text{Variance} &= \frac{n}{n-1}(\mu_{(2)} - \mu_{(1)}^2)\,, \\
\text{Skewness} &= \frac{\mu_{(3)} - 3\mu_{(1)}\mu_{(2)} + 2\mu_{(1)}^3}{(\mu_{(2)} - \mu_{(1)}^2)^{1.5}}\,, \\
\text{Kurtosis} &= \frac{\mu_{(4)} - 4\mu_{(1)}\mu_{(3)} + 6\mu_{(1)}^2\mu_{(2)} - 3\mu_{(1)}^4}{(\mu_{(3)} - 3\mu_{(1)}\mu_{(2)} + 2\mu_{(1)}^3)^2}\,.
\end{aligned}$$

3.2 Sobol' Indices

The information contained in this section can be found in the previously published article [46]. We include it here for self-completeness. Thus the reader can find the description, in this book chapter, of all iterative methods currently implemented in Melissa.

In order to compute Sobol' indices, we use the so-called pick-freeze scheme that uses two random independent and identically distributed samples of the model inputs [45, 19, 24]. One-pass iterative Sobol' indices formulas directly derive from the iterative variance (presented in Section 3.1) and iterative covariance [35]. Note that another iterative computation of Sobol' indices has been introduced in [15] for the case of a scalar output.

Our goal is to compute in transit the Sobol' indices of each input parameter X_i (Figure. 2). We explain below the so-called pick-freeze scheme that uses two random independent and identically distributed samples of the model inputs [45, 19, 24].

We first define the p variable input parameters of our study as a random vector, with a given probabilistic law for each parameter. We then randomly draw two times n sets of p parameters, to obtain two matrices A and B of size $n \times p$ (each row is a set of parameters for one simulation):

$$A = \begin{pmatrix} a_{1,1} & \cdots & a_{1,p} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,p} \end{pmatrix}; B = \begin{pmatrix} b_{1,1} & \cdots & b_{1,p} \\ \vdots & \ddots & \vdots \\ b_{n,1} & \cdots & b_{n,p} \end{pmatrix}.$$

For each $k \in [1, p]$ we define the matrix C^k, which is equal to the matrix A but with its column k replaced by column k of B. Each row of each matrix is a set of input parameters:

$$C^k = \begin{pmatrix} a_{1,1} & \cdots & a_{1,k-1} & b_{1,k} & a_{1,k+1} & \cdots & a_{1,p} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ a_{i,1} & \cdots & a_{i,k-1} & b_{i,k} & a_{n,k+1} & \cdots & a_{i,p} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,k-1} & b_{n,k} & a_{n,k+1} & \cdots & a_{n,p} \end{pmatrix}.$$

Then, a study consists in running the $n \times (p+2)$ simulations defined by the matrices A, B and C^k for $k \in [1, p]$. For each matrix M with n rows, $\forall i \in [1, n]$, let M_i be the i^{th} row of M, and $M_{[:i]}$ the matrix of size $i \times p$ built from the i first lines of M. For example :

$$C_i^k = \begin{pmatrix} a_{i,1} & \cdots & a_{i,k-1} & b_{i,k} & a_{i,k+1} & \cdots & a_{i,p} \end{pmatrix}$$

and

$$C_{[:i]}^k = \begin{pmatrix} a_{1,1} & \cdots & a_{1,k-1} & b_{1,k} & a_{1,k+1} & \cdots & a_{1,p} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ a_{i,1} & \cdots & a_{i,k-1} & b_{i,k} & a_{n,k+1} & \cdots & a_{i,p} \end{pmatrix}.$$

Let Y_i^A be the result of $G(A_i)$, and $Y^A \in \mathbb{R}^n$ the vector built from component i of Y_i^A, $\forall i \in [1,n]$. We define Y_i^B and Y^B in the same way, as well as $Y_i^{C^k}$ and Y^{C^k} $\forall k \in [1,p]$. Let $\mathrm{Var}(x)$ be the unbiased variance estimator, and $\mathrm{Cov}(x,y)$ the unbiased covariance estimator, as defined in [35]. First order Sobol' indices S_k can be estimated by the following formula, called Martinez estimator [2]:

$$S_k(f,A,B) = \frac{\mathrm{Cov}(Y^B, Y^{C^k})}{\sqrt{\mathrm{Var}(Y^B)}\sqrt{\mathrm{Var}(Y^{C^k})}}\,, \tag{6}$$

while total order Sobol' indices ST_k are estimated by:

$$ST_k(f,A,B) = 1 - \frac{\mathrm{Cov}(Y^A, Y^{C^k})}{\sqrt{\mathrm{Var}(Y^A)}\sqrt{\mathrm{Var}(Y^{C^k})}}\,. \tag{7}$$

Since variances and covariances can be updated iteratively, first order and total Sobol' indices can be computed from these formulas. The covariance update formula between two random variables X and Y writes [35]:

$$\mathrm{Cov}_{\mathcal{S}} = \mathrm{Cov}_{\mathcal{S}_1} + \frac{n-1}{n}\left[x - \mu_{(1),\mathcal{S}_1}(X)\right]\left[y - \mu_{(1),\mathcal{S}_1}(Y)\right] \tag{8}$$

where $\mathcal{S} = \mathcal{S}_1 \cup \{x,y\}$, $n = \mathrm{card}(\mathcal{S})$ and the mean update $\mu_{(1)}$ comes from Eq. (5).

There are many other estimators than those of Eqs (6) and (7) (see for example [38]) relying on the matrices A, B and C^k to compute the variance and the covariance with different formulas. We use the Martinez estimator because it provides an asymptotic confidence interval [2], which is very simple to express, and is easy to compute in an iterative fashion. In addition, it has been shown to be unbiased and one of the most numerically stable estimator.

3.3 Order Statistics: Quantiles

The classical estimator of the α-quantile y_α of the random variable Y is the empirical quantile, based on the notion of order statistics [10]. Essentially, we associate with the independent and identically distributed sample $(Y_1,\ldots,Y_N)$ the ordered sample $(Y_{(1)},\ldots,Y_{(N)})$ in which $Y_{(1)} \leq \ldots \leq Y_{(N)}$. The empirical estimator then writes:

$$\hat{q}_\alpha = Y_{(\lfloor \alpha N \rfloor + 1)}, \tag{9}$$

where $\lfloor x \rfloor$ is the integer part of x.

For an iterative statistical estimation, the Robbins-Monro estimator [42] consists in updating the quantile estimate $q_\alpha(n)$ at each new observation Y_{n+1} with the following rule:

$$q_\alpha(n+1) = q_\alpha(n) - \frac{C}{n^\gamma}\left(\mathbf{1}_{Y_{n+1} \leq q_\alpha(n)} - \alpha\right)\,, \tag{10}$$

with $n = 1 \dots N$, $q_\alpha(1) = Y_1$ an independent realization of Y, $\hat{q}_\alpha = q_\alpha(N)$, $\mathbb{1}_x$ the indicator function, C a strictly positive constant and $\gamma \in]0, 1]$ the step of the gradient descent of the stochastic algorithm. Under several hypotheses with $\gamma \in]0.5, 1]$, this algorithm has been shown to be consistent and asymptotically normal. A fine tuning of the constant C is important; several numerical tests have shown that a value of C of the order of the dispersion of Y (for example its standard deviation or an interquantile interval) would be satisfactory [20, 21].

Asymptotically (where N is large), a value $\gamma = 1$ is known to be optimal. However, in practical studies, N is often not large. For example, in nuclear safety studies (see for example [8, 23]), $\alpha = 0.95$ and N is in the order of several hundreds of simulated values. In this case and as we look for γ values that can work for different distributions of Y (which are unknown in practice), we propose to define γ as a function of n. Indeed, one can observe that a good γ value for a certain type of probability distribution produces bad results for another type of distribution (for example, $\gamma = 0.6$ gives good results for a normal distribution and incorrect quantile estimates for a uniform one). We then use the following heuristic formula for γ:

$$\gamma(n) = 0.1 + 0.9\frac{n-1}{N-1} \ . \tag{11}$$

The idea is to have strong mixing properties at the beginning of the algorithm (with small γ), then to slow down the potential variation of the quantile estimation all along the iterations of the algorithm.

Slightly different linear profiles can be proposed as $\gamma(n) = 0.5[1+(n-1)/(N-1)]$. Several tests on simple analytical functions (where the true quantile can be known) have been performed in order to calibrate and validate these γ-profiles [40, 20]. Other algorithmic developments are currently under study to further improve the robustness of the Robbins-Monro estimate.

3.4 Probability of Threshold Exceedance

If y_{crit} is a safety output value, a classical failure probability estimation problem occurs:

$$p_f = \mathbb{P}(Y > y_{\text{crit}}) \ , \tag{12}$$

well-known in structural reliability [28, 4]. For this issue, without loss of generality, we can turn to:

$$p_f = \mathbb{P}(Y < 0) \ .$$

Computing a failure probability can be seen as a direct problem of uncertainty propagation [33].

If the the failure domain is defined by $\mathcal{D}_f = \{x \in \chi \subseteq \mathbb{R}^p \mid G(x) \leq 0\}$, the probability that the event – failure occurs is given by

$$p_f = \mathbb{P}(G(X) \leq 0) = \int_{\mathcal{D}_f} f_X(x)dx = \int_{\mathcal{X}} \mathbb{1}_{G(x)\leq 0} f_X(x)dx = \mathbb{E}[1_{G(X)\leq 0}], \quad (13)$$

where f_X is the joint probability density function of X. One of the goals of a structural reliability study is to provide an estimate of p_f and the uncertainty involved. The complexity of models and large potential number of input variables means that, in general, we cannot calculate the exact probability of failure. The evaluation of the integral in formula (13) is the subject of numerous mathematical techniques, laid out in an abundant array of international scientific literature [11, 44]. The use of Monte Carlo simulation methods is the most common. The naive Monte Carlo estimator is

$$\hat{p_f} = \frac{1}{n}\sum_{i=1}^{n} 1_{\{G(x^{(i)})\leq 0\}}, \quad (14)$$

where the $x^{(i)}$ are n independent and identically distributed random vectors simulated according to f_X. This is an unbiased estimate of the quantity of interest, for which it is possible to control the precision via its variance and the provision of a confidence interval (thanks to the central limit theorem).

Of course, as this estimation is based on an expectation, the trivial iterative algorithm for the mean (see Section 3.1) applies to compute p_f on the fly.

4 The Melissa Framework

Melissa (Modular External Library for In Situ Statistical Analysis) proposes a new approach to compute statistics at large scale by avoiding to store the intermediate results produced by the multiple parallel simulation runs. The key enabler is the use of iterative formulations for the required statistics. This allows for updating statistics on-the-fly each time new simulation results are available. To manage the simulation runs as well as the in transit computation of iterative statistics, we developed a full framework built around an elastic and fault tolerant parallel client/server architecture. The benefits of this framework are multiple:

- **Storage saving:** no intermediate files are generated. Melissa fully avoids storage of intermediate data on disks.
- **Time saving:** simulations run faster when sending data to the server than when writing their results to disk. Statistics are computed while simulations are running, saving the time of post hoc statistic computing that, in addition, requires time to read back simulation results once all are performed.
- **Ubiquitous:** performance and scalability gains enable computing ubiquitous multidimensional and time varying statistics, i.e. everywhere in space and time, instead of providing statistics for a limited sample of probes as usually done with post hoc approaches to reduce the amount of temporary data storage.
- **Adaptive:** simulations can be defined, started or interrupted on-line according to past runs behavior or the statistics already computed.

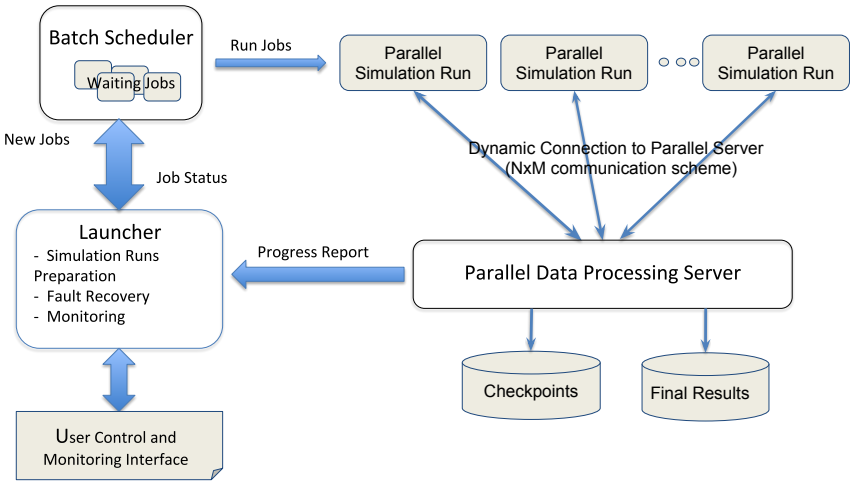


Figure 3 Melissa three tier architecture. The launcher oversees the execution in tight link with the batch scheduler. The job scheduler regulates the number of simulation jobs to run according to the machine availability, leading to an elastic resource usage. The parallel server, started first, process incoming data as soon as received from the connected simulations. A fault tolerance mechanism automatically restarts the failing simulation runs or a failing parallel server.

- **Elasticity:** Melissa enables the dynamic adaptation of compute resource usage according to availability. Simulations are independent and connect dynamically to the parallel server when they start. They are submitted as independent jobs to the batch scheduler. Thus, the number of concurrently running simulations providing data to the server can vary during the course of a study to adapt to the availability of compute resources.
- **Fault tolerance:** Melissa's asynchronous client/server architecture supports a simple yet robust fault tolerance mechanism. Only some lightweight bookkeeping and a few heartbeats are required to detect issues and restart the server or the simulations, with limited loss of intermediate results.

4.1 Melissa Architecture

Melissa is an open source framework [1] that relies on a three tier architecture (Figure 3). The *Melissa Server* aggregates the simulation results and updates iterative statistics as soon as a new result is available. The *Melissa clients* are the parallel simulations, providing their outputs to the server. *Melissa Launcher* interacts with

[1] `https://melissa-sa.github.io`

the batch scheduler and server, for creating, launching, and supervising the server and clients. We present in this section an overview of the Melissa architecture. Please refer to [46] for more details.

4.1.1 Melissa Server

Melissa Server is parallel and runs on several nodes. The number of nodes required for the server is driven by 1) memory needs 2) data pressure. The amount of memory needed for each computed statistic field is of same order as the size of the output field of one simulation (number of timesteps $\times$ the number of cells or points in the mesh). The number of server nodes should also be large enough to process incoming data without stalling the simulations.

4.1.2 Dynamic Connection to Melissa Server

When a simulation starts, it dynamically connects to the Melissa Server. Each simulation process opens individual communication channels to each necessary server process for enabling a $N \times M$ data redistribution. Every time new results are available, simulation processes push the results toward Melissa Server.

Melissa is designed to keep intrusion into the simulation code minimal. Melissa provides 3 functions to integrate in the simulation code through a dynamic library. The first function (Initialize) allocates internal structures and connects the simulation to the server. At each timestep, the second function (Send) sends the simulation data to its corresponding Melissa Server processes. The third function (Finalize) disconnects the simulation and releases the allocated structures.

4.1.3 Melissa Launcher

Melissa Launcher takes care of generating the parameter sets, requesting the batch scheduler to start the server and the clients, and track the progress of the running server and clients jobs. It first submits to the batch scheduler a job for the Melissa Server. Then, the launcher retrieves the server node addresses (the server is parallelized on several nodes) and submits the simulation jobs. Each simulation is submitted to the batch scheduler as a standalone job, making Melissa very elastic, i.e. capable of adapting to a varying compute resource availability. Simulations can be submitted all at once or at a more regulated pace depending on the machine policy for job submissions.

4.1.4 Fault Tolerance

The Melissa asynchronous client/server architecture leverages the iterative statistics computations to support a simple yet robust fault tolerance mechanism. Melissa supports detection and recovery from failures (including straggler issues) of Melissa Server and simulations, through heartbeats and server checkpointing. Melissa Launcher communicates with the server and the batch scheduler to detect simulation or server faults. As every simulation runs in a separate job, the failure of one simulation does not impact the ongoing study: Melissa launcher simply restarts it and the server discard already processed messages. Please refer to [46] for a complete description of this fault tolerance system.

5 An Illustrative Example

5.1 A Large Scale Study

We used *Code_Saturne* [1], an open-source computational fluid dynamics tool designed to solve the Navier-Stokes equations, with a focus on incompressible or dilatable flows and advanced turbulence modeling. *Code_Saturne* relies on a finite volume discretization and allows the use of various mesh types, using an unstructured polyhedral cell model, allowing hybrid and non-conforming meshes. The parallelization [13] is based on a classical domain partitioning using MPI, with an optional second (local) level using OpenMP.

5.1.1 Use Case

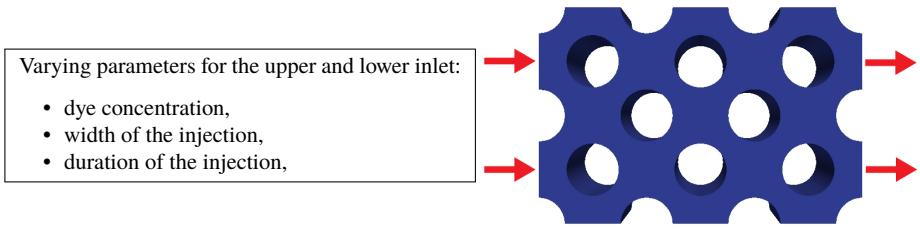


Figure 4 Use case: water flows from the left, between the tube bundle, and exits to the right with 6 varying parameters.

We validated our implementation on a fluid mechanics demonstration case simulating a water flow in a tube bundle (Figure 4). The mesh is composed of 6002400

hexahedra. We generate a multi-run sensitivity study by simulating the injection of a tracer or dye along the inlet, with 2 independent injection surfaces, each defined by three varying parameters (Figure 4). The solved scalar field representing dye concentration could be replaced by temperature or concentration of chemical compounds in actual industrial studies.

To initialize our multi-run study, we first ran a single 1000 timesteps simulation, to obtain a steady flow. Each simulation consists of 100 time steps starting from this steady flow, with different parameter sets.

This study ran a total of 80 000 simulations for computing ubiquitous variances, covariances, Sobol' indices, and the 5^{th}, 25^{th}, 50^{th}, 75^{th} and 95^{th} percentiles on the 6M hexahedra and 100 timesteps. Sobol' index computations rely on the pick-freeze method that requires to run groups of simulations with non-independent parameter sets. These correlated simulations are not used for the quantiles that are computed from the remaining independent 20 000 simulations.

The study took a total 260 000 CPU hours for the simulations and 11 112 CPU hours for the server (4% of the total CPU time). Melissa Server processed on-line a cumulated total of 288 TB of data coming from the simulations.

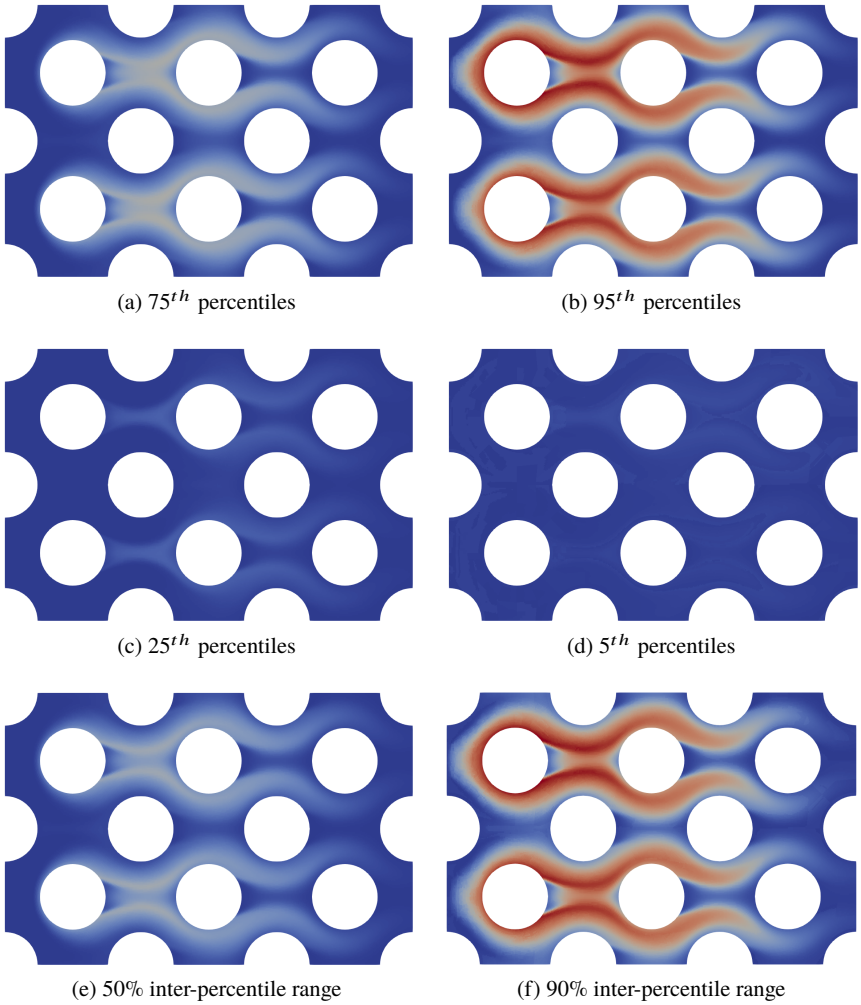
5.2 Ubiquitous Statistic Interpretation

In this section we interpret the ubiquitous statistics computed during the experiments. By ubiquitous statistics we mean the statistics of multidimensional and time varying physical quantities, i.e. statistics everywhere in space and time. Using ParaView we have chosen a timestep and performed a slice on a mid-plane of the mesh presented in Figure 4. This slice is aligned with the direction of the fluid. The chosen timestep belongs to the last temporal part of the simulation (80^{th} timestep over 100). This operation reduces the ubiquitions statistics to 2D spatial maps, thus allow us to generate the images of this section. We remark that this is just a choice for illustration purposes, any other visualisation pipeline can be applied to the ubiquitous statistics.

5.2.1 Quantiles

Figure 5 presents six spatial maps extracted from the ubiquitous quantiles. We concentrate in percentiles because they are easily interpreted. We recall that a percentile is not a per cent but a type of quantile: a 10-quantile. Our system can also calculate 4-quantiles, the so-called quartiles that are popular measures in statistics, or any other kind of quantile.

On the four top panels of Figure 5, Figure 5a, 5b, 5c and 5d, we present the 75^{th}, 95^{th}, 25^{th}, 5^{th} percentiles, respectively. On the two bottom panels the interpercentile ranges containing 50% and 90% of the samples are shown. Interpercentile ranges are easily computed from percentiles by substraction: the 50% interpercentile range corresponds to the 75^{th} percentile minus the 25^{th} percentile; the 90% interpercentile



(a) 75^{th} percentiles (b) 95^{th} percentiles

(c) 25^{th} percentiles (d) 5^{th} percentiles

(e) 50% inter-percentile range (f) 90% inter-percentile range

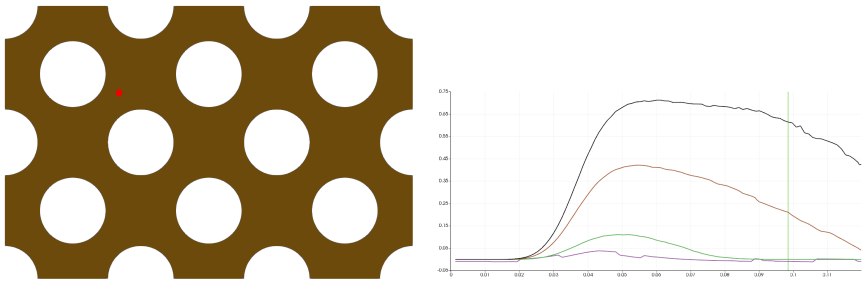
Figure 5 Percentiles and inter-percentile ranges maps on a slice of the mesh at timestep 80. The top line corresponds to the percentiles while the bottom line corresponds to inter-percentile ranges. All maps share the same scale.

range corresponds to the 95^{th} percentile minus the 5^{th} percentile. In Figure 5 each column shows an interpercentile map on the bottom and the percentile maps that served for its calculation above it. Looking at these maps an analyst can deduce several things:

1. Extreme high percentile maps such as 95^{th}, Figure 5b, give an idea of the distribution of the upper bounds of all simulations. In our use case, we can asses which spatial areas contain low quantities of contaminant. Indeed areas coloured

in blue necessarily contain low contaminant concentrations for any simulation in the multi-run study. Extreme low percentiles maps, such as 5^{th} has also a direct interpretation in the opposite sense.

2. Interpercentile-range maps such as Figure 5e or 5f are maps that show the spatial variability of statistical dispersion. Indeed, scalar interpercentile ranges are non-parametric measures of statistical dispersion, which means that no a priori knowledge about the distribution of the data is needed. This characteristic makes these ranges both general and robust. Visualising a map of such a measure of dispersion allows to understand how the data distribution is spatially concentrated. In our use case the low percentile maps used to calculate the interpercentile maps are mainly close to zero for all cells of the mesh, which makes these maps resemble the higher percentiles maps. However, this is in general not true. Moreover, if we visually compare Figure 5a and 5e we realise that both maps are different.



(a) Probe position (b) Quantiles evolution over time

Figure 6 A probe in a cell of the mesh allows an extraction of the temporal evolution of percentiles at a specific spatial location.

The maps shown in Figure 5 are static and 2D but we recall that we calculate ubiquitous percentiles, thus 3D and time dependent data is available. Figure 6 shows the temporal evolution of a probe positioned in the mesh using ParaView. At a specific location, a temporal evolution of all computed quantiles can be performed. In Figure 6b this evolution is plotted for the 95^{th}, 75^{th}, 25^{th} and 5^{th} percentiles. The vertical line indicates the position of the current time step (80^{th} time step). This figure clearly shows how the output variability of the ensemble study depends on time. Indeed, all simulations contain no dye for the first 15 time-steps, which is the time the dye takes to propagate from the top injector to the spatial location of the probe. After this point, we observe a moment where the variability of the dye concentration is the highest before a general decrease.

Figure 6b can be seen as the evolution of a Tukey boxplot [47] over time. In fact, the 25^{th} and 75^{th} percentiles correspond to the 1^{st} and 3^{rd} quartiles thus delimit the central box of the plot, while the 5^{th} and 95^{th} quantiles can be a choice for the whiskers. Using this analogy, we can easily observe that the dispersion of the dye

concentration on the whole ensemble moves over time. Furthermore, the distribution of this quantity is not symmetrical and its asymmetry is evolving over time.

Finally, Figure 7 shows a different representation of the evolution of the dye concentration at a fixed probe (the same than in Figure 6). At different regularly sampled time steps, the quantile functions of the concentration values are plotted (as a function of the order of the quantiles, between 0% and 100%). One can first observe zero-valued quantile functions for the first time steps (time steps 4 and 14). Indeed, at the probe, the dye concentration is zero during the first times of the injection. Then, from time step 24 to time step 44, all the values of the quantile functions regularly increase. It means that the dye concentration values homogeneously increase from 0, reaching a maximal value close to 0.82 for the 100%-order quantile. At the end of the simulation time, from time step 54 to time step 94, the quantile functions are regularly displaced to the right. The values close to zero disappear and the concentration of strong values becomes more and more important. As a conclusion, thanks to the quantile functions, this graph allows to finely and quantitatively analyze the temporal evolution of this dye concentration phenomena. We remark that, because we have calculated the ubiquitous percentiles, it is possible to obtain Figure 6 and Figure 7 for any location on the simulation domain.

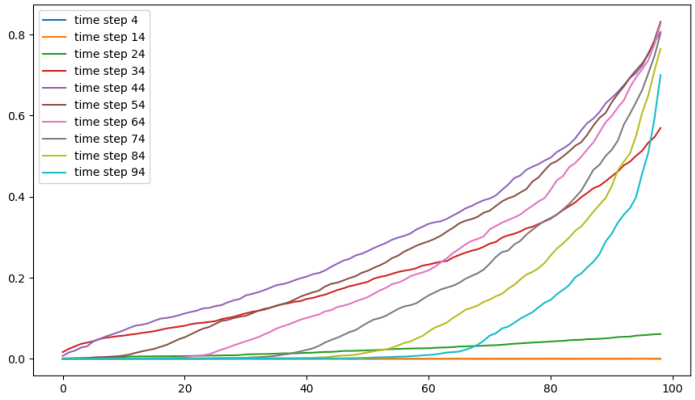
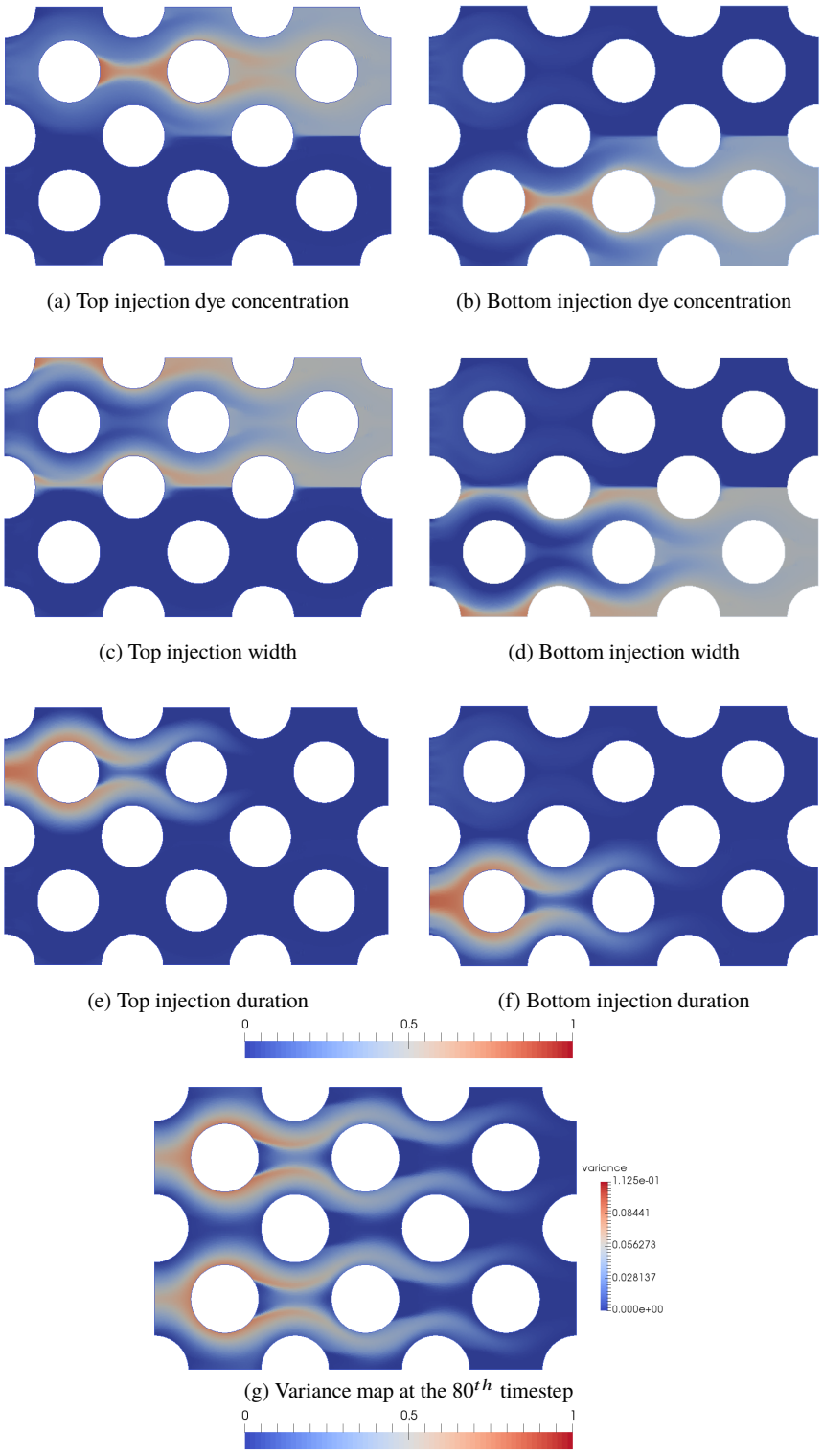


Figure 7 Percentile functions of the dye concentration at different time steps of the simulation. Vertical axis represents dye concentration. Horizontal axis represents percentiles. Each curve corresponds to different time steps of the simulation. All curves have been extracted from the probe position shown in Figure 6a.

5.2.2 Sobol' Indices

In this section we interpret the Sobol' indices computed during the experiments. Figure 8 presents six spatial first order Sobol' maps extracted from the ubiquitous indices. Looking at these figures an analyst can deduce several things:



(a) Top injection dye concentration

(b) Bottom injection dye concentration

(c) Top injection width

(d) Bottom injection width

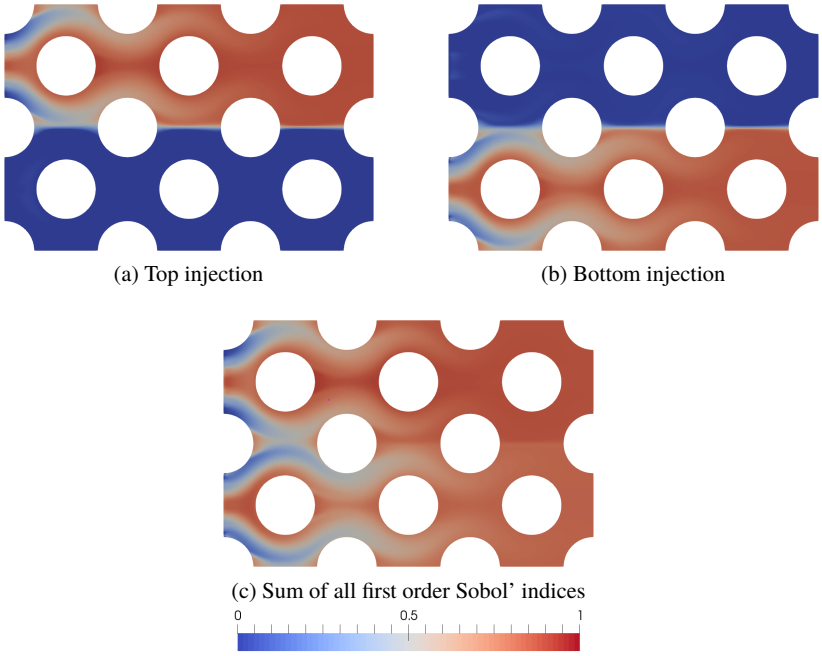
a

(e) Top injection duration

(f) Bottom injection duration

(g) Variance map at the 80^{th} timestep

Figure 8 First order Sobol' index maps on a slice of the mesh at timestep 80. The left column corresponds to the Sobol' indices for the upper injector while right corresponds to the bottom injector. All maps are scaled between zero (blue) and one (red).



(a) Top injection (b) Bottom injection

(c) Sum of all first order Sobol' indices

Figure 9 Addition of first order Sobol' index maps on a slice of the mesh at timestep 80. The left panel corresponds to the addition of the Sobol' indices for the upper injector while right corresponds to addition for the bottom injector. All maps are scaled between zero (blue) and one (red).

1. The **width of the injections** influence locations far up or down of each injector (Figure 8c and 8d). This is because the injectors are located in the center of the inlet segments, as depicted by the arrows in Figure 4. The parameter controlling the aperture of an injector makes a thinner or wider aperture around a central position. When the aperture's width is small, all dye is injected on the center while very few will go up and down. On the opposite, when the injector is at its maximal aperture, dye can easily flow to extreme vertical locations. In any case, the dye always flows in the center of the injectors which makes this parameter not influential around these areas.
2. The Sobol' maps for the **duration of the injection** (Figure 8e and 8f) can be understood by thinking about the temporal evolution of the simulation. When the simulations are just started, they all inject dye and the duration represents the time at which this injection is stopped. Figure 8 represents a state near the end of the simulation. Thus, it is not surprising that this parameter does not influence the right part of the domain (where all simulations were injecting dye) while it strongly influences the left side (where some simulations already stopped injecting while others continued).

3. Finally, the **dye concentration** mostly influences the areas were the other parameters have less influence, that is to say the center of the top and bottom channels, and the right side of the flow (Figure 8a and 8b).

We recommend co-visualizing Sobol' indices with the variance of the whole model output; Figure 8g shows this variance map for the Sobol' indices presented in Figure 8. One of the reasons of this co-visualization is that $Var(Y)$ appears as a denominator in Eq. (3), consequently when $Var(Y)$ is very small or zero, the Sobol' indices have no sense due to numerical errors or can even produce a zero division. Furthermore, it is not conceptually interesting to try to understand which input parameters influence low variance areas of the simulation once we know that 'not much happens' in these areas.

5.3 Combining Sobol' Indices

The sum of Sobol' indices should be 1 and they partition the total variance of the outputs. These two characteristics allow Sobol' indices to be grouped (added) or subtracted. If the indices are added, the resulting "Sobol group index" keeps normalised between zero and one. We show two examples of the use of such groups:

1. Figure 9a corresponds to the sum of the Sobol' indices for the upper injector while Figure 9b corresponds to the sum for the bottom injector. We clearly observe that the three parameters that define the behavior of the upper injector have no influence in the lowest half part of the simulation domain. This phenomenon can also be observed by simultaneously looking at the Sobol' maps of the three parameters for the upper injector (Figure 8a, 8c and 8e). However, the grouped Sobol map is clearer and gives a straightforward answer to the question "What is the influence of the upper injector?". Respectively, the parameters of the bottom injector do not influence the upper part of the simulation as directly observed on Figure 9b (or looking at Figure 8b, 8d and 8f).
2. We also calculate $S_1 + ... + S_n$, with $n = 6$, shown in Figure 9c, which gives us an indication of the importance of the interactions between parameters. Red areas of Figure 9c indicate that the first order Sobol indices are responsible of the behaviour of the simulation. On the contrary, when this sum is near zero (blue in the map) this means that the kind of relationship between input parameters and model outputs is complex and involves cross-effects among several parameters. We observe that the areas where interactions among Sobol' indices are important concentrates spatially around the injectors.

6 Conclusion

Dealing with uncertainty quantification may require executing from thousands to millions of runs of the same simulation, making it an extremely compute-intensive process that will fully benefit from Exascale machines. However, the large amount of data generated is a strong I/O bottleneck if the intermediate data is saved to disk. The proposed approach, implemented in the Melissa framework, demonstrates that combining one-pass statistics algorithms with an elastic and fault-tolerant in transit data processing architecture drastically reduces the amount of intermediate data stored, making it possible to compute ubiquitous statistics.

Melissa currently allows the in transit execution of large scale uncertainty quantification studies. This open-source software[2] currently implements the computation of the statistical tools necessary for this purpose: moment-based statistics (mean, standard deviation, higher orders), quantiles, Sobol' indices, and threshold exceedance. Future work will include the integration of Melissa with the OpenTurns uncertainty quantification software to consolidate and broaden their respective capabilities.

References

1. Archambeau, F., Méchitoua, N., Sakiz, M.: Code_saturne: a finite volume code for the computation of turbulent incompressible flows. International Journal on Finite Volumes **1** (2004)
2. Baudin, M., Boumhaout, K., Delage, T., Iooss, B., Martinez, J.M.: Numerical stability of sobol' indices estimation formula. In: Proceedings of the 8th International Conference on Sensitivity Analysis of Model Output (SAMO 2016). Le Tampon, Réunion Island, France (2016)
3. Baudin, M., Dutfoy, A., Iooss, B., Popelin, A.: Open TURNS: An industrial software for uncertainty quantification in simulation. In: R. Ghanem, D. Higdon, H. Owhadi (eds.) Springer Handbook on Uncertainty Quantification, pp. 2001–2038. Springer (2017)
4. Bect, J., Ginsbourger, D., Li, L., Picheny, V., Vazquez, E.: Sequential design of computer experiments for the estimation of a probability of failure. Statistics and Computing **22**, 773–793 (2012)
5. Bennett, J., Grout, R., Pébay, P., Roe, D., Thompson, D.: Numerically stable, single-pass, parallel statistics algorithms. In: Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on, pp. 1–8. IEEE (2009)
6. Bennett, J.C., Abbasi, H., Bremer, P.T., Grout, R., Gyulassy, A., Jin, T., Klasky, S., Kolla, H., Parashar, M., Pascucci, V., et al.: Combining in-situ and in-transit processing to enable extreme-scale scientific analysis. In: High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for, pp. 1–9. IEEE (2012)
7. Bennett, J.C., Krishnamoorthy, V., Liu, S., Grout, R.W., Hawkes, E.R., Chen, J.H., Shepherd, J., Pascucci, V., Bremer, P.T.: Feature-based statistical analysis of combustion simulation data. IEEE transactions on visualization and computer graphics **17**(12), 1822–1831 (2011)
8. Cannamela, C., Garnier, J., Iooss, B.: Controlled stratification for quantile estimation. Annals of Apllied Statistics **2**, 1554–1580 (2008)
9. Chan, T.F., Golub, G.H., LeVeque, R.J.: Updating formulae and a pairwise algorithm for computing sample variances. In: COMPSTAT 1982 5th Symposium held at Toulouse 1982, pp. 30–41. Springer (1982)
10. David, H., Nagaraja, H.: Order statistics, third edn. Wiley, New-York (2003)

[2] `https://melissa-sa.github.io/`

11. de Rocquigny, E., Devictor, N., Tarantola, S. (eds.): Uncertainty in industrial practice. Wiley (2008)
12. Finch, T.: Incremental calculation of weighted mean and variance. Tech. rep., University of Cambridge (2009)
13. Fournier, Y., Bonelle, J., Moulinec, C., Shang, Z., Sunderland, A., Uribe, J.: Optimizing code_saturne computations on petascale systems. Computers & Fluids **45**(1), 103 – 108 (2011). 22nd International Conference on Parallel Computational Fluid Dynamics (ParCFD 2010)ParCFD
14. Gamboa, F., Janon, A., Klein, T., Lagnoux, A.: Sensitivity analysis for multidimensional and functional outputs. Electronic Journal of Statistics **8**(1), 575–603 (2014)
15. Gilquin, L., Arnaud, E., Prieur, C., Monod, H.: Recursive estimation procedure of sobol' indices based on replicated designs. Submitted, http://hal.univ-grenoble-alpes.fr/hal-01291769 (2017)
16. Glynn, P.W.: Importance sampling for monte carlo estimation of quantiles. In: Proceedings of Second International Workshop on Mathematical Methods in Stochastic Simulation and Experimental Design, pp. 180–185. Publishing House of Saint Petersburg University (1996)
17. Hesterberg, T., Nelson, B.: Control variates for probability and quantile estimation. Management Science **44**, 1295–1312 (1998)
18. Higdon, D., Gattiker, J., Williams, B., Rightley, M.: Computer model calibration using high-dimensional output. Journal of the American Statistical Association **103**, 571–583 (2008)
19. Homma, T., Saltelli, A.: Importance measures in global sensitivity analysis of non linear models. Reliability Engineering and System Safety **52**, 1–17 (1996)
20. Iooss, B.: Estimation itérative en propagation d'incertitudes : réglage robuste de l'algorithme de Robbins-Monro. Actes des 52èmes Journées de Statistiques de la Société Française de Statistique (SFdS) pp. 466–471 (2020)
21. Iooss, B.: Robust tuning of Robbins-Monro algorithm for quantile estimation in iterative uncertainty quantification. Preprint hal-02918478, https://hal.archives-ouvertes.fr/hal-02918478v1 (2020)
22. Iooss, B., Lemaître, P.: A review on global sensitivity analysis methods. In: C. Meloni, G. Dellino (eds.) Uncertainty management in Simulation-Optimization of Complex Systems: Algorithms and Applications, pp. 101–122. Springer (2015)
23. Iooss, B., Marrel, A.: Advanced methodology for uncertainty propagation in computer experiments with large number of inputs. Nuclear Technology **205**, 1588–1606 (2019)
24. Janon, A., Klein, T., Lagnoux, A., Nodet, M., Prieur, C.: Asymptotic normality and efficiency of two sobol index estimators. ESAIM: Probability and Statistics **18**, 342–364 (2014)
25. Lakshminarasimhan, S., Jenkins, J., Arkatkar, I., Gong, Z., Kolla, H., Ku, S.H., Ethier, S., Chen, J., Chang, C.S., Klasky, S., et al.: Isabela-qa: query-driven analytics with isabela-compressed extreme-scale scientific data. In: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, p. 31. ACM (2011)
26. Lampitella, P., Inzoli, F., Colombo, E.: Note on a formula for one-pass, parallel computations of arbitrary-order, weighted, multivariate central moments (2015)
27. Le Gratiet, L., Iooss, B., Browne, T., Blatman, G., Cordeiro, S., Goursaud, B.: Model assisted probability of detection curves: New statistical tools and progressive methodology. Journal of Nondestructive Evaluation **36**, 8 (2017)
28. Lemaire, M., Chateauneuf, A., Mitteau, J.C.: Structural reliability. Wiley (2009)
29. Marrel, A., De Lozzo, M.: Sensitivity analysis with dependence and variance-based measures for spatio-temporal numerical simulators. Stochastic Environmental Research and Risk Assessment, In Press (2017)
30. Marrel, A., Iooss, B., Jullien, M., Laurent, B., Volkova, E.: Global sensitivity analysis for models with spatially dependent outputs. Environmetrics **22**, 383–397 (2011)
31. Marrel, A., Perot, N., Mottet, C.: Development of a surrogate model and sensitivity analysis for spatio-temporal numerical simulators. Stochastic Environmental Research and Risk Assessment **29**, 959–974 (2015)
32. Marrel, A., Saint-Geours, N.: Sensitivity analysis of spatial and/or temporal phenomena. In: R. Ghanem, D. Higdon, H. Owhadi (eds.) Springer Handbook on Uncertainty Quantification. Springer (2017)

33. Morio, J., Balesdent, M.: Estimation of rare event probabilities in complex aerospace and other systems. Woodhead Publishing (2016)
34. Nanty, S., Helbert, C., Marrel, A., Pérot, N., Prieur, C.: Uncertainty quantification for functional dependent random variables. Comput. Stat. **DOI 10.1007/s00180-016-0676-0** (2016)
35. Pébay, P.: Formulas for robust, one-pass parallel computation of covariances and arbitrary-order statistical moments. Sandia Report SAND2008-6212, Sandia National Laboratories **94** (2008)
36. Pébay, P., Thompson, D., Bennett, J., Mascarenhas, A.: Design and performance of a scalable, parallel statistics toolkit. In: Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on, pp. 1475–1484. IEEE (2011)
37. Popelin, A.L., Iooss, B.: Visualization tools for uncertainty and sensitivity analyses on thermal-hydraulic transients. In: Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo 2013 (SNA + MC 2013). Paris, France (2013)
38. Prieur, C., Tarantola, S.: Variance-based sensitivity analysis: Theory and estimation algorithms. In: R. Ghanem, D. Higdon, H. Owhadi (eds.) Springer Handbook on Uncertainty Quantification. Springer (2017)
39. Ribés, A., Pouderoux, J., Popelin, A.L., Iooss, B.: Visualizing statistical analysis of curves datasets in Paraview. In: 2014 IEEE Conference on Visual Analytics Science and Technology (VAST). Paris, France (2014)
40. Ribés, A., Terraz, T., Fournier, Y., Iooss, B., Raffin, B.: Large scale in transit computation of quantiles for ensemble runs. Unpublished Technical Report, arXiv: 1905.04180 (2019)
41. Ribés, A., Pouderoux, J., Iooss, B.: A Visual Sensitivity Analysis for Parameter-Augmented Ensembles of Curves. Journal of Verification, Validation and Uncertainty Quantification **4**(4) (2020). DOI 10.1115/1.4046020. URL `https://doi.org/10.1115/1.4046020`. 041007
42. Robbins, H., Monro, S.: A stochastic approximation method. The Annals of Mathematical Statistics **22**, 400–407 (1951)
43. Saltelli, A., Chan, K., Scott, E. (eds.): Sensitivity analysis. Wiley Series in Probability and Statistics. Wiley (2000)
44. Smith, R.: Uncertainty quantification. SIAM (2014)
45. Sobol, I.: Sensitivity estimates for non linear mathematical models. Mathematical Modelling and Computational Experiments **1**, 407–414 (1993)
46. Terraz, T., Ribés, A., Fournier, Y., Iooss, B., Raffin, B.: Melissa: Large scale in transit sensitivity analysis avoiding intermediate files. In: International Conference for High Performance Computing, Networking, Storage and Analysis (SC'17). Denver (2017)
47. Tukey, J.W.: Exploratory data analysis, vol. 2. Reading, Mass. (1977)
48. Welford, B.: Note on a method for calculating corrected sums of squares and products. Technometrics **4**(3), 419–420 (1962)