Large scale in transit computation of quantiles for ensemble runs

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

Large scale in transit computation of quantiles for ensemble runs

Alejandro Ribés[1], Théophile Terraz[2], Bertrand Iooss[3,4], Yvan Fournier[3], Bruno Raffin[2]

[1] *EDF Lab Paris-Saclay, France*
[2] *Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, France*
[3] *EDF Lab Paris-Chatou, France*
[4] *Institut de Mathématiques de Toulouse, Université Paul Sabatier, Toulouse, France*

Abstract

While estimating the uncertainties of numerical simulation model outputs, quantiles are important statistical quantities that can be used in risk analysis, outlier detection or computation of confidence intervals. Quantiles being order statistics, the classical approach for their computation requires availability of the full sample before ranking it. In numerical simulation, this approach is not suitable at exascale as large ensembles of model runs would need to gather a prohibitively large amount of data. This paper solves this problem by using an iterative approach based on the stochastic quantile algorithm of Robbins-Monro whose parameters are finely tuned in order to gain robustness. The computational part of the approach relies on the Melissa framework, a file avoiding, adaptive, fault tolerant and elastic framework. Quantiles are updated on-the-fly as soon as the in transit parallel server receives results from one of the running simulations. We validate our approach on a use case based on 3000 fluid dynamics parallel simulations of 6M hexahedra and 100 time-steps. This validation case was executed on two supercomputers, avoiding 11 TB of file storage per execution. Ubiquitous spatio-temporal maps of quantiles and inter-quantile based intervals are then produced via our robustly tuned Robbins-Monro algorithm.

Keywords: In Situ Data Processing, Parametric Studies, Uncertainty Quantification, Iterative Statistics, Quantile, Robbins-Monro

1. Introduction

On-line analytics for large scale numerical simulations has demonstrated its potential to contain the I/O bottleneck, improve simulation and analytics performance, and overall reduce the human time scientists spent in handling large data sets. A key enabler is the availability of one-pass analytics algorithms, i.e. algorithms that can operate on a reduced window of data recently produced by the simulation to avoid the need for massive storage and its associated performance pitfalls. In this paper, we focus on ensemble runs where multiple instances, usually thousands, of the simulation are run to sweep across the parameter space. Such process is typically led in engineering practice during the uncertainty quantification stage of some simulation models [1]. This stage mainly consists in computing statistical quantities of the model outputs (mean, variance, quantile, probability of threshold exceedance, ...) or estimating sensitivity indices between model outputs and inputs (linear correlation coefficients, Sobol' indices, ...) [2].

A major difficulty arises when the ensemble runs produce massive amount of data that have to be statistically aggregated, making them extremely vulnerable to the I/O bottleneck. To keep a manageable amount of data, the classical approach, used in most of the studies, consists in reducing the resolution of the simulation and the sampling points for the statistics [3]. A more suitable technique would be to use one-pass statistical algorithms. One-pass statistical algorithms, also called iterative, online or even parallel statistics have the interesting property of requiring only to store the current results that can next be updated with incoming new samples. One-pass variance algorithms were proposed in [4, 5, 6]. Numerically stable, one-pass formulas for arbitrary centered statistical moments and co-moments are presented in [7, 8]. [8] also contains update formulas for higher order moments (skewness, kurtosis and more), setting the bases for a module of parallel statistics in the VTK scientific visualization toolkit [9]. In this context the one-pass algorithms enables to compute partial results in parallel before to perform 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 [10], compressed data files [11] or in situ [12]. In sensitivity analysis of model outputs, for the estimation of Sobol' indices, [13] introduced a one-pass iterative computation for the case of a scalar model output, while [14] applied the iterative covariance formulas on massive output data (a spatio-temporal model output).

Various packages are designed for managing uncertainty quantification and sensitivity analysis from ensemble runs (see for example [3]). However, they all rely on classical non-iterative algorithms, requiring to accumulate first all simulation results in file or memory if doable. Based on a different architecture, the Melissa framework [15, 14] has been recently proposed for the on-line data aggregation of high resolution ensemble runs. Other in situ processing frameworks (see [16, 17, 18, 19]) enable in situ and in transit processing but for a single simulation. In Melissa, each simulation handles its output as soon as available to a set of staging nodes. These nodes process these incoming data to update the statistics on a first-come first-served basis thanks to the one-pass algorithm. This in transit processing mode enables to fully avoid storage of intermediate data on disks. Melissa runs a parallel server that stores the current state of the computed statistics. Simulations dynamically connects to this server and send their results a soon as available to update the statistics. This architecture, complemented with a fault tolerance mechanism allows for efficient elastic executions.

Melissa currently supports the estimation of the following statistical quantities: standard deviation, skewness, kurtosis, minimum, maximum, threshold exceedance probability and Sobol' indices. This paper focuses on extending the Melissa framework to the quantile estimation issue, useful quantity for risk analysis, outlier detection or computation of non-parametric confidence intervals. Low or high-order quantiles are often required during uncertainty quantification studies, especially for industrial safety issues [20, 21, 22]. Standard approaches deal with the problem of quantile estimation of scalar outputs [23, 24, 25]. However, simulation models often return more complex objects as outputs, such as temporal curves and spatial fields, which can be considered as functions. Recent studies have considered quantiles of one-dimensional functional outputs (temporal curves) [26, 27, 28] and demonstrated the interest for the practitioners to compute these functional quantiles.

Quantiles being order-statistics, the straightforward and classical approach consists in ordering the sample, then finding the appropriate quantiles [29]. This strategy necessitates the storage of the full-sample, thus making its application in an on-line context impossible. To avoid this storage, one can use stochastic algorithms which are devoted to the recursive estimation of statistical quantities. For instance, the Robbins-Monro algorithm allows for the iterative computation of quantiles [30, 31]. It has been introduced in the context of simulation model by [32], but only for a scalar output. In this paper, a new robust version of this algorithm is developed. Then, instead of providing quantiles for a limited sample of probes as usually done, full spatio-temporal model outputs can be considered in order to estimate ubiquitous (i.e. everywhere in space and time) multidimensional and time varying quantiles.

The paper is organized as follows. After presenting the quantile estimation issue and proposed algorithm (Sec. 2), its implementation in the Melissa architecture is discussed (Sec. 3) before presenting the experimental results (Sec. 4). The visual analysis of the obtained data reveals the potential of ubiquitous quantile statistics (Sec. 5). A conclusion closes the paper (Sec. 6).

2. In Transit Computation of Quantiles

Quantiles are important in descriptive statistics because they display variation in samples of a statistical population without making any assumptions of the underlying statistical distribution. In the context of ensemble analysis and visualization, a priori knowledge about output data distributions is in general not known thus making these statistics highly useful. In fact, quantiles have long been used for data visualization and understanding, the earliest example being the Tukey boxplot [33], which is a method for graphically depicting scalar data distribution through their quartiles. In the present work, we are interested in the non-parametric characterization of the output variability of ensemble runs. We remark that these outputs not being scalar values, but ubiquitous multidimensional and time varying quantiles, their computation, visualization and interpretation represents a challenge.

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, online or even parallel [8]), 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.

Estimation procedures based on recursive algorithms are efficient techniques that are able to deal with large and voluminous samples. The Robbins-Monro algorithm [30] is such a procedure and has been developed in many situations and applications [34, 31, 35]. When the variable under study is not a simple scalar but a functional variable, the literature is much less abundant. One can cite the works [36, 37] where infinite dimensional Banach or Hilbert space are considered, but such developments remain preliminary. Our present work considers a high-dimensional vector of scalar variables (coming from the discretized spatio-temporal field) that are treated independently of each other. Dealing with the functional space where the spatio-temporal field lives remains a challenge and will be the topic of further works.

2.1. Empirical Quantile Estimator

Let us consider a N-sample $(Y_1, \dots, Y_N)$ of i.i.d. 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

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

The classical estimator of the α-quantile is the empirical quantile, based on the notion of order statistics [29]. Essentially, we associate with the sample $(Y_1, \dots, Y_N)$ the ordered sample $(Y_{(1)}, \dots, Y_{(N)})$ in which $Y_{(1)} \leq \dots \leq Y_{(N)}$. The empirical estimator then writes

$$\hat{q}_\alpha = Y_{(\lfloor \alpha N \rfloor + 1)}, \tag{3}$$

where $\lfloor x \rfloor$ is the integer part of x. When the probability density of Y is differentiable in y_α, a central limit theorem ($N \to \infty$) exists, shown for example in [29], which says that $\hat{q}_\alpha$ is an asymptotically normal estimator with variance $\alpha(1-\alpha)/[(N+2) f_Y^2(y_\alpha)]$.

2.2. Tuning a Robbins-Monro Estimator Algorithm

In the particular case of a quantile estimation, the Robbins-Monro estimator [30] consists in updating the quantile estimate at each new observation Y_n with the following rule

$$q_\alpha(n+1) = q_\alpha(n) - \frac{C}{n^\gamma} \left(\mathbb{1}_{Y_{n+1} \leq q_\alpha(n)} - \alpha \right) \ , \tag{4}$$

with $n = 1 \dots N$, $q_\alpha(n)$ the α-quantile estimate at the nth observation, $q_\alpha(1) = Y_1$ an independent realization of Y, $\mathbb{1}_x$ the indicator function, C

a strictly positive constant and $\gamma \in]0, 1]$ the step of the gradient descent of the stochastic algorithm. When $\gamma \in]0.5, 1]$ and under several hypotheses, this algorithm has been shown to be consistent and asymptotically normal. The Robbins-Monro algorithm has been introduced in the context of simulation model (scalar) output by [32] and has been used by [38] to solve the problem of conditional quantile estimation of stochastic simulation models.

In the equation (4), C and γ have to be chosen. In the following, we fix $C = 1$ and we concentrate our efforts to tune the γ values. Asymptotically ($N \to \infty$), a value $\gamma = 1$ is known to be optimal. However, in practical studies, N is not large enough to reach the asymptotic regime. For the type of engineering studies we consider (Y is the output of a costly computer code, see for example [25, 22]), $\alpha = 0.95$ and N is in the order of several hundreds of simulated values. To understand the algorithm behavior, a first numerical test is performed with $N = 1000$, $\alpha = 0.95$ and Y following a standard Gaussian distribution $\mathcal{N}(0, 1)$. $N = 1000$ is the typical order of magnitude of our studies (our application in Section 4 will use 3000 simulations).

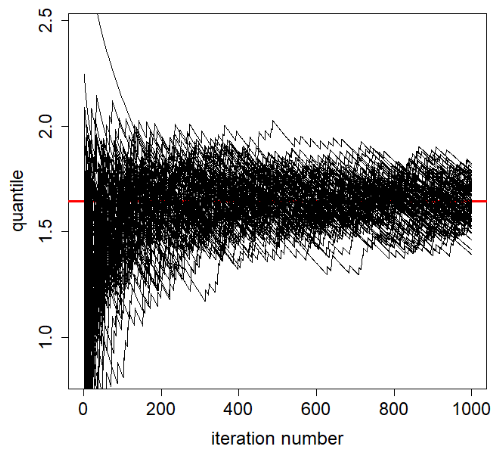
The Figure 1 shows 100 different and independent trajectories of the Robbins-Monro quantile estimates $q_{0.95}(n)$ for $n = 1, \ldots, 1000$ considering different values of γ. One can observe that:

- small values of γ (fig. 1a) induce large mixing of the quantile estimate during its evolution (all along the iterations). Then, the problem is a lack of stabilization because the final number of iterations could not be large enough;

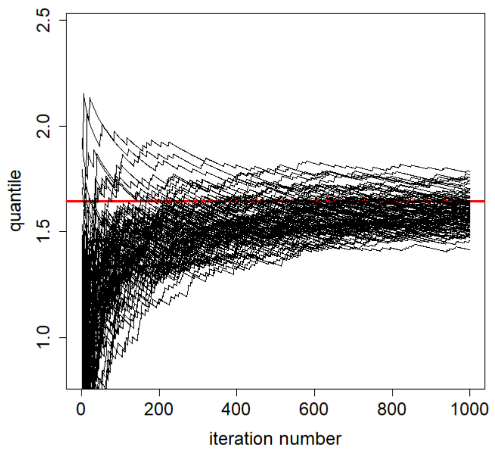
- larger values of γ (figs. 1b and 1c) induce small perturbations during the evolution of the quantile estimate. Then, the problem arrives when the initialization value ($n = 1$) is far from the quantile exact value because the evolution cannot correct it enough.

The Figure 1d shows the result of an algorithm (that will be detailed below) which consists in having small values of γ during the first iterations and large values of γ during the last iterations. A good convergence seems to have been reached (well-centered and not too dispersed distribution of the values around the exact value).

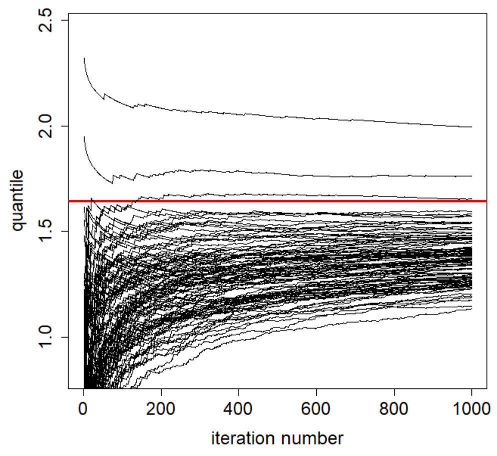
Another issue is our needs of robustness for the choice of γ. Indeed, we look for γ values which can work for different distributions of Y (which are unknown in practice). The problem is that good γ values for a certain type of probability distribution give bad results for another type of distribution (for



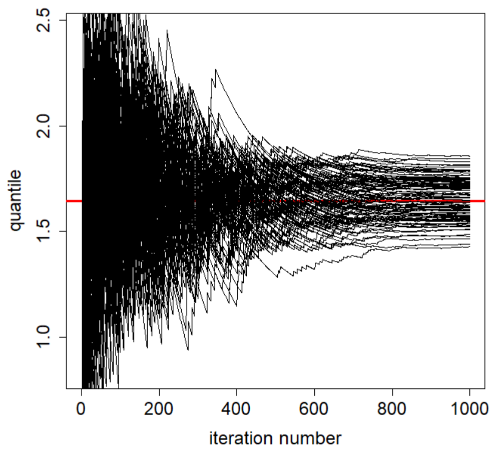
(a) $\gamma = 0.5$.



(b) $\gamma = 0.7$.



(c) $\gamma = 0.9$.



(d) Linear γ evolution.

Figure 1: For different choices of γ, simulation of 100 independent trajectories ($n = 1, \ldots, 1000 = N$) of the Robbins-Monroe estimation of the 0.95-quantile of a $\mathcal{N}(0,1)$ variable. The red horizontal line is the exact 0.95-quantile.

example, $\gamma = 0.6$ gives good results for a normal distribution and incorrect quantile estimates for a uniform one). Therefore, we introduce a new way to deal with the Robbins-Monro algorithm by defining γ as a function of n. The heuristic formula, inspired by a linear temperature profile choice in the simulated annealing algorithm, is the following:

$$\gamma(n) = 0.1 + 0.9\frac{n-1}{N-1} \,. \tag{5}$$

The idea is to have a 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.

Several numerical tests on different distributions of Y and simple analytical functions (where the true quantile can be known) have been performed to calibrate and validate this linear γ-profile. Figure 2 shows the results of four tests considering different probability density functions for Y, $\alpha = 0.95$ and $N = 1000$. Comparisons are made between the results given by the linear γ-profile and by different constant values for γ. We are particularly interested by knowing if we can obtain similar results with the Robbins-Monro estimator to those of the empirical estimator, which is our reference (because it is based on the storing of all sample values that is not the case with the Robbins-Monro estimators). For each estimator, the test consists in repeating 200 times the algorithm to obtain distributed values of the estimates. So, distribution-based comparisons are made.

For all the cases (figs. 2a, 2b, 2c and 2d), the distributions of estimates obtained by the linear γ-profile are well-centered, not too dispersed and rather close to the empirical estimator based results. This corresponds to the robustness we look for. We observe also that a good constant γ value in some cases is a really poor choice in other cases. For example, $\gamma = 0.5$ gives excellent results in figs. 2a and 2d and dramatic ones in figs. 2b and 2c, while $\gamma = 0.6$ gives excellent results in figs. 2b and 2c and poor ones in figs. 2a and 2d. All these results confirm the choice of the linear γ-profile that will be used in our practical study in the following.

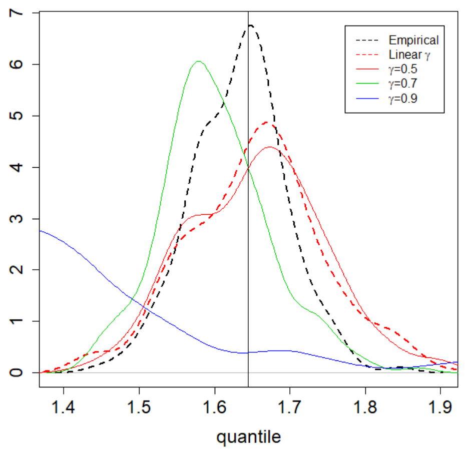
3. The Melissa Framework

We present in this section an overview of the Melissa framework (see [14] for a more detailed description).

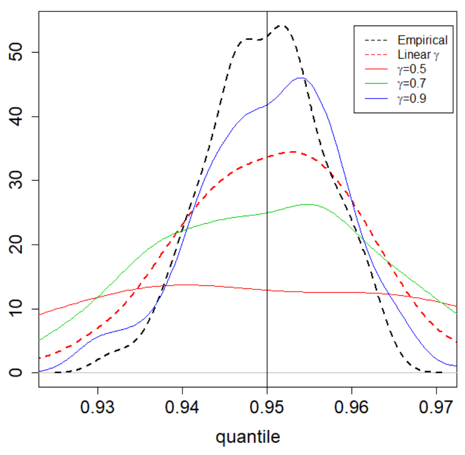
3.1. Melissa Architecture

Melissa (Modular External Library for In Situ Statistical Analysis) is an open source framework [1] that relies on a three tier architecture (Fig. 3). The

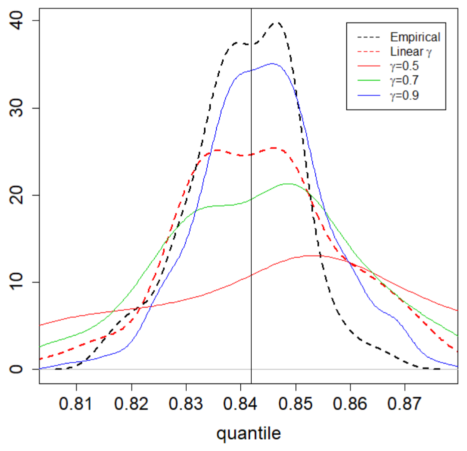
[1] `https://melissa-sa.github.io`



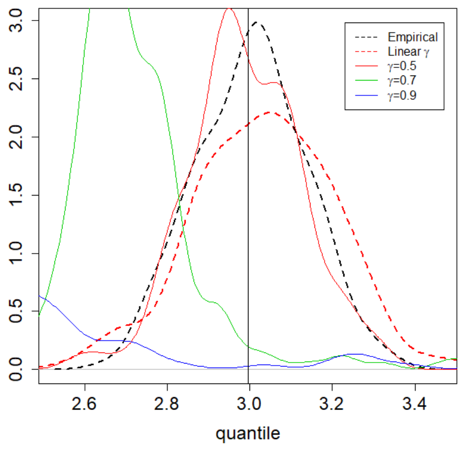
(a) Gaussian, $Y \sim \mathcal{N}(0,1)$.



(b) Uniform, $Y \sim \mathcal{U}[0,1]$.



(c) Triangular, $Y \sim \mathcal{T}(0, 0.5, 1)$.



(d) Exponential, $Y \sim \mathcal{E}(1)$.

Figure 2: For different choices of probability density functions of Y, probability densities of different estimators of the 0.95-quantile of Y: Empirical estimator and Robbins-Monro estimators (with $\gamma = 0.5$, $\gamma = 0.7$, $\gamma = 0.9$ and the linear profile). The vertical line is the exact 0.95-quantile.

Melissa clients are the parallel simulations, providing their outputs to the server. The *Melissa Server* aggregates the simulation results and updates iterative statistics as soon as a new result is available. *Melissa Launcher* interacts with the supercomputer batch scheduler and Melissa Server, for creating, launching, and supervising the server and clients.

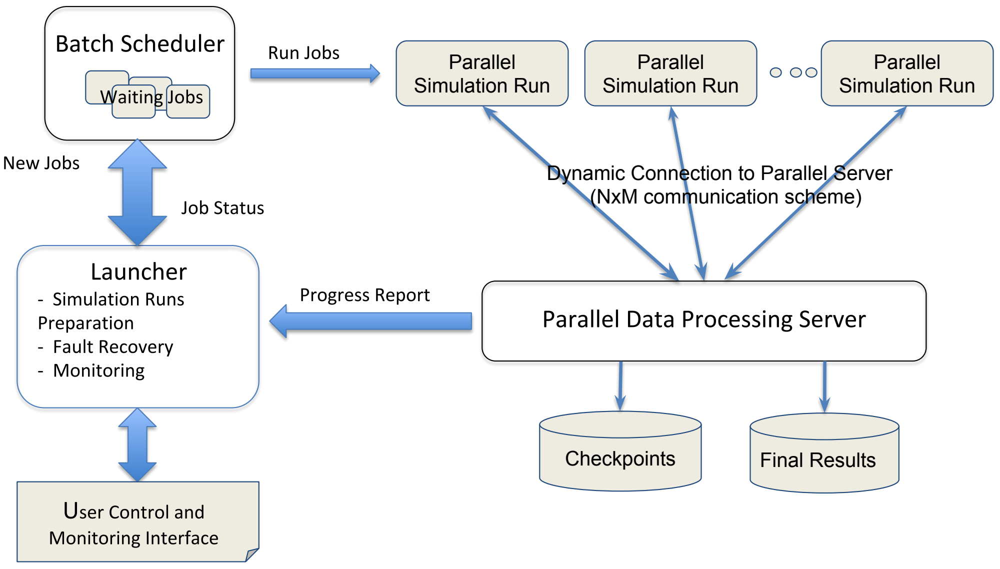


Figure 3: Melissa three tier architecture. The launcher oversees the execution in tight link with the batch scheduler. The job scheduler regulates the number of simulation jobs to run according to the machine availability, leading to an elastic resource usage. The parallel server, started first, process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

3.1.1. Melissa Server

Melissa Server is parallel and runs on several nodes. The number of nodes required for the server is mainly defined by its memory needs. The amount of memory needed for each computed statistic field is in the order of the size of the output field of one simulation (number of time-steps $\times$ the number of cells or points in the mesh). The simulation domain is evenly partitioned in space among the different server processes at starting time. Melissa uses its own static space partitioning of the data. This partitioning is different than the simulation partitioning, requiring a data redistribution between each client and the server. Updating the statistics is a local operation that requires neither communication nor synchronization between the server processes. The one-pass statistic algorithms allow the server processes to update their local statistics each time they receive a new data message, coming from any simulation, in any order.

3.1.2. Dynamic Connection to Melissa Server

When a simulation starts, it dynamically connects to Melissa Server. Each simulation process opens individual communication channels to each server process that needs data according to the data redistribution pattern. Every time new results are available, simulation processes send the results toward Melissa Server.

Melissa was designed to keep intrusion into the simulation code minimal. Melissa provides three 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 time-step, 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.

Melissa components are connected by ZeroMQ communication sockets. ZeroMQ is a multi-threaded library for the efficient asynchronous transfer of messages between a client and a server [39]. ZeroMQ bufferizes messages in a background thread both on the client and server side. This allows to regulate data transfers between clients and server without blocking the executions. Communications only become blocking when both buffers are full.

3.1.3. 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 various running job progress. 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. Next, each simulation is submitted to the batch scheduler in an independent job, making Melissa very elastic. Simulations can be submitted all at once or at a more regulated pace depending on the cluster policy for job submissions.

The launcher is the main actor of Melissa fault tolerance mechanism. It regularly checks for the jobs status, receives a heartbeat from the server, and is able to resubmit the server or the simulation jobs if needed.

For each use case, the user needs to provide a script for the Melissa Launcher to generate the parameter sets and to launch the simulations.

3.1.4. Fault Tolerance

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 check-pointing. 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 discards already processed messages.

Melissa Server regularly checkpoints. On failure, Melissa Launcher kills the running simulation jobs, restarts the server from the last checkpoint and the associated missing simulations.

Errors on Melissa Launcher are fail-safe: the running simulations proceed to completion with the server aggregating the incoming data. After a given time without any new message, the server checkpoints and stops.

The server check-pointing enables to manually restart any study for adding extra simulations. This is convenient for instance if the system killed the running experiment because it reached the wall-time limit, or simply because the user estimates that more simulation runs are required for improving the quality of the statistics.

4. Experiments

This section presents the large scale experiment illustrating the computation of one-pass quantiles with Melissa.

4.1. Fluid Simulation with Code_Saturne

The fluid numerical simulation is performed with *Code_Saturne* [40], 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 is based on a classical domain partitioning using MPI, with an optional second (local) level using OpenMP [41].

4.2. Use Case

Our implementation is validated on a fluid mechanics use case simulating a water flow in a tube bundle (Fig. 4). The solved scalar field represents a dye concentration and could be replaced by temperature or concentration of chemical compounds in actual industrial studies. The mesh is composed of 6002400 hexahedra. An ensemble study is generated by simulating the injection of a tracer or dye along the inlet, with 2 independent injection surfaces, each defined by three varying parameters:

1. dye concentration on the upper inlet,
2. dye concentration on the lower inlet,
3. width of the injection on the upper inlet,
4. width of the injection on the lower inlet,
5. duration of the injection on the upper inlet,
6. duration of the injection on the lower inlet.

All these parameters are uncertain and modelled as independent random variables. Their probability distribution functions are uniform on $[0.002, 0.1]$ for the two injection duration and uniform on $[0.1, 0.9]$ for the two concentrations and the two injection widths.

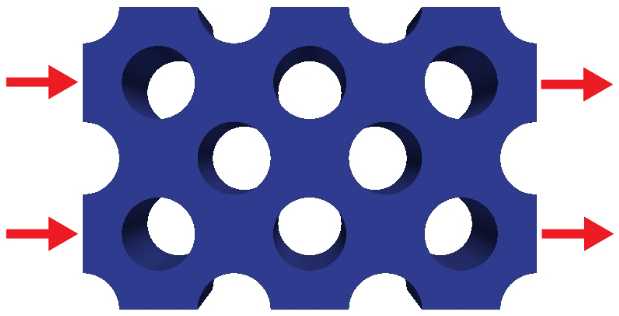


Figure 4: Use case: water flows from the left, between the tube bundle, and exits to the right.

To initialize our ensemble study, we first ran a single 1000 time-steps simulation, to obtain a steady flow. Assuming the resulting flow is independent of the scalar (dye concentration) values, we then use the final state of this simulation as the frozen velocity, pressure, and turbulent variable fields, on which we perform our experiment. This option allows solving only the convection-diffusion equation associated to the scalar, so simulations run much faster while generating the same amount of data. Each simulation consists of 100 time-steps on these frozen fields, with different parameter sets.

This study ran a total of 3000 simulations for computing all the ubiquitous percentiles, the 100-quantile, on the 6M hexahedra and 100 time-steps.

4.3. Performance

The experiment presented in this section run on a supercomputer called "Eole", ranked 128th at the top500.org of November 2016, when it was installed. At the time of submission of this article Eole ranks 460th, list of November 2018. Eole is composed of three kind of nodes:

1. 1164 standard nodes, each containing 2 Intel processors Xeon E5-2680v4 14C 2.4GHz and 128 GB of memory. Each processor contains 14 cores thus a standard node contains 28 cores.
2. 162 big memory nodes with the same architecture than the standard nodes but with upgraded memory, between 256 GB and 2 TB.
3. 64 graphical nodes equipped with Nvidia K80 GPUs.

All the nodes of Eole are connected by an Intel Omni-Path network. In our experiment, each *Code_Saturne* simulation runs on one node and it is parallelized on 28 cores. On the server side, Melissa Server must have enough memory to keep all the updated statistics and to queue the inbound messages from the simulations, and must compute the statistics fast enough to consume the data faster than they arrive. Otherwise, Melissa Server inbound message queue will end up full, eventually blocking the simulations. For this study, Melissa Server runs on 8 nodes (224 cores).

We have run the same experiment on Occigen2, a supercomputer that ranks 77th at the top500.org of November 2018. This test was performed to evaluate robustness. Because our validation was positive and the produced results are the same in both supercomputers, we thus just present the experimentation performed on EOLE.

Fig. 5 presents a plot showing the temporal evolution of the study. On the horizontal axis time evolves, in minutes, from left to right. Total execution time was 210 minutes (3.5 hours). The plot shows the evolution of the number of simultaneous fluid dynamics simulations performed over time. This is equivalent to the number of cores because each simulation runs, for this experiment, of 28 cores. We fixed a limit of 40 simultaneous simulations (1120 cores) but we see that this limit is seldom achieved because the scheduler of Eole allocates less resources for our study. This is a common situation faced by most users of supercomputers, which being shared resources are not available at will. We see how the elastic nature of Melissa allows the study to adapt to the available resources, sometimes using less resources, sometimes the full capacity (40 simultaneous simulations in this case) and even not running any simulation for some period of time, as we can see around time 200 minutes.

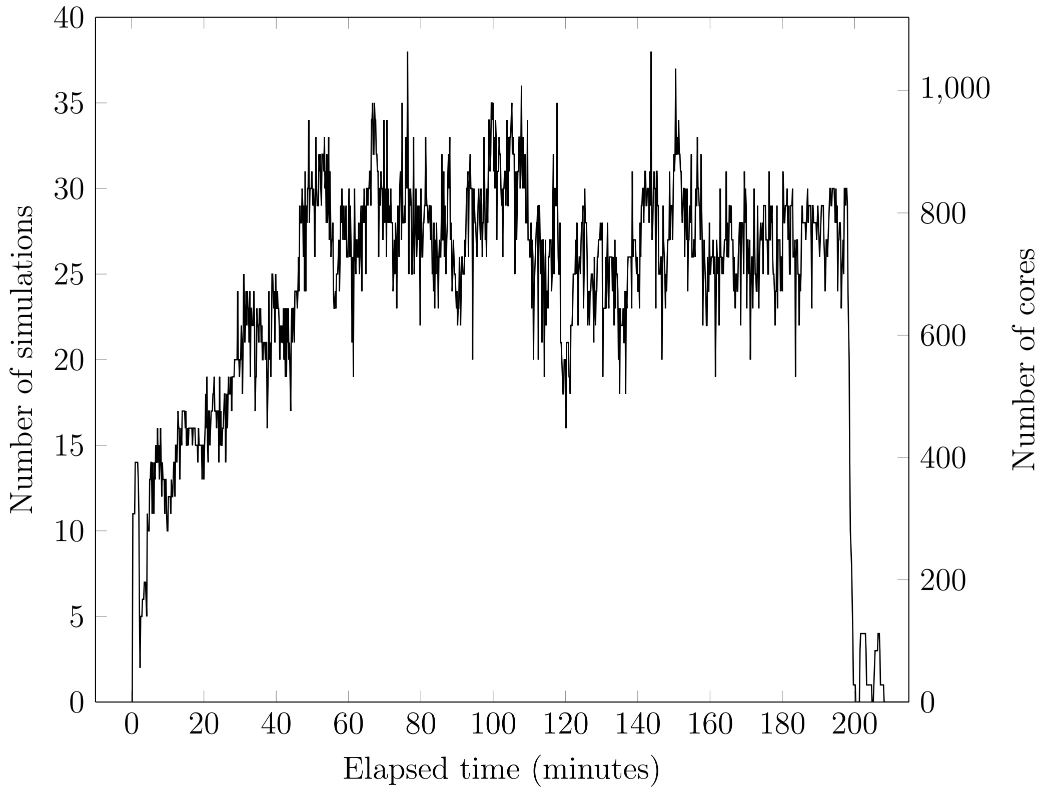


Figure 5: Evolution of the number of simultaneously running simulations (equivalent to the number of cores) during the execution of our use case running a total of 3000 simulations.

During this study, Melissa Server processed 11 TB of data coming from the simulations. In a classical study, all these data would be written to the file-system, and read back to compute the quantiles. This would not have been possible on Eole or Occigen simply due to the storage capacity (quota) being limited per user.

5. Ubiquitous Quantile Visualization

This section presents the quantiles computed during the experiments. Figure 6 presents six spatial maps extracted from the ubiquitous quantiles. By use of the Open-Source visualization tool ParaView, we have chosen a time-step and performed a slice on a mid-plane aligned with the direction of the fluid. The chosen time-step belongs to the last temporal part of the simulation (80^{th} time-step over 100). This section focuses on the interpretation of the computed percentiles. However, the system can compute any other kind of quantile.

On the four top panels of Figure 6, Fig. 6a, 6b, 6c and 6d, we present the 75^{th}, 95^{th}, 25^{th}, 5^{th} percentiles, respectively. On the two bottom panels (Fig. 6e and Fig. 6f) the inter-percentile ranges containing 50% and 90% of the samples are shown. Inter-percentile ranges are easily computed from percentiles by substraction: the 50% inter-percentile range corresponds to the 75^{th} percentile minus the 25^{th} percentile; the 90% inter-percentile range corresponds to the 95^{th} percentile minus the 5^{th} percentile. In Figure 6 each column shows an inter-percentile 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 percentile maps such as 95^{th}, Fig. 6b, give an idea of the distribution of the upper bounds of all simulations. In our use case, we can assess which spatial areas contain low quantities of dye. Areas colored in blue necessarily contain low dye concentrations for any simulation in the ensemble study. Extreme low percentile maps, such as 5^{th} has also a direct interpretation in the opposite sense.
2. Inter-percentile range maps such as Fig. 6e or 6f are maps that show the spatial variability of statistical dispersion. Indeed, scalar inter-percentile 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. Visualizing a map of such a measure of dispersion allows to understand how the data distribution spatially concentrates. In our use case, the low percentile maps used to calculate the inter-percentile 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.

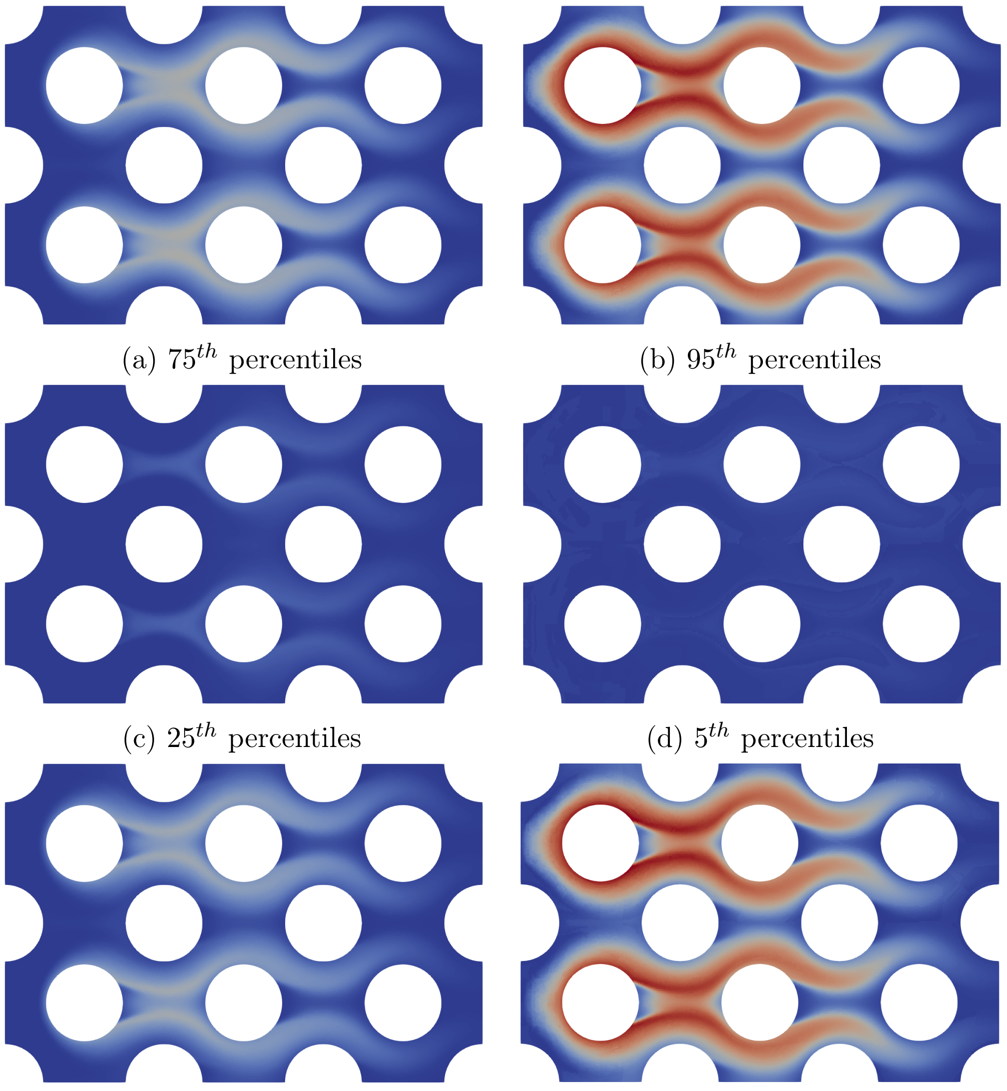
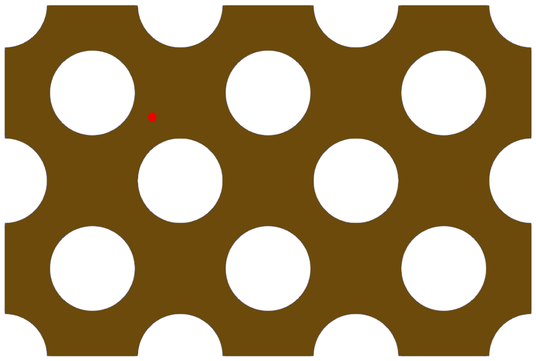
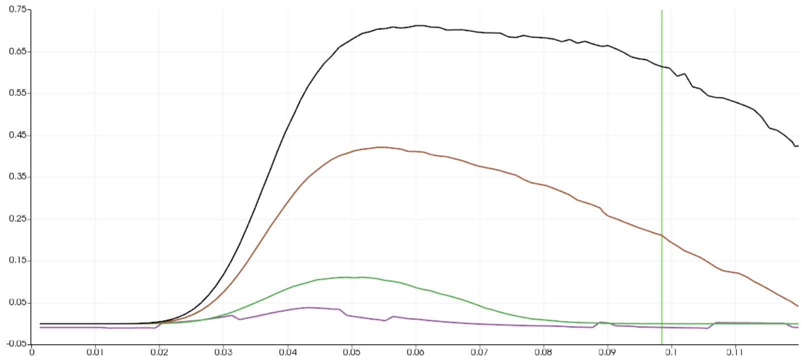

(a) 75th percentiles (b) 95th percentiles (c) 25th percentiles (d) 5th percentiles (e) 50% inter-percentile range (f) 90% inter-percentile range

Figure 6: Percentiles and inter-percentile range maps on a slice of the mesh at 80th time-step. The four top panels correspond to the percentiles while the two bottom panels correspond to inter-percentile ranges. All maps share the same scale.

The maps shown in Figure 6 are static and 2D but we recall that we calculate ubiquitous percentiles, thus 3D and time dependent data is available. The Figure 7 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 Fig. 7b this evolution is plotted for the 95th, 75th, 25th and 5th percentiles. The vertical line indicates the position of the current time step (80th 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.



(a) Probe position



(b) Evolution of the percentiles for the dye concentration (vertical axis) over time (horizontal axis)

Figure 7: A probe in a cell of the mesh allows an extraction of the temporal evolution of percentiles at a specific spatial location.

Fig. 7b can be seen as the evolution of a Tukey boxplot [33] over time. In fact, the 25th and 75th percentiles correspond to the 1st and 3rd quartiles thus delimit the central box of the plot, while the 5th and 95th 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, Fig. 8 shows a different representation of the evolution of the dye concentration at a fixed probe (the same than in Fig. 7). 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 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 Fig. 7 and Fig. 8 for any location on the simulation domain.

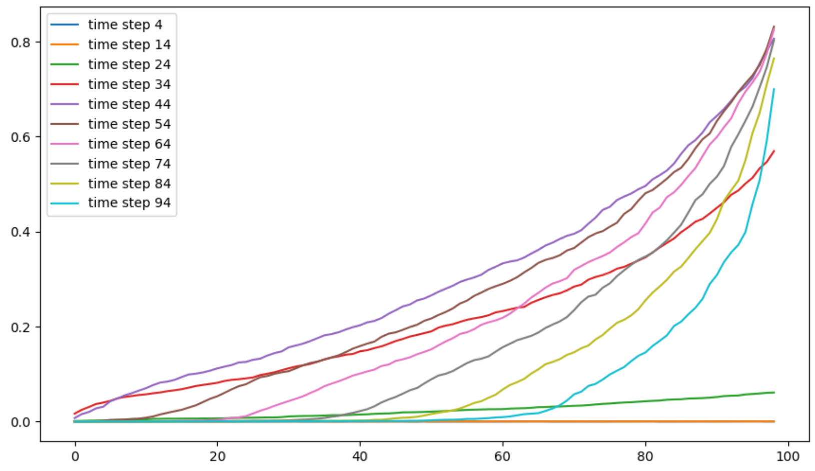


Figure 8: Percentile functions of the dye concentration at different time steps of the simulation. Vertical axis represents dye concentration. Horizontal axis represents percentiles. Each curve corresponds to different time steps of the simulation. All curves have been extracted form the probe position shown in Fig. 7a.

6. Conclusion

Exascale machines will shortly become a reality. But so far only very few applications are able to take benefit of the level of parallelization these machines provide. Ensemble runs and uncertainty quantification approaches may require to execute from thousands to millions 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 these intermediate data are saved to disk. An alternative consists in processing these data in transit as proposed by the Melissa framework. But this approach requires one-pass data processing algorithms to limit the amount of memory needed.

This paper proposes the computation of quantiles by use of a parallel one-pass strategy based on a new robust version of the stochastic quantile algorithm of Robbins-Monro [30]. The algorithm is experimented at large scale with the Melissa in transit, elastic and fault-tolerant processing framework. On a fluid dynamics application case, 3000 simulations on a $6M$ cell mesh allow to compute all spatio-temporal percentiles at full resolution, saving 11TB of intermediate storage thanks to Melissa. Because the I/O bottleneck has been bypassed, ubiquitous spatio-temporal maps of percentiles and inter-percentile based intervals have been successfully visualized, revealing their interest for the interpretation by the users. In addition to quantiles, Melissa currently supports average, standard deviation, skewness, kurtosis, minimum, maximum, threshold exceedance and Sobol' indices.

Several ways of improvement are identified on the iterative quantile estimation algorithm that we use. First, the constant C and the γ-profile has been defined in an heuristic manner and a theoretical consolidation of such a choice seems necessary. Different versions of the Robbins-Monro algorithm also exist (for example the averaged one) and should be tested. Second, the user of the method should avoid to fix a priori a total number of runs. A more interesting approach would iteratively give an estimate of the quantile, and then stop when a sufficient precision (defined by the user) is achieved. This precision is not theoretically known in a non-asymptotic context but approximate results are maybe accessible and have to be tested.

Last, the quantiles of the spatio-temporal outputs have been computed cell per cell and time-step per time-step. The interpretation of this ubiquitous quantiles (for instance in the form of static spatial maps, temporal probes or videos) is much richer than the traditional predefined probe-based or sub-sampled approaches. However, the functional space where the spatio-temporal field lives has not be estimated. Dealing with this space (as in [37]), the ubiquitous quantile estimates would conserve the geometrical and temporal structure of the ensemble run study.

7. Acknowledgement

This work was partly funded in the Programme d'Investissements d'Avenir, grant PIA-FSN2-Calcul intensif et simulation numérique-2-AVIDO. This work was granted access to the HPC resources of CINES under the allocation 2018-A0040610366 made by GENCI.

8. References

[1] R. Smith, Uncertainty quantification, SIAM, 2014.

[2] M. Baudin, A. Dutfoy, B. Iooss, A. Popelin, Open TURNS: An industrial software for uncertainty quantification in simulation, in: R. Ghanem, D. Higdon, H. Owhadi (Eds.), Springer Handbook on Uncertainty Quantification, Springer, 2017, pp. 2001–2038.

[3] R. Ghanem, D. Higdon, H. Owhadi (Eds.), Springer Handbook on Uncertainty Quantification, Springer, 2017.

[4] B. Welford, Note on a method for calculating corrected sums of squares and products, Technometrics 4 (3) (1962) 419–420.

[5] T. F. Chan, G. H. Golub, R. J. LeVeque, Updating formulae and a pairwise algorithm for computing sample variances, in: COMPSTAT 1982 5th Symposium held at Toulouse 1982, Springer, 1982, pp. 30–41.

[6] T. Finch, Incremental calculation of weighted mean and variance, Tech. rep., University of Cambridge (2009).

[7] J. Bennett, R. Grout, P. Pébay, D. Roe, D. Thompson, Numerically stable, single-pass, parallel statistics algorithms, in: Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on, IEEE, 2009, pp. 1–8.

[8] P. Pébay, Formulas for robust, one-pass parallel computation of covariances and arbitrary-order statistical moments, Sandia Report SAND2008-6212, Sandia National Laboratories 94.

[9] P. Pébay, D. Thompson, J. Bennett, A. Mascarenhas, Design and performance of a scalable, parallel statistics toolkit, in: Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on, IEEE, 2011, pp. 1475–1484.

[10] J. C. Bennett, V. Krishnamoorthy, S. Liu, R. W. Grout, E. R. Hawkes, J. H. Chen, J. Shepherd, V. Pascucci, P.-T. Bremer, Feature-based statistical analysis of combustion simulation data, IEEE transactions on visualization and computer graphics 17 (12) (2011) 1822–1831.

[11] S. Lakshminarasimhan, J. Jenkins, I. Arkatkar, Z. Gong, H. Kolla, S.-H. Ku, S. Ethier, J. Chen, C.-S. Chang, S. Klasky, 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, ACM, 2011, p. 31.

[12] J. C. Bennett, H. Abbasi, P.-T. Bremer, R. Grout, A. Gyulassy, T. Jin, S. Klasky, H. Kolla, M. Parashar, V. Pascucci, 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, IEEE, 2012, pp. 1–9.

[13] L. Gilquin, E. Arnaud, C. Prieur, H. Monod, Recursive estimation procedure of Sobol' indices based on replicated designs, Preprint, http://hal.univ-grenoble-alpes.fr/hal-01291769.

[14] T. Terraz, A. Ribes, Y. Fournier, B. Iooss, B. Raffin, 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.

[15] T. Terraz, B. Raffin, A. Ribes, Y. Fournier, In situ statistical analysis for parametric studies, in: Proceedings of the Second Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization, ISAV2016, ACM, 2016.

[16] T. Tu, C. A. Rendleman, D. W. Borhani, R. O. Dror, J. Gullingsrud, M. O. Jensen, J. L. Klepeis, P. Maragakis, P. Miller, K. A. Stafford, D. E. Shaw, A Scalable Parallel Framework for Analyzing Terascale Molecular Dynamics Simulation Trajectories, in: Conference on Supercomputing, 2008, pp. 56:1–56:12.

[17] M. Dorier, G. Antoniu, F. Cappello, M. Snir, L. Orf, Damaris: How to Efficiently Leverage Multicore Parallelism to Achieve Scalable, Jitter-free I/O, in: CLUSTER - IEEE International Conference on Cluster Computing, IEEE, 2012.

[18] F. Zheng, H. Zou, G. Eisnhauer, K. Schwan, M. Wolf, J. Dayal, T. A. Nguyen, J. Cao, H. Abbasi, S. Klasky, N. Podhorszki, H. Yu, FlexIO: I/O middleware for Location-Flexible Scientific Data Analytics, in: IPDPS'13, 2013.

[19] M. Dreher, B. Raffin, A Flexible Framework for Asynchronous In Situ and In Transit Analytics for Scientific Simulations, in: 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CC-Grid'14), IEEE, Chicago, 2014.
URL `https://hal.inria.fr/hal-00941413`

[20] E. de Rocquigny, N. Devictor, S. Tarantola (Eds.), Uncertainty in industrial practice, Wiley, 2008.

[21] J. Morio, M. Balesdent, Estimation of rare event probabilities in complex aerospace and other systems, Woodhead Publishing, 2016.

[22] B. Iooss, A. Marrel, Advanced methodology for uncertainty propagation in computer experiments with large number of inputs, Nuclear Technology, In press.

[23] P. W. Glynn, Importance sampling for monte carlo estimation of quantiles, in: Proceedings of Second International Workshop on Mathematical Methods in Stochastic Simulation and Experimental Design, Publishing House of Saint Petersburg University, 1996, pp. 180–185.

[24] T. Hesterberg, B. Nelson, Control variates for probability and quantile estimation, Management Science 44 (1998) 1295–1312.

[25] C. Cannamela, J. Garnier, B. Iooss, Controlled stratification for quantile estimation, Annals of Apllied Statistics 2 (2008) 1554–1580.

[26] A.-L. Popelin, B. Iooss, 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.

[27] A. Ribés, J. Pouderoux, A.-L. Popelin, B. Iooss, Visualizing statistical analysis of curves datasets in Paraview, in: 2014 IEEE Conference on Visual Analytics Science and Technology (VAST), Paris, France, 2014.

[28] S. Nanty, C. Helbert, A. Marrel, N. Pérot, C. Prieur, Uncertainty quantification for functional dependent random variables, Comput. Stat. DOI 10.1007/s00180-016-0676-0.

[29] H. David, H. Nagaraja, Order statistics, 3rd Edition, Wiley, New-York, 2003.

[30] H. Robbins, S. Monro, A stochastic approximation method, The Annals of Mathematical Statistics 22 (1951) 400–407.

[31] M. Duflo, Random iterative models, Springer-Verlag Berlin Heidelberg, 1997.

[32] M. Kohler, A. Krzyzak, H. Walk, Nonparametric recursive quantile estimation, Statistics and probability Letters 93 (2014) 102–107.

[33] J. W. Tukey, Exploratory data analysis, Vol. 2, Reading, Mass., 1977.

[34] A. Benveniste, M. Métivier, P. Priouret, Adaptive algorithms and stochastic approximation iterative models, Springer-Verlag, New-York, 1990.

[35] H. Kushner, G. Yin, Stochastic approximation and recursive algorithms and applications, 2nd Edition, Springer-Verlag Berlin Heidelberg, 2003.

[36] S. Smale, Y. Yao, Online learning algorithms, Foundations of Computational Mathematics 6 (2006) 145–170.

[37] H. Cardot, P. Cénac, P.-A. Zitt, Efficient and fast estimation of the geometric median in hilbert spaces with an averaged stochastic gradient algorithm, Bernoulli 19 (2013) 18–43.

[38] T. Labopin-Richard, F. Gamboa, A. Garivier, Conditional quantile sequential estimation for stochastic codes, Preprint, https://arxiv.org/abs/1508.06505.

[39] P. Hintjens, ZeroMQ, Messaging for Many Applications, O'Reilly Media, 2013.

[40] F. Archambeau, N. Méchitoua, M. Sakiz, Code_Saturne: a finite volume code for the computation of turbulent incompressible flows, International Journal on Finite Volumes 1.

[41] Y. Fournier, J. Bonelle, C. Moulinec, Z. Shang, A. Sunderland, J. Uribe, Optimizing code_Saturne computations on petascale systems, Computers & Fluids 45 (1) (2011) 103 – 108.