Institut für Numerische Simulation
Rheinische Friedrich-Wilhelms-Universität Bonn
Friedrich-Hirzebruch-Allee 7 • 53115 Bonn • Germany
phone +49 228 73-69828 • fax +49 228 73-69847
www.ins.uni-bonn.de

S. Rezaeiravesh, C. Gscheidle, A. Peplinski,
J. Garcke, P. Schlatter

In-situ Estimation of Time-averaging Uncertainties in Turbulent Flow Simulations

INS Preprint No. 2204

2022

In-situ Estimation of Time-averaging Uncertainties in Turbulent Flow Simulations

S. Rezaeiravesh[a,b,c,*], C. Gscheidle[d,**], A. Peplinski[b,c], J. Garcke[d,e], P. Schlatter[f,b,c]

[a]*Department of Fluids and Environment, The University of Manchester, M139PL Manchester, UK*
[b]*SimEx/FLOW, Engineering Mechanics, KTH Royal Institute of Technology, 10044 Stockholm, Sweden*
[c]*Swedish e-Science Research Centre (SeRC), Stockholm, Sweden*
[d]*Fraunhofer SCAI, 53757 Sankt Augustin, Germany*
[e]*Institute for Numerical Simulation, University of Bonn, 53115 Bonn, Germany*
[f]*Institute of Fluid Mechanics (LSTM), Friedrich–Alexander Universität Erlangen–Nürnberg (FAU), 91058 Erlangen, Germany*

Abstract

The statistics obtained from turbulent flow simulations are generally uncertain due to finite time averaging. The techniques available in the literature to accurately estimate these uncertainties typically only work in an offline mode, that is, they require access to all available samples of a time series at once. In addition to the impossibility of online monitoring of uncertainties during the course of simulations, such an offline approach can lead to input/output (I/O) deficiencies and large storage/memory requirements, which can be problematic for large-scale simulations of turbulent flows. Here, we designed, implemented and tested a framework for estimating time-averaging uncertainties in turbulence statistics in an in-situ (online/streaming/updating) manner. The proposed algorithm relies on a novel low-memory update formula for computing the sample-estimated autocorrelation functions (ACFs). Based on this, smooth modeled ACFs of turbulence quantities can be generated to accurately estimate the time-averaging uncertainties in the corresponding sample mean estimators. The resulting uncertainty estimates are highly robust, accurate, and quantitatively the same as those obtained by standard offline estimators. Moreover, the computational overhead added by the in-situ algorithm is found to be negligible. The framework is completely general and can be used with any flow solver and also integrated into the simulations over conformal and complex meshes created by adopting adaptive mesh refinement techniques. The results of the study are encouraging for the further development of the in-situ framework for other uncertainty quantification and data-driven analyses relevant not only to large-scale turbulent flow simulations, but also to the simulation of other dynamical systems leading to time-varying quantities with autocorrelated samples.

Keywords: Uncertainty quantification, Time-averaging uncertainty, In-situ estimation, Turbulent flows, Autocorrelation.

1. Introduction

Turbulent fluid flows are fundamentally unsteady and contain vortical structures of a wide range of spatial and temporal scales. For numerical simulation of turbulent flows, various approaches have been developed, see e.g. Sagaut et al. [28]. Due to their capabilities in accurately capturing the physics of turbulent flows, scale-resolving approaches such as large-eddy simulation (LES) and direct numerical simulation (DNS) are

*Principal Corresponding Author
**Corresponding Author
Email addresses: saleh.rezaeiravesh@manchester.ac.uk (S. Rezaeiravesh), christian.gscheidle@scai.fraunhofer.de (C. Gscheidle), adam@mech.kth.se (A. Peplinski), jochen.garcke@scai.fraunhofer.de (J. Garcke), philipp.schlatter@fau.de (P. Schlatter)

of particular interest for both academic and industrial flows. It is recalled that LES, mostly, and DNS, fully, resolve the flow structures in time and space. A main challenge when applying these approaches is the required excessive computational cost, which would become prohibitive for wall-bounded turbulent flows at high-Reynolds numbers [6]. For such flows, which appear in many engineering applications, the computational cost of the scale-resolving simulations is driven by the requirement of accurately resolving the inner part of the turbulent boundary layers (TBL) [5, 11]. In recent decades, the progress in high-performance computing (HPC) technologies has made it possible to employ scale-resolving turbulence simulations such as LES and DNS at higher Reynolds numbers and more complex flows.

The HPC developments have, however, led to new requirements and aspects to consider in the design of the next generation of CFD (computational fluid dynamics) software. In particular, the NASA 2030 vision [31] has listed a technology development roadmap and specified the readiness level of various technologies. Among such are the uncertainty quantification (UQ) techniques for assessing the reliability and accuracy of the CFD simulations' outcomes. Among others, the uncertainties in CFD simulations can be due to the numerical settings, programming, turbulence modeling, initial/boundary data, and finite time-averaging [19, 32, 23, 26]. The focus of the present study is on the latter that is also known as statistical or sampling uncertainty, appearing due to the finite number of samples considered when computing the time-averaged quantities and turbulence statistics. In practice, after the turbulent flow becomes statistically stationary, sample mean estimators (SME) are evaluated by averaging the samples which are autocorrelated by nature. Different techniques for estimating the uncertainty in SMEs of turbulent flow quantities have been used, see [23, 27, 36, 14], a short review of which is given in Section 2. As extensively discussed in Ref. [36], the hyperparameters appearing in each of these techniques have to be properly adopted in order to avoid any bias in the estimated time-averaging uncertainties.

In large-scale CFD simulations, there is an increasing gap between the amount of data that is generated during the runtime and what can actually be stored to disk. For current HPC systems, this gap due to the limitations in the data input/output (I/O) can be as large as up to four orders of magnitudes, even for highly parallel systems [13]. To overcome this limitation, workflows for in-situ visualizations have been set up to export compressed pictures of the flow field during runtime, see [1] for an in-situ visualization workflow with Nek5000 [9]. A common technique to visualize turbulence are iso-surfaces of the so-called Q-criterion exported for each time-step, which can later be integrated into a video. Even though the resolution and the number of extracted pictures can be increased significantly by an in-situ visualization workflow, a quantitative analysis based on these will not be possible after the end of the simulation. Moreover, in the field of data analytics and machine learning, in-situ algorithms, also known as updating, streaming or incremental, have recently gained attention for two reasons. First, there are cases where not all of the training data fits into memory. Therefore, the data processing is usually performed out-of-memory, where only one sample or a small batch of samples is loaded and processed at a time to update the model. This also applies for scenarios where data is constantly being produced, e.g. by a simulation or measurements of a system. But here, as soon as new data is available it can be processed step by step in an in-situ fashion. Note that when processing data in batches, often the accuracy of the results depends on the batch size, e.g. when computing streaming singular-value decomposition (SVD) [15]. In these cases, a good balance between memory usage on the HPC system and the accuracy of the results is required. Second, when working in-situ one can exploit the intermediate data analysis results or measurements for an online monitoring of the system. Both motivations also apply to CFD simulations. There, very large amounts of data are generated that cannot be held in memory at once. An in-situ analysis can thus significantly increase the accuracy and accessibility to the data analysis results. Besides, based on an analysis of intermediate simulation results, we can build a monitoring system that provides criteria to abort or steer the simulation and eventually save computing time and resources.

In the literature, studies for the evaluation of time-averaging uncertainties have essentially been based on the assumption that the uncertainty estimators have access to the whole set of time samples, see [23, 27, 36]. In practice, this requires that for each quantity, the generated samples are stored on disk and then loaded to the memory. These would lead to both severe memory and computational deficiencies, especially for large-scale simulations. To rectify these issues, the present study provides the necessary tools in addition to a framework for an in-situ evaluation of the uncertainties in sample mean estimators for autocorrelated

samples. These include updating estimators for the SME and associated uncertainties, as well as an interface between the CFD solver and UQ module. The memory requirements will be minimized in approaches where the UQ estimator still needs to keep information in memory at runtime.

The remainder of the paper is organized as follows. Section 2 introduces the problem of estimating time-averaging uncertainty in SMEs of turbulence time series and proposes a formula for obtaining a smooth model for the autocorrelation function (ACF) of such time series. The focus of Section 3 is on updating UQ algorithms. Section 3.1 reviews the updating formulae for an in-situ estimation of the sample mean and variance of a time series. The formulae are then used in Section 3.2 to derive expressions for an in-situ estimation of time-averaging uncertainty in SMEs based on batch-based methods. The updating method for estimating the time-averaging uncertainties proposed in the present study is discussed in Section 3.3, followed by the description of the workflow and software that implements it in Section 4. Then, in Section 5, the flow solvers and use cases to which the framework is applied are described. In Section 6, the results are presented and thoroughly discussed. This includes the assessment of the accuracy, sensitivity and robustness of the new techniques proposed in Sections 2 and 3.3, as well as the assessment of the computational efficiency of the framework developed in Section 4 when applied to the use cases. Summary and conclusions are provided in Section 7.

2. Uncertainty in a Sample Mean Estimator (SME)

In practice when only one simulation (realization) of a physical process is performed, the true expectation or mean value of a quantity x belonging to the process cannot be obtained. Instead, for each realization of the process, a set of time samples for x is obtained from which a sample mean estimate can be computed. Let $\mathbf{x}$ denote a time series sample of size n, i.e. $\mathbf{x} = \{x_i\}_{i=1}^{n}$, where $x_i = x(t_i)$. Henceforth, we assume the times t_i are equi-spaced. The sample mean estimator (SME) for the time series reads as

$$\hat{\mu} := \hat{\mathbb{E}}[x] = \frac{1}{n}\sum_{i=1}^{n} x_i \,. \tag{1}$$

Note that throughout the text, estimated values/estimators are represented with the overhat symbol $\hat{\cdot}$. The SME is unbiased, see e.g. [34], and converges to the true but unobserved expectation of x, i.e. $\mu := \mathbb{E}[x]$. Furthermore, the SME has a Gaussian distribution:

$$\hat{\mu} \sim \mathcal{N}\left(\mu, \sigma^2(\hat{\mu})\right) \,, \tag{2}$$

where $\sigma(\hat{\mu})$ is the standard deviation and hence a measure of uncertainty of $\hat{\mu}$. For many processes including flow turbulence, the time samples of any flow variable are autocorrelated over a generally unknown number of time lags. For autocorrelated samples in $\mathbf{x}$, an analytical expression can be derived for the variance of $\hat{\mu}$, see e.g. [34]:

$$\mathbb{V}[\hat{\mu}] = \sigma^2(\hat{\mu}) = \frac{1}{n}\left[\gamma_0 + 2\sum_{m=1}^{(n-1)}\left(1-\frac{m}{n}\right)\gamma_m\right] \,, \tag{3}$$

where γ_m is the autocovariance of x at lag m. Trivially, this expression can be written in terms of the autocorrelations $\rho_m = \gamma_m/\gamma_0$ with γ_0 denoting the variance of x. The sample-estimated autocovariances are obtained from

$$\hat{\gamma}_m = \widehat{\mathrm{cov}}(x_i, x_{i-m}) = \frac{1}{n-m}\sum_{i=1}^{n-m} x_i' x_{i-m}' \,, \tag{4}$$

where $x_i' = x_i - \hat{\mathbb{E}}[x]$. Plugging $\hat{\gamma}_m$ into Eq. (3) leads to estimates for $\hat{\sigma}^2(\hat{\mu})$, which can, however, be inaccurate noting there are non-vanishing oscillations of $\hat{\gamma}_m$ at higher lags for any finite number of time samples (sample size).

In the literature, two main approaches have been suggested to circumvent the issues arising by the use of $\hat{\gamma}_m$ in Eq. (3). In the batch-based approaches, the original samples are divided into a set of batches and

then one works with the mean values of the samples in each batch without using Eq. (3). Depending on how particularly the batch means are used, different approximations of $\hat{\sigma}(\hat{\mu})$ are achieved. For a review of the batch-based uncertainty estimators as well as the more efficient Batch-Means Batch-Correlation (BMBC) algorithm, see [27]. In Section 3.2, we provide a brief overview of the batch-based methods relevant to the present study.

In the second type of approaches, Eq. (3) is directly evaluated by introducing a modeled autocorrelation function (ACF) or autocovariance function (ACovF) that is smooth for all lags. As first applied to turbulent flow simulations by Oliver et al. [23], an autoregressive model (ARM) can be fitted to the original samples $\mathbf{x}$, and then the estimated ARM coefficients are used to fit a smooth power-law ACF for $m \in [0, \infty)$, see Appendix A. This approach is believed to be the most accurate method of estimating uncertainty in the SME in turbulent flows conditioned on adopting appropriate values for the hyperparameters involved. The latter include the order of the ARM and the set of sample-estimated autocorrelations to construct a smooth ACF model, see Xavier et al. [36]. The described method is not of interest in the present study, because it requires the entire time-series data to be in memory for fitting the ARM and is therefore unsuitable for an in-situ implementation. Instead, we can rely on ad-hoc algebraic functions to create smooth models for the ACFs, meaning that fitting an ARM to the time samples is skipped. Along this line of thought, in the studies mainly relevant to atmospheric turbulence the following function proposed in Ref. [14] is widely used to model the ACFs:

$$\rho(m) = \exp(-a\, m)\,, \tag{5}$$

for $m = 0, 1, 2, \ldots$, where $1/a$ is the turbulence integral time-scale. There have been other forms of functions for modeling ACF, see e.g. [29] and the references therein. However, as discussed in Section 6.1, the model function (5) may not be appropriate for data obtained from wall-bounded turbulence simulations. Instead, we propose to use the slightly modified function

$$\rho(m) = a \exp(-b\, m) + (1 - a) \exp(-c\, m)\,, \tag{6}$$

to model ACFs of turbulence time series. Here, $m \geq 0$ is the time lag (integer), and parameters b and c are strictly positive. Obviously, $\rho(0) = 1$ and $\rho(m \to \infty) \to 0$. In comparison to Eq. (5), the suggested function has two more parameters to estimate. To construct a smooth ACF model using Eq. (6), a set of training sample-estimated autocorrelations $\{\rho_i\}_{i=1}^{n_{\text{train}}}$ at lags $\{m_i\}_{i=1}^{n_{\text{train}}}$ are used. As discussed in Section 6.1, as one of its main advantages, the model function (6) is flexible in terms of how the training lags are selected. To estimate the model parameters a, b, and c, a non-linear least-squares method can be employed. Hereafter, we refer to the estimation of the $\sigma(\hat{\mu})$ using the ACF model (6) as the *MACF method*.

3. Updating UQ Algorithms

To the author's knowledge, there has not been any prior study addressing the in-situ formulation of neither batch-based methods nor estimators relying on the modeling of the ACF function. In what follows, first the basic definitions for constructing updating uncertainty estimators are provided. Then, updating versions of the batch-based methods as well as the proposed MACF method are presented. In all the following derivations, the samples are assumed to be equidistant in time. For the special case of statistically stationary turbulent flows, the variable x in the following formulations can be any of the flow variables at any spatial location inside the flow domain.

3.1. Updating sample mean and variance estimators

Following Chan et al. [4], the sample mean of a time series considering the i-th to the j-th time samples is defined as

$$M_{i,j} = \frac{1}{(j - i + 1)} \sum_{k=i}^{j} x_k\,. \tag{7}$$

Correspondingly, the sample variance estimation is given by $S_{i,j}/(j-i+1)$ where,

$$S_{i,j} = \sum_{k=i}^{j} x_k^2 - (j-i+1)M_{i,j}^2 \,. \tag{8}$$

Welford [35] proposed updating formulations for $M_{i,j}$ and $S_{i,j}$:

$$M_{i,j} = M_{i,j-1} + \frac{1}{(j-i+1)}\left(x_j - M_{i,j-1}\right) = M_{i,j-1} + \Delta M_{i,j} \,, \tag{9}$$

$$S_{i,j} = S_{i,j-1} + (j-i)(j-i+1)\Delta M_{i,j}^2 \,. \tag{10}$$

These formulae are the building blocks for the updating batch-based estimators of $\sigma(\hat{\mu})$, as detailed below.

3.2. Updating batch-based methods

In the standard (offline) batch-based methods, see Ref. [27], first a set of batches are created using the time samples collected from a turbulence time series. For the non-overlapping batch mean (NOBM) method that we discuss here, the set of samples $\{x_i\}_{i=1}^{n}$ are divided into K batches of size N_b. Then, $\bar{x}_k$ for $k = 1, 2, \ldots, K$ is computed as the sample mean of the samples included in the k-th batch. Using $\{\bar{x}_k\}_{k=1}^{K}$, the sample mean of the original time series is estimated by

$$\hat{\mu} = \frac{1}{K}\sum_{k=1}^{K} \bar{x}_k \,, \tag{11}$$

the variance of which is estimated as

$$\hat{\sigma}^2(\hat{\mu}) = \frac{1}{K(K-1)}\sum_{k=1}^{K} (\bar{x}_k - \hat{\mu})^2 \,. \tag{12}$$

Performing these steps in an in-situ (updating) manner is straightforward. The main point is that two sets of sample means should be updated using Eq. (9): one for computing the batch-means, and the other one for computing the sample mean of the time series using the updated batch-means. But Eq. (10) is used only "per request" to update the estimated variance of the SME of the time series. These steps are summarized below and also schematically represented in Figure 1.

1. Specify the batch size N_b;
2. In the flow simulation's time loop, with a given sampling interval:
   (a) compute sample mean of every N_b samples using updating formula (9) $\rightarrow \bar{M}_i$;
   (b) update the SME of the time series and its variance via Eqs. (9) and (10), respectively $\rightarrow M_\mu, S_\mu$.

To improve the accuracy of $\hat{\sigma}(\hat{\mu})$ estimated by the NOBM method, Russo and Luchini [27] suggested considering the first-lag correlation between the batch means. The resulting batch-means and batch-correlation (BMBC) estimator for the SME uncertainty reads as,

$$\hat{\sigma}^2(\hat{\mu}) = \frac{1}{(K-1)(K-2)}\left(\sum_{k=1}^{K}(\bar{x}_k^2 - \hat{\mu}^2) + 2\sum_{k=1}^{K-1}(\bar{x}_k\bar{x}_{k+1} - \hat{\mu}^2)\right) \,, \tag{13}$$

which, compared to Eq. (12), has the second summation on its right-hand-side (RHS). In order to formulate an updating formula for this term, Eq. (16) below can be used with $m = 1$ and substituting x by $\bar{x}$. The implementation of the updating BMBC (iBMBC) method requires the last computed batch mean to be stored in memory.

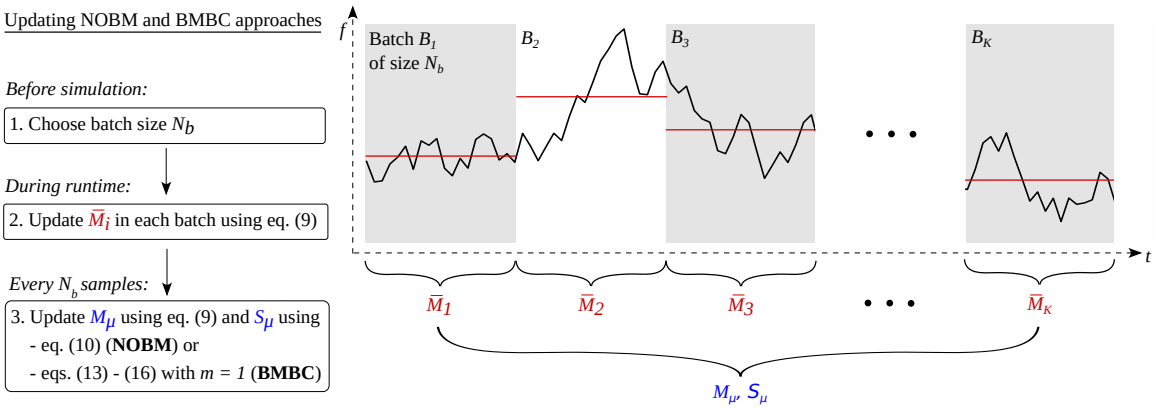


Figure 1: Schematic representation of the updating algorithm applied to the NOBM and BMBC methods. The batch means and the SME of x are updated using Eq. (9). The variance of the SME for the NOBM and BMBC methods is computed using Eqs. (10) and (13), respectively.

3.3. Updating MACF (iMACF) method

As a main drawback, the accuracy of the NOBM and BMBC methods is controlled by the batch size N_b, which, in general, cannot be chosen intuitively prior to the simulations. Moreover, the optimal value of N_b depends on the flow variable and also changes with the spatial location [36]. These restrict the suitability of adopting the batch-based uncertainty estimators for in-situ applications. Therefore, we focus on the approach of modeling the autocorrelation $\rho_m = \gamma_m/\gamma_0$ in Eq. (3) for any lag $m > 0$. In particular, the proposed MACF method outlined in Section 2 is considered, where a set of modeling parameters are estimated using the training sample-estimated ACF values $\{\rho_i\}_{i=1}^{n_{\text{train}}}$ corresponding to the time lags $\{m_i\}_{i=1}^{n_{\text{train}}}$. The only potential subjectivity in this method could be due to the choice of the training data set which, however, as discussed in Section 6.1, does only have a negligible influence on the accuracy of the predicted uncertainties. This is a clear indication of the robustness of the proposed ACF model (6).

The standard estimation of the sample autocovariance at lag $m \geq 0$ via Eq. (4) using n time samples requires having all $n - m$ samples at once. Instead, we aim at deriving a formulation for sample-estimated ACF which is updating and requires a minimum amount of storage. As the first step, we define the sample-estimated ACovF at lag m by including the samples from i to j, as:

$$\tilde{\Gamma}_{i,j}^m = \Gamma_{i,j}^m/(j - i + 1)\,, \tag{14}$$

where,

$$\Gamma_{i,j}^m = \sum_{k=i}^{j}(x_k - M_{i,j})(x_{k-m} - M_{i,j})\,, \tag{15}$$

where $M_{i,j}$ is the updating SME of x defined in Eq. (7). Expanding this definition results in the following expression:

$$\begin{aligned}\Gamma_{i,j}^m \;=\; & \Gamma_{i,j-1}^m - \Delta M_{i,j}\sum_{k=i}^{j-1}(x_k + x_{k-m}) + (j - i - m + 1)M_{i,j}^2 - (j - i - m)M_{i,j-1}^2 \\ & + x_j x_{j-m} - M_{i,j}(x_j + x_{j-m})\,,\end{aligned} \tag{16}$$

where i is, in practice, fixed (i.e. chosen after the statistically stationary condition of the turbulent flow is established), and for a given m it is needed that $i > m$. As detailed in Appendix B, for $m = 0$ this expression becomes identical to Eq. (10) for the variance of x. For computing all the terms on the RHS of Eq. (16) except the last two, no storage of the samples of x is needed. The storage demanded by the two terms containing x_{j-m} can basically increase with j and m, however, our optimal implementation requires a

Algorithm 1: Updating MACF (iMACF) method.

Input: Choose $m_{\max}$ and T_s, where $m_{\max}$ is sufficiently larger than T_s. Although these choices are, in general, arbitrary, choose the $m_{\max}$ such that $m_{\max} = M_{\max}T_s$ with $M_{\max}$ being an integer. This makes the initialization step more efficient.

1 Initialize the list of training lags as $\mathbf{m}_{\text{train}} = [T_s, 2T_s, \ldots, M_{\max}T_s]$. For each $m \in \mathbf{m}_{\text{train}}$, a buffer array is created to store the corresponding $\Gamma^m_{i,j}$.
2 **while** CFD simulation is not finished **do**
3 **if** time t is divisible by T_s **then**
4 **if** $t \leq m_{\max} = M_{\max}T_s$ **then**
5 fill the buffer array with the samples of x.
6 **else**
7 Compute $\Delta M_{i,j}$ in Eq. (9),
8 Update $S_{i,j}$ using Eq. (10),
9 **forall** $m \in \mathbf{m}_{\text{train}}$ **do**
10 update Γ^m_{ij} using Eq. (16),
11 **end forall**
12 Update the buffer array,
13 Update $M_{i,j}$ in Eq. (9).
14 (optional) Use the training autocorrelations at lags $\mathbf{m}_{\text{train}}$ to fit the ACF model (6) which is then plugged into Eq. (3) to estimate $\hat{\sigma}(\hat{\mu})$.
15 **end if**
16 **end if**
17 **end while**

buffer of a fixed size. In fact, this size can be kept small even for obtaining a high accuracy of the estimated uncertainty of an SME, thanks to the flexibility of the uncertainty estimation techniques described below. In principle, going from $j-1$ to j, first the sample at $j-m$ that is already available in the buffer is used to update the last two terms in Eq. (16). Then, the oldest stored sample is removed from the buffer followed by adding the most recent sample x_j to the buffer.

Our investigation in Section 6.1 shows that, for accurately modeling the ACF of a time series using Eq. (6), only a few sample-estimated autocorrelations are required. Moreover, there is a great flexibility regarding the selection of the training samples. Therefore, Eq. (16) should be evaluated at a set of $m \in \mathbf{m}_{\text{train}}$. Bearing in mind the issue with x_{j-m}, the maximum value of m, hereafter $m_{\max}$, can be a main factor in driving the overall cost of the in-situ estimation of the sample-estimated autocorrelations using Eq. (16).

In practice, to enhance the computational efficiency and minimize the required memory storage of the iMACF method for in-situ estimation of the uncertainty of an SME, see Algorithm 1, time samples can be taken with a frequency of $T_s\Delta t$, where, T_s is a positive integer and Δt is the time-step size in the flow simulation. The selection $\mathbf{m}_{\text{train}}$ can be linked to T_s to create an efficient computational algorithm.

Line 14 in Algorithm 1 is executed upon a user request or automatically following a preset schedule, as this step is used for monitoring the convergence of the turbulence statistics. The procedure for computing the sample training autocovariances is shown schematically in Figure 2. Recall that the buffer is required for the purpose of in-situ evaluation of the terms containing x_{j-m} in Eq. (16). The low-storage feature of the algorithm can be inferred from Line 12: the sample at the smallest lag in the array is removed, other samples are shifted, and the new sample is appended to the end of the buffer array. However, to increase the efficiency in practice, the oldest value in the buffer is overwritten by the newest value while keeping all other values untouched, resulting in a cyclic writing to the buffer array. In particular, for the Python library Numpy [10] used in the present work, this has been implemented by using numpy.roll which creates copies of the buffer array for each execution.

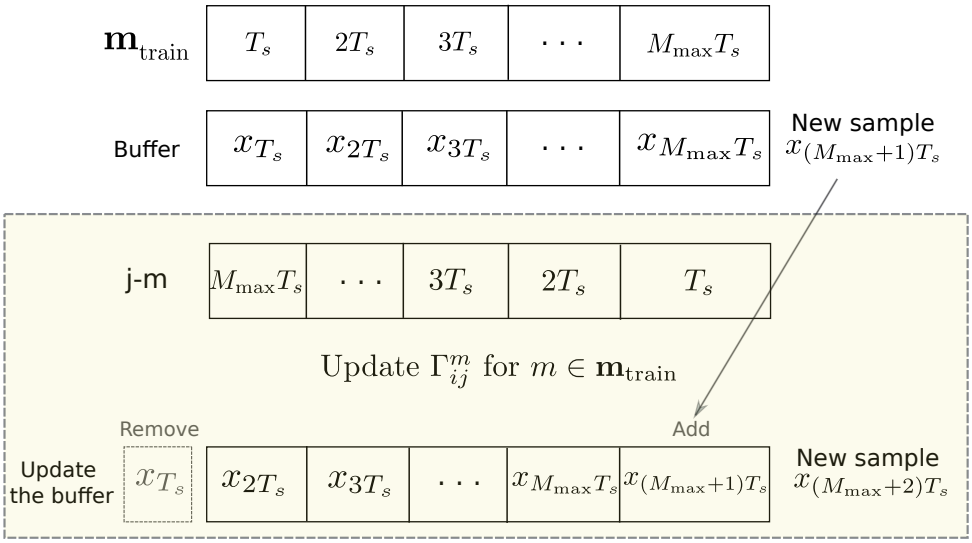


Figure 2: Schematic representation of the updating algorithm to compute sample autocovariances from Eq. (14) using the updating expression (16), see Algorithm 1. The operations within the shaded area are repeated in a loop over $m \in \mathbf{m}_{\text{train}}$.

4. Workflow Design and Software Implementation

The updating UQ methods from Section 3 are implemented in Python as an extension of UQit [25] to allow for a flexible adaptation of the algorithms during the development and validation of the framework. In this section, the proposed workflow applied to the CFD simulations and its components are described.

The CFD simulation mesh is processed block-wise on each MPI (Message Passing Interface) rank to take advantage of the matrix manipulations in linear algebra libraries, more specifically Numpy [10], and provide a straight-forward high-level parallelization of our code. For the purpose of benchmarking the algorithms in a realistic simulation environment, we utilize the ParaView Catalyst [2] interface. Therefore, algorithms are wrapped as VTK (Visualization Toolkit) filters and executed in a Catalyst pipeline together with native VTK filters, for instance, to extract slices from the original mesh. Although this setup is generally independent of the flow solver, the tests in the following sections were performed using Nek5000 [9] with an additional Catalyst interface [2]. The performance analysis of the entire in-situ workflow is discussed in Section 6.5.

Referring to Section 3.3, we note that the algorithm can be split into three major parts: the updating formula to compute the ACF at different lags, the estimation of the continuous ACF via curve-fitting (i.e. Eq. (6)), and finally the estimation of uncertainty in the SME of turbulence quantities. While all three steps could, in principle, be executed during the runtime of a simulation, we may prefer to run the second and third steps offline. This allows us to take advantage of the reduced data I/O due to the online computation of sample-estimated ACF with only a small amount of additional runtime and memory allocation. On the other hand, the UQ results are only needed at the end of the simulation or at a few intermediate time steps preset or requested by a user. Thus, the computationally intense curve-fitting step can be detached from the workflow to avoid slowing down the simulation for which the in-situ UQ is enabled. Following this, we implemented the second and third steps of the algorithm in a ParaView plugin to further process the data that is generated during the simulation. Note that if the computational architecture allows it, idle computational resources can be used for the detached steps, see [12].

The verification of the framework has been carried out by comparing the proposed updating algorithm to the corresponding offline algorithms from the statsmodels Python library [30], which has access to the full time series, using the data from a DNS of the 3D flow around a cylinder at Re = 3900 as introduced in Section 5. For this purpose, we set up a Catalyst workflow to extract a 2D slice from the original mesh

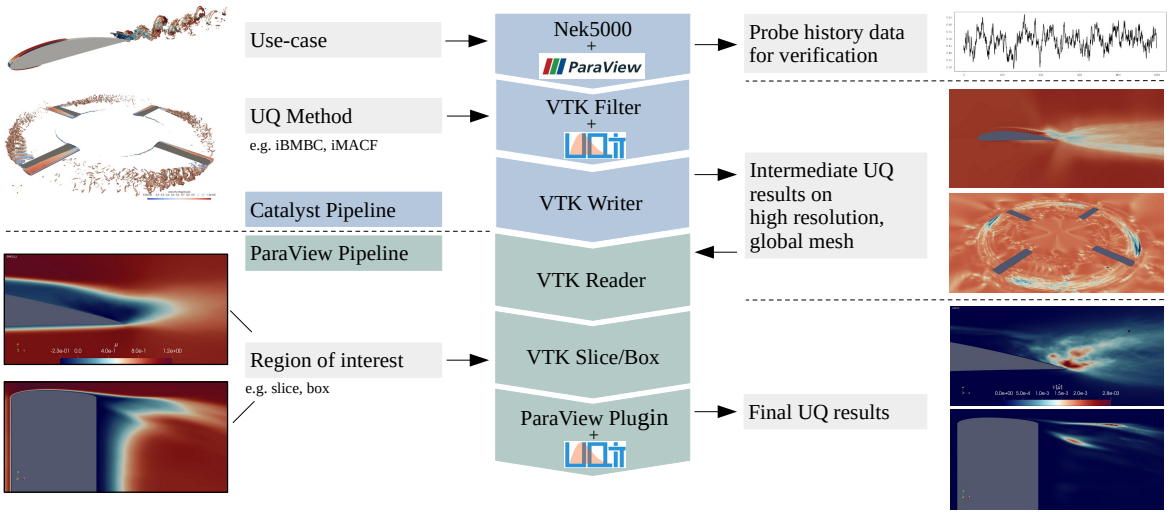


Figure 3: The workflow of the in-situ UQ framework proposed in Section 3.3 to be applied to CFD simulations.

that gets both processed in-situ by the described algorithm and written to disk in a VTK format, at the same time steps. That way, we can process the exact same data in both online and offline modes, see Figure 3. For test data from the cylinder use case with a set of 100000 time-steps, both results are identical up to around 14 significant figures, providing a strong indication that the implementation of the in-situ UQ algorithm is practically error-free. As there are many additions of values that differ by several orders of magnitude during the evaluation of the updating algorithm, we also investigated the influence of reducing the floating point precision to 32-bit. In this case, after 100000 time-steps the accuracy drops down to around 4 significant figures. Thus, single precision floats should provide sufficient accuracy of the results for most cases, as the numerical error is still much smaller than the sampling uncertainty in the sample mean estimations.

5. Flow Solver and Use Cases

5.1. Flow solver

The described workflow can be applied to any flow solver, however, we consider Nek5000 [9] in the present study. Nek5000 is an open-source spectral-element [24] flow solver developed at Argonne National Laboratory (US). The low numerical dissipation and high parallel performance [20] makes this software perfect for high-fidelity simulation of large-scale advection-diffusion problems, and in particular turbulent flows, see e.g. Refs. [7, 18, 33]. In the spectral-element method, the computational domain is decomposed into a set of non-overlapping spectral subdomains called elements. The Navier–Stokes equations in their weak form are then discretized over such elements treated as spectral domains. We focus here on the staggered grid formulation, so-called $P_N - P_{N-2}$ formulation, in which velocity and pressure are represented on different meshes. In this case, the functional spaces of the primitive variables velocity and pressure are spanned by the Lagrangian interpolants on the Gauss–Lobatto–Legendre (GLL) and Gauss–Legendre (GL) quadrature points, respectively. A high spatial resolution is obtained by adopting high-order polynomials with order N. In the simulations discussed in the present work, we have used N equal 5 for the rotating parts simulation, 7 for the channel flow and 11 for the NACA4412 flow. For integration in time, a semi-implicit

scheme is adopted in which the nonlinear terms in the Navier–Stokes equations are treated explicitly and the remaining unsteady Stokes problem is solved implicitly. More in-depth discussion of the algorithm can be found in e.g. Ref. [8].

The domain decomposition into elements is a main source of both the algorithm parallelism and meshing flexibility. The latter can be significantly improved by including an adaptive mesh refinement (AMR) strategy, which is a self-adapting algorithm allowing to dynamically modify the mesh according to the estimated computational error. An AMR version of Nek5000 [21, 22] was developed at KTH using h-type refinement, in which the total number of grid points is modified by splitting/merging the elements while keeping the number of collocation points per elements fixed. This implementation follows a conforming-space/nonconforming-mesh approach [8], in which a special interpolation operator is applied at nonconforming interfaces avoiding the construction of the so-called mortar elements [16]. Although this approach limits the permitted mesh configurations, it allows to use simple and efficient mesh management tools based on the octree refinement (e.g. p4est library [3]), and has a relatively small impact on the solver parallel performance. Regarding the AMR simulations in Section 5.2.2, the mesh refinement was driven by a spectral error indicator formulated by Mavriplis [17].

5.2. Use cases

In this section, the flow cases to which the in-situ UQ framework is applied are briefly introduced. Due to their complexity and engineering relevance, wall-bounded turbulent flows are considered. In order to show the generality of the UQ framework, the simulations include both conformal and non-conformal (AMR) computational grids.

5.2.1. Simulations with conformal mesh

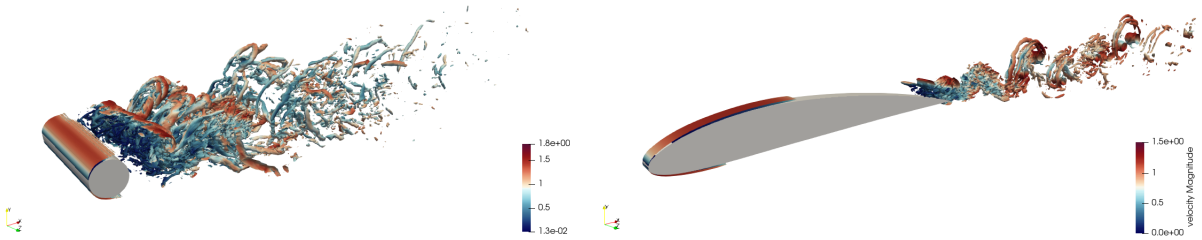
As one of the most canonical wall-bounded turbulent flows, the turbulent channel flow is considered which comprises of two parallel flat walls separated by distance 2δ. This internal flow is periodic in the streamwise and spanwise directions, which correspond to x and z, respectively. The wall-normal direction is represented by y. The absence of uncertainty due to initial and boundary conditions makes the turbulent channel flow suitable for fundamental studies. For a-priori assessment of the proposed UQ algorithm, the data of turbulent channel flow at the friction Reynolds number $\mathrm{Re}_\tau = 300$ are used in Sections 6.1 and 6.2. This Reynolds number is defined as $\mathrm{Re}_\tau = u_\tau \delta/\nu$, where u_τ is the averaged wall friction velocity and ν denotes the kinematic viscosity. The data includes the wall-normal profile of the streamwise velocity component and wall friction velocity averaged over time and the periodic directions. The simulation was performed using polynomial order $N = 7$ in Nek5000 and a total number of 10^5 time samples at time intervals $0.008\delta/U_b$ (U_b is the bulk velocity) were collected to analyze the UQ algorithm in an offline mode.

The turbulent flow around a cylinder at $\mathrm{Re} = 3900$ as well as a NACA4412 wing-section at $\mathrm{Re} = 75000$ is considered as an external aerodynamic use case to which the UQ framework is integrated in an in-situ way. Figure 4 represents the iso-surfaces of the Q-criterion of a snapshot of the turbulent flow fields. In the spanwise direction, a periodic boundary condition is applied. The flows contain several interesting flow features, such as laminar separation and periodic structures in the wake region. Estimation of the uncertainty in the flow statistics in the wake region is especially challenging due to the strong autocorrelation of the time samples over large time averaging intervals. In Section 6.3, the in-situ UQ framework is applied to estimate the uncertainty in the sample mean velocity at all GLL points of the three-dimensional mesh. Furthermore, an analysis of the computational performance of the in-situ UQ framework integrated into the CFD simulations is detailed in Section 6.5.

5.2.2. AMR simulation of a rotor

The most complex flow case presented in this work is a "toy" rotor studied within the EU project EXCELLERAT[1] in cooperation with CINECA[2]. The rotor was built out of four blades with NACA0012

[1] https://www.excellerat.eu/
[2] https://www.cineca.it/en



(a) (b)

Figure 4: Flow around a (a) circular cylinder at Re = 3900 and (b) a NACA4412 wing section at Re = 75000 computed with Nek5000. Visualization of the instantaneous flow fields using isosurfaces of the Q-criterion colored by the velocity magnitude.

airfoil of length 3 (using an airfoil chord as a unit length), rounded wing tips and an angle of attack equal to 5°. The simulation was performed in a rotating reference frame using an AMR framework [21, 22]. The simulation time unit and an angular rotation speed of the reference frame, Ω, were adjusted to set a linear velocity of an external blade tip (located at radius $r_0 = 6.5$) to 1. This gives the rotation period $T = 2\pi/\Omega = 2\pi r_0$ equal 40.84. The Reynolds number based on the chord length and the rotation speed at the position of the external blade tip was equal to Re = 10000. The spectral error indicator was based on the Cartesian components of the velocity field averaged over 0.2 simulation time units. At each refinement stage 10% of the elements with the highest estimated computational error were refined, and the elements with an indicated error below 5.0×10^{-7} were marked for coarsening. This process was repeated multiple times until the required resolution was reached by increasing the number of elements from 346336 (initial conforming mesh) up to 1088595. A maximum allowed mesh resolution was defined by setting the maximum refinement level to 3. The rotor was embedded in a cylindrical domain of radius 33.2 and extended in the vertical direction (y axis in the simulation coordinate system) from -18.2 to 13.2. The simulation was run for 1.9 full rotations before the UQ framework was executed. The visualization of the instantaneous flow field at a simulation time $t = 77.6$ together with the cut through the domain mid-plane showing the element boundaries is presented in Figure 5.

6. Results and Discussion

In this section, we elaborate on the application and performance assessment of the proposed in-situ UQ framework. Section 6.1 is focused on the assessment of the accuracy and robustness of the proposed ACF model (6). The use of this model to estimate uncertainty in the sample mean estimator (1) is discussed in Section 6.2. The assessment of the in-situ estimation of such uncertainties for turbulent flow simulations introduced in Section 5.2 is detailed in Section 6.3. This is followed by Section 6.5 where the computational performance of the algorithm is discussed.

6.1. Accuracy and robustness of the proposed ACF model

The core requirement for precisely estimating the uncertainty of an SME using the iMACF method of Section 3.3 is to accurately model the ACF that is plugged into Eq. (3). For this purpose, we proposed the algebraic function (6), the optimality of which is motivated in this section. For quantitative assessment of the accuracy and robustness of the proposed ACF model, two sets of time series data are employed: velocity samples of a turbulent channel flow introduced in Section 5.2.1, and synthetic samples generated from a first-order autoregressive model, AR(1). In particular, AR(1) is defined as

$$x_i = a x_{i-1} + b\varepsilon_i\,, \quad i = 1, 2, \ldots, n\,, \tag{17}$$

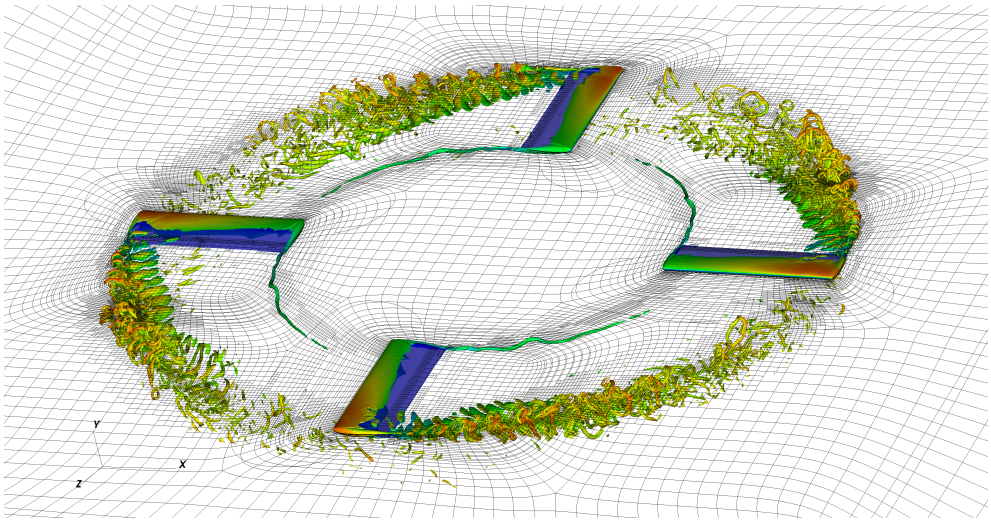


Figure 5: Flow around a toy rotor at Re = 10000 in a rotating reference frame computed with the AMR version of Nek5000 [21, 22]. The plot presents the iso-surfaces of the λ_2-criterion of the instantaneous flow field and the cut through the domain mid-plane showing the element boundaries. Variable resolution and nonconforming interfaces are clearly visible in the wake region behind the blades.

where ε_i are uniformly-distributed as $\varepsilon_i \sim \mathcal{U}[0,1]$ and $a, b \in \mathbb{R}$. To ensure the time series is statistically stationary, it is required that $|a| \leq 1$. The particular values of $a = 0.1$ and $b = 0.9$ are considered for the analyses in the present section and Section 6.2. For any n samples, the analytical value of the variance of the expectation of x can be obtained from

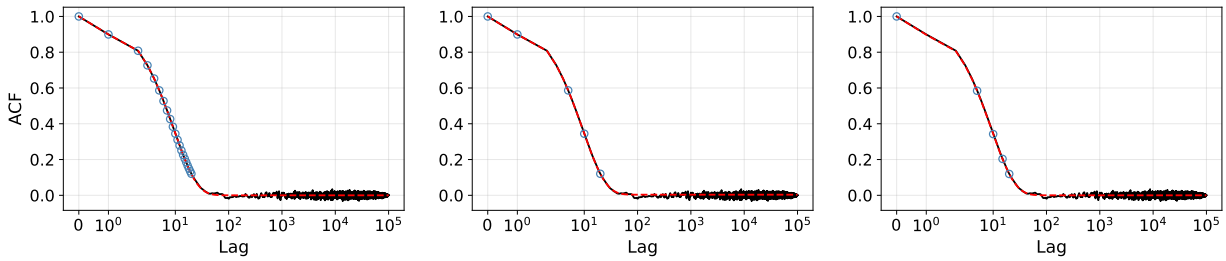
$$n\mathbb{V}[\mathbb{E}[x]] = b^2/(12(1-a)^2)\,. \tag{18}$$

This expression can be used to validate the values of $\hat{\sigma}^2(\hat{\mu})$ estimated by employing the UQ approaches introduced in Section 3. Both of the data sets used in the section (channel flow and AR(1)) have a sample size of $n = 10^5$.

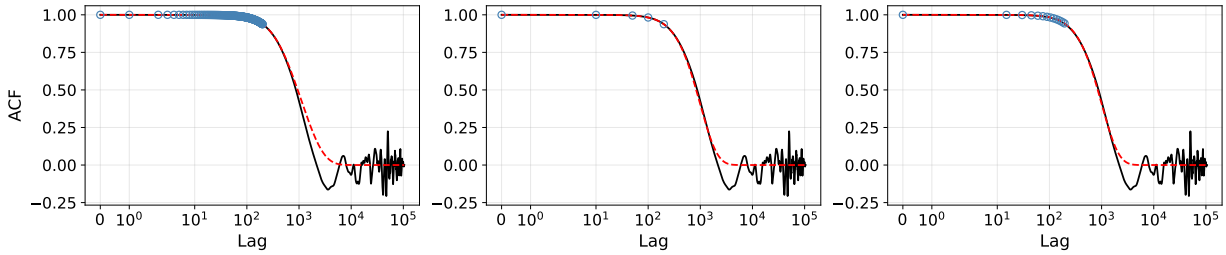
Figure 6 shows the sample-estimated (black lines) and modeled ACFs (dashed red lines) of the time series samples of the AR(1), and also the channel flow streamwise velocity samples at a distance from the wall. The modeled ACF is obtained by the proposed MACF model (6) using different sets of the training sample-estimated ACFs (shown by markers) which are subsets of the available samples of each of the two time series data sets. The training sample-estimated ACFs in Figure 6 are particularly chosen in two different ways over the range of lag zero and a given maximum lag: i) at a set of sparse lags with non-equal spacings (middle column), and ii) a set of lags with a fixed spacing that is a multiplication of the time lag between the original samples (right column). For reference, the modeled ACF from a power-law model, see Appendix A, is also shown (left), which is computed using the training sample ACFs at all the original sampled times. For all the cases, the modeled ACF is found to be accurate for both AR(1) and channel flow data up to the point where the sample-estimated ACFs show spurious wiggles. This particularly confirms the robustness and accuracy of the ACF model (6), since even with a small number of training samples an accurate smooth prediction of the ACF is achieved.

As explained in Section 3.3, for the in-situ MACF estimator (iMACF), the size of the training data must be as small as possible and at fixed sampling intervals to ensure the low-storage (memory-efficient) property. Furthermore, the training samples should be taken at the lowest possible lags. These requirements are met in the construction represented in the right column of Figure 6, which is also used in practice in the in-situ UQ framework as detailed in Sections 6.3 and 6.5.

A general rule found in the present study is to choose the sampling frequency T_s for selection of the training data such that the first non-zero time lag does not exceed ≈ 15 to obtain an accurate ACF model.



(a) Data: AR(1) defined in Eq.(17) (synthetic data).



(b) Data: $\langle u \rangle_{xz}$ at $y^+ = 94$ of a turbulent channel at $\mathrm{Re}_\tau = 300$. Note that $\langle \cdot \rangle_{xz}$ represents averaging over the wall-parallel directions x and z. The inner-scaled wall distance y^+ is defined as $y^+ = y u_\tau / \nu$.

Figure 6: Sample estimated ACF (solid black line), sample training ACF data (markers), and modeled ACF (dashed red line) for 10^5 samples. The modeled ACFs are obtained using: (left) The power-law model (see Appendix A), (middle) the ACF model (6) with a sparse set of training lags $[0, 1, 5, 10, 20]$ for the synthetic data and $[0, 10, 50, 100, 200]$ for the channel flow data, and (right) the ACF model (6) with sampling intervals 5 for the synthetic and 15 for the channel flow data, respectively. The maximum lag at which the sample-estimated ACF data are used for training the ACF model is 20 for the synthetic and 200 for the channel flow data.

Less important is the maximum training lag (i.e. $M_{\max} T_s$ in Section 3.3), which could be chosen to minimize the number of elements in the buffer by considering the training ACF up to ≈ 0.8. This value was found through experimentation to make a balance between the size of the buffer array and the accuracy of the modeled ACFs. In any case, this upper limit should not be so large to include the wiggles around zero ACF. These guidelines are important to be imposed in practice noting that driven by the physics, the characteristics of the ACF of turbulence time series depends on the quantities as well as the locations in the flow domain (more specifically at different wall-distances in wall-bounded flows).

The final point regarding Figure 6 is that the most optimal scenario in terms of the number of training data required, is to use a sparse set of sample ACFs with non-uniform lag distances, i.e. the middle column in the figure. However, constructing an automatic procedure for updating the buffer in the in-situ algorithm, as it is made with a fixed sampling interval T_s in Figure 2, may not be possible. Another interesting observation is that the ACF modeled by the power-law method (see Appendix A), which is more involved than the proposed MACF, may deviate more from the sample ACF and thus result in less accurate estimates for uncertainties, see the plots in the bottom of Figure 6.

As briefly mentioned in Section 2, in some previous studies in the literature, function (5) has been used to model the ACF. However, Figure 7 shows that the use of this model is limited to processes such as AR(1), where the integral time scale, i.e. the area below the ACF-time curve, is small. Conversely, for the turbulence time series where the history effects last for a longer time and the corresponding autoregressive model is of high order, the ACF model (5) fails to accurately obtain a smooth function for the ACF. In particular, in Figure 7, the ACF inaccurately modeled by Eq. (5) is extended over longer time lags compared to the sample-estimated ACF, resulting in an over-estimation of the uncertainty in the associated SME when it is employed in Eq. (3).

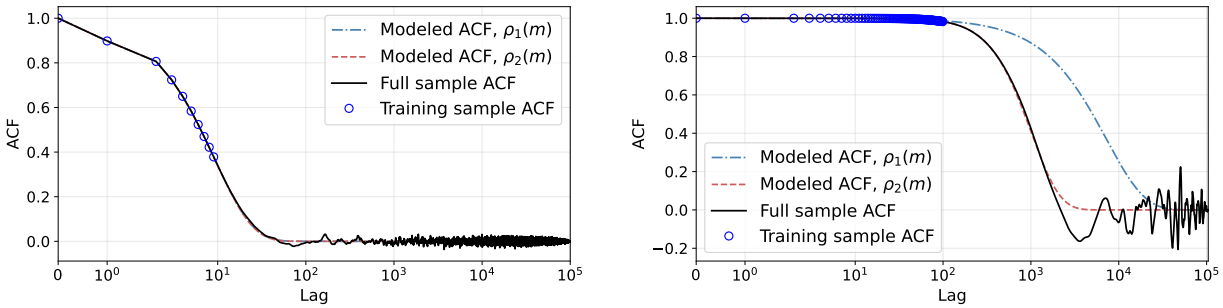


Figure 7: Impact of the function used for modeling the ACF from 10^5 time samples taken from (left) ARM(1) defined in Eq. (17), and (right) the turbulent channel flow averaged streamwise velocity $\langle u \rangle_{xz}$ at $y^+ = 94$. The training sample-estimated ACF data are taken at the first 10 lags for the AR(1) and 100 lags for the channel flow data, respectively. The ACF models $\rho_1(m)$ and $\rho_2(m)$ refer to Eqs. (6) and (5), respectively.

6.2. Validation and robustness of the uncertainties estimated by the MACF method

As motivated above, function (6) can lead to accurate smooth models for the ACF, which can be plugged into Eq. (3) to accurately estimate $\hat{\sigma}(\hat{\mu})$, i.e. the standard-deviation of a sample mean estimation $\hat{\mu}$. This will be shown in this section with two different studies applied to the data of the AR(1) defined in Eq. (17), and the turbulent channel flow velocity data.

For the first study, several realizations of a time series are generated, and the distribution of their sample-estimated SME's uncertainty is compared to the associated empirical uncertainty obtained from Eq. (20). The latter can reflect the true or population uncertainty. For this purpose, an ensemble of size N_e of the computationally inexpensive AR(1), Eq. (17), is considered. Here, each of the AR(1) simulations starts from an independent random sample taken from the uniform distribution $\mathcal{U}[0, 1]$ and continues to $1.5n$ samples. The first $0.5n$ samples are discarded to account for the burn-in. Hence, for each of the N_e realizations there are n samples of the AR(1) to which the MACF estimator is applied for estimating the corresponding $\hat{\sigma}(\hat{\mu})$. For the MACF estimator to be consistent, the probability density function (PDF) of the resulting $\hat{\sigma}(\hat{\mu})$ should contain the empirically-estimated uncertainty of the ensemble expectation μ_e,

$$\mu_e = \mathbb{E}[\hat{\mu}] \approx \frac{1}{N_e} \sum_{i=1}^{N_e} \hat{\mu}_i \,, \tag{19}$$

measured by σ_e that is given by,

$$\sigma_e^2 = \mathbb{V}[\hat{\mu}] \approx \frac{1}{N_e} \sum_{i=1}^{N_e} (\hat{\mu}_i - \mu_e)^2 \,. \tag{20}$$

Figure 8 illustrates the outcome of the described procedure for different values of the sample size n where the ACF modeled by Eq. (6) is constructed using two different sets of training sample-estimated ACFs. In all cases, the empirical σ_e is very close to the exact $\sigma(\hat{\mu})$ (given by Eq. (18)), and falls within the PDF of the $\hat{\sigma}(\hat{\mu})$ estimated by the MACF method. Note that as expected, by increasing the sample size n the PDF of $\hat{\sigma}(\hat{\mu})$ becomes narrower, but more importantly the mode of the PDF (most probable value of $\hat{\sigma}(\hat{\mu})$) becomes almost the same as the exact $\sigma(\hat{\mu})$.

For a single realization of the turbulent channel flow, a study to validate the $\hat{\sigma}(\hat{\mu})$ estimated by the MACF method is to make a comparison with the power-law method which has been used in the literature as in Refs. [23, 36], see Appendix A. According to Figure 9, at all wall-normal locations, the agreement between the MACF and power-law methods is good, in general. There is a small discrepancy in the outer layer of the mean velocity profile, i.e. $y^+ \gtrsim 150$, which originates from the slight imperfection of the modeled ACF potentially by both methods (although in Figure 6, the power-law method is found to be slightly less accurate than MACF for modeling the ACF). To ensure the robustness of the MACF method, the estimated

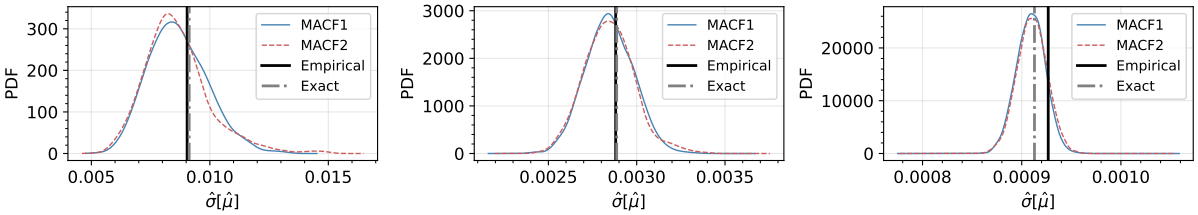


Figure 8: Validation of the $\hat{\sigma}(\hat{\mu})$ estimated by the MACF method using the samples of the ARM(1) defined in Eq. (17). The plots show the PDF of the $\hat{\sigma}(\hat{\mu})$ using the sample-estimated ACF at the first time lags (MACF1) and lags $[0, 2, 5, 8, 15]$ (MACF2) for training the ACF model (6). The PDFs are obtained by 2000 repetitions of the independent realizations of the AR(1) with the sample size n per realization being equal to (left) 10^3, (middle) 10^4, and (right) 10^5. The solid and dash-dotted vertical lines respectively show the empirical estimate σ_e given by Eq. (20), and the exact value of $\sigma(\hat{\mu})$ obtained from Eq. (18).

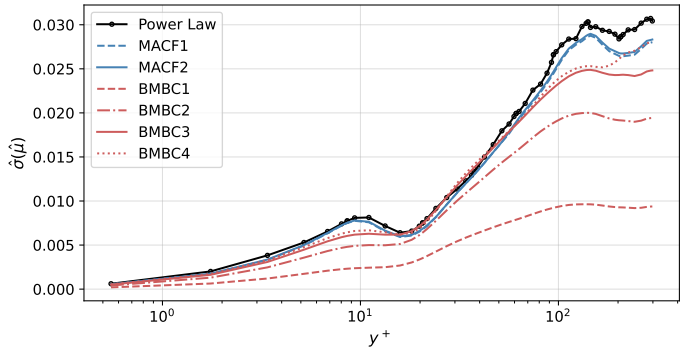
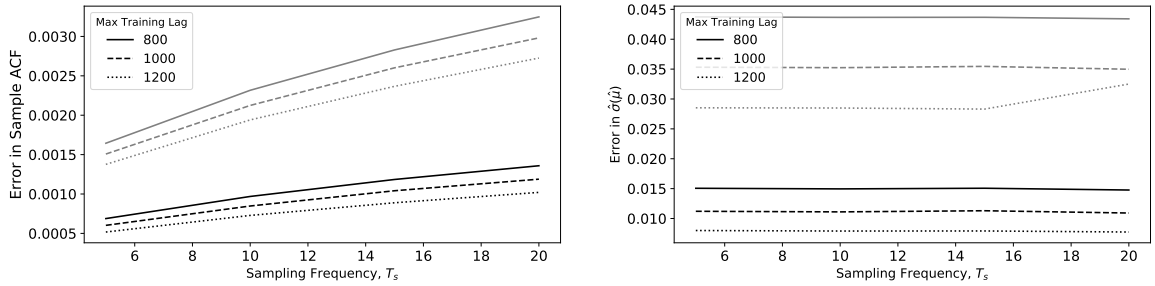


Figure 9: Estimated standard deviation of the SME of $\langle u \rangle_{xz}$, the averaged streamwise velocity of the turbulent channel flow, versus the inner-scaled wall distance y^+ (10^5 time samples are used). The following hyperparameters are used for different uncertainty estimators; power-law method (see Appendix A): the order of the ARM is $p = 100$ and the sample-estimated ACFs at first 300 time lags are used for training the ACF model; MACF1 and MACF2: the MACF method with the ACF model (6) trained by the sample-estimated ACF at first 300 lags (MACF1) and lags corresponding to [0,10,50,100,200,250,300] (MACF2); BMBC1 to BMBC4: the BMBC method with the batch size equals to 100, 500, 1000, 2000, respectively.

uncertainties associated to two different sets of ACF training samples are found to be almost the same. This is in fact a main advantage of the proposed ACF model (6) for estimating $\hat{\sigma}(\hat{\mu})$, i.e. the resulting estimates are not biased with respect to the choice of the hyperparameters. To make this point clearer, plots from the BMBC approach [27] are also provided in Figure 9. Clearly, the estimated $\hat{\sigma}(\hat{\mu})$ are sensitive to the batch-size and for none of the considered batch sizes the estimates would be close enough to those by the MACF and power-law methods. As formerly stated, this potential bias in the $\hat{\sigma}(\hat{\mu})$ values is the main barrier for using the batch-based methods for in-situ applications. But if we need to choose one of such approaches, the BMBC approach of [27] is preferred to the NOBM, see the discussion in Appendix C.

6.3. Robustness of the in-situ UQ algorithm

In this section, we consider the iMACF algorithm 1, the in-situ version of the MACF method described in Section 3.3. The aim is to assess the impact of the relevant hyperparameters on the accuracy of the modeled ACF and the resulting $\hat{\sigma}(\hat{\mu})$ when the sample ACFs are computed in an in-situ way following Eq. (16). To this end, the time series of the streamwise velocity of the turbulent channel flow at any distance from the wall can be considered. The three hyperparameters of the iMACF method investigated here are $m_{\max}$, the maximum lag up to which the sample-estimated ACF from Eqs. (14) and (16) are used to train the ACF model (6), T_s, the sampling interval, and, n the sample size.



(a) (b)

Figure 10: Impact of the maximum training lag $m_{\max}$, sampling frequency T_s and sample size n on the error in (a) the sample-estimated ACF, e_ρ, and (b) the standard deviation of the SME of $\langle u \rangle$, $e_{\hat{\sigma}}$, obtained from the iMACF method. The data belong to the channel flow streamwise velocity $\langle u \rangle_{xz}$ at the wall distance $y^+ = 262$ with size n equal (black lines) 100000 and (gray lines) 50000. The error in the plots (a) and (b) is defined by Eqs. (21) and (22), respectively.

The two metrics evaluated and plotted in Figure 10 are the error in the sample-estimated ACF, e_ρ, (left plot) and the error in estimated $\hat{\sigma}(\hat{\mu})$, $e_{\hat{\sigma}}$, (right plot) that are respectively defined as,

$$e_\rho = \|\rho_{st}(m) - \rho_{up}(m)\|_2 / M_{\max}\,, \tag{21}$$

$$e_{\hat{\sigma}} = |\hat{\sigma}_{st}(\hat{\mu}) - \hat{\sigma}_{up}(\hat{\mu})| / \hat{\sigma}_{st}(\hat{\mu})\,. \tag{22}$$

The subscripts *st* and *up* denote the standard (offline using all data) and updating (in-situ) values, respectively. When evaluating e_ρ, the $M_{\max}$ is set as $M_{\max} = \text{int}(m_{\max}/T_s)$. According to Figure 10, for fixed values of n and $m_{\max}$, less frequent sampling (i.e. higher T_s) leads to higher values of e_ρ, however, the corresponding impact on the estimated $e_{\hat{\sigma}}$ is negligible. What, instead, has the highest influence on $e_{\hat{\sigma}}$ is the variation of n and $M_{\max}$. As the number of samples decreases, meaning that the averaging is run for a shorter time interval, both e_ρ and $e_{\hat{\sigma}}$ increase. Considering more lags for modeling an ACF, i.e. increasing $M_{\max}$, can lead to more accurate ACF computed in an updating fashion and consequently more accurate $\hat{\sigma}$. In any case, as shown for the particular data set here, the largest error in $\hat{\sigma}$ estimated by the updating algorithm for the considered range of hyperparameters is less than 5% compared to what could be estimated by the standard method. This confirms the robustness and accuracy of the iMACF algorithm 1.

6.4. Application to complex flows

In this section, we study the application of the in-situ UQ framework to the scale-resolving simulations of the turbulent flow around a NACA4412 wing section and a moving rotor with an adaptively refined mesh as introduced in Sections 5.2.1 and 5.2.2, respectively. In the former case, we compare the iMACF approach with an incremental implementation of the BMBC method for three different batch sizes, while the latter demonstrates the integration of the in-situ UQ framework with a transient AMR simulation. In both cases, the iMACF approach is applied to all points of the three-dimensional computational grid. As explained in Section 3.3, in the iMACF algorithm 1 we only run lines 7 to 13 during the simulation while line 14 is executed offline. There are multiple reasons for this choice. First, the fitting of the ACF model to the training data in line 14 usually contributes most to the overall computing time of the UQ workflow and can thus extend the total simulation time. However, there is no need to do this step online unless we want to monitor the uncertainty of the turbulence statistics during the runtime. Second, the user might only be interested in the UQ results for a specific region of the simulation domain, such as a 2D slice or a 3D subdomain. Selection of these can be done after the termination of the simulation to cut down the computing time significantly. Third, there are a few hyperparameters that can influence the convergence of fitting the MACF model, including the selection of training lags. Thus, by running Line 14 in Algorithm 1 offline we can avoid faulty results that would require a rerun of the entire CFD simulation. Therefore, at

the end of the simulation, the ACF training values are saved on disk, where the values have been updated during the simulation using the streaming algorithm. Depending on the maximum number of the training lags, $M_{\max}$, the size of data to be stored is equivalent to only a couple of flow snapshots, and hence a small fraction of the data processed during the in-situ iMACF algorithm.

The NACA4412 use case is run for overall 5.37 time units equivalent to 200000 time steps. For the UQ estimate of the streamwise component of the velocity, u, we apply both the iMACF and iBMBC methods considering every 20-th time sample of the simulation (i.e. $T_s = 20$). For the iMACF algorithm 1, we choose $M_{\max} = 50$ to ensure sample ACF values are computed over a long enough range of lags at all spatial locations within the region of interest. However, to improve the quality of the modeled ACF especially at larger lags m, only a selection of the training ACFs was used by i) neglecting lags with the ACF values below ≈ 0.4 and ii) using ACFs at every 5-th lag. Both decisions are drawn from what was observed from the a-priori tests in Figure 6 (bottom row, left and middle). Altogether, the UQ results only show weak dependence on these parameters. It is noteworthy that we did not use a larger T_s during the in-situ process to avoid increasing the uncertainties in the updating SMEs and also due to the fact that the first non-zero lag at which the sample ACF is computed should not be too large to avoid errors in the modeled ACF, see the discussion in Section 6.1.

The UQ results of applying the iBMBC and iMACF to the NACA4412 wing are shown in Figures 11 and 12, respectively. The plots are made at a 2D slice through the midplane in the spanwise direction of the wing at $z = 0.025$. As expected, the estimated uncertainties by the BMBC method strongly depend on the batch size. For small batch sizes, the BMBC method underestimates the uncertainty and convergences towards the sample variance where the correct impact of the autocorrelations is ignored. For large batches, the number of batch means falls below the limit necessary for unbiased estimation of the uncertainties. These findings are in accordance with Eq. (3): The uncertainty in an SME depends on both the variance of the time series and the variation of the ACF with time lag. Thus, an underestimated uncertainty means failing in accurate estimation of these two contributors, where the role of the modeled ACF is more probable. We should bear in mind that the batch size imposes a cut-off to a modeled ACF, see [36]. Therefore, there is only a small range of batch sizes that could lead to reliable uncertainty estimates, where the range depends on the spatial location and QoI. In fact, for the NACA4412 simulation, we observed a difference in the inflection point of the ACF of up to two orders of magnitude depending on the specific location in the flow domain. In conclusion, applying the BMBC method in an in-situ way cannot be automated since the proper batch sizes cannot be chosen adaptively, instead they have to be chosen a priori. For the batch size of 100 samples, the BMBC results are found to be most similar to those of the MACF approach. However, we still see slightly lower values of the BMBC uncertainties and small differences in their spatial distribution that can be explained by the observations described above.

The NACA0012 rotor use case with a 5°AoA is simulated for a total 3.6 time units equivalent to 32400 time steps with variable size. For estimating the uncertainties, the data are interpolated during runtime to equi-distant time steps with a fixed $\Delta t = 2{\cdot}10^{-3}$ that approximately corresponds to every 18-th sample of the simulation (i.e. $T_s = 18$). As a result of this, a total number of 1800 samples are considered for the iMACF approach and $M_{\max} = 20$ is chosen for the UQ estimates. The sample-estimated ACF values below 0.4 are neglected when constructing the ACF model (6). The SME of the streamwise velocity component at two planar slices, $y = 0$ and $z = 6$, together with their uncertainty estimated by the iMACF method are shown in Figure 13. Similar to the NACA4412 wing case, the largest uncertainty can be found in the regions with large turbulence fluctuations and/or long autocorrelations (long-lasting history effects). The examples of the latter can be found in the regions of slowly convecting fluctuations or laminar separated flows. It should be emphasized that the uncertainty does not necessarily increase with the SME of the velocity, and particularly regions can be identified with SME values close to zero but with high uncertainties, e.g. downstream of the trailing edge of the rotor.

6.5. Computational performance of the in-situ UQ framework

In this section, we discuss the computational performance of the in-situ UQ framework for different mesh sizes and number of MPI ranks considering the NACA4412 wing use case. The weak and strong scaling tests of the framework are carried out using Nek5000 version 19.0, ParaView Catalyst 5.9 and Python 3.8 on

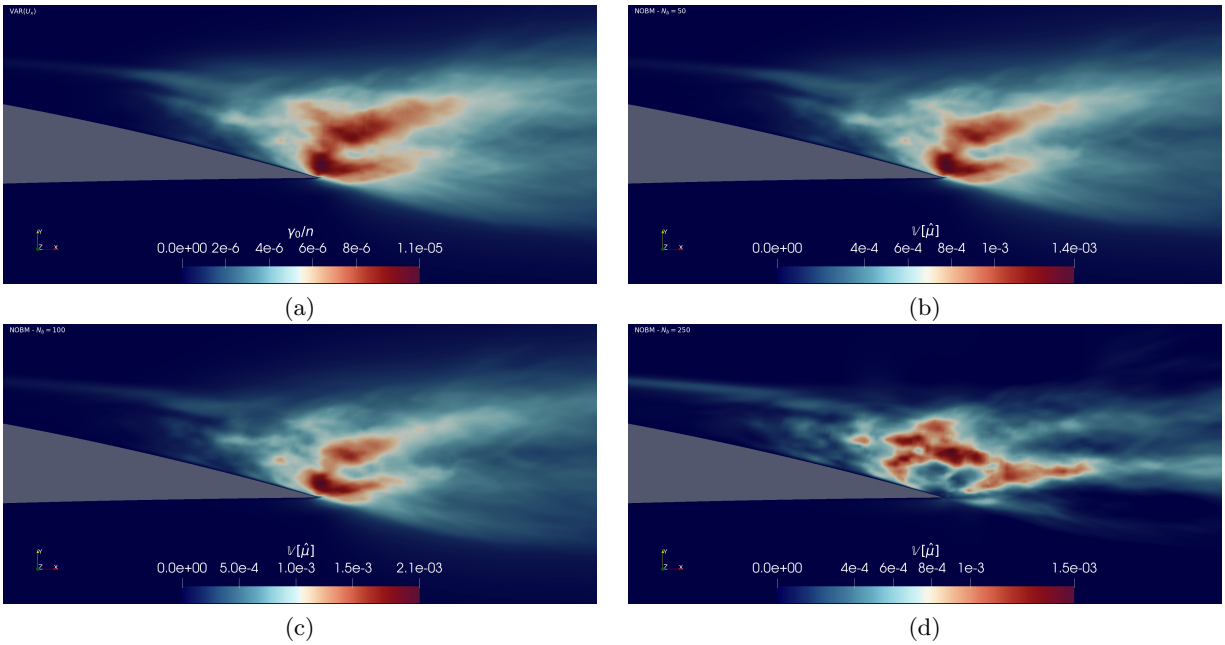


Figure 11: Contours of the (a) sample variance of the streamwise velocity component u of the NACA4412 wing simulation at $z = 0.025$ plane, and (b)-(d) estimated variance of associated SME using the iBMBC method with the batch size equal to (b) 50, (c) 100, and (d) 250.

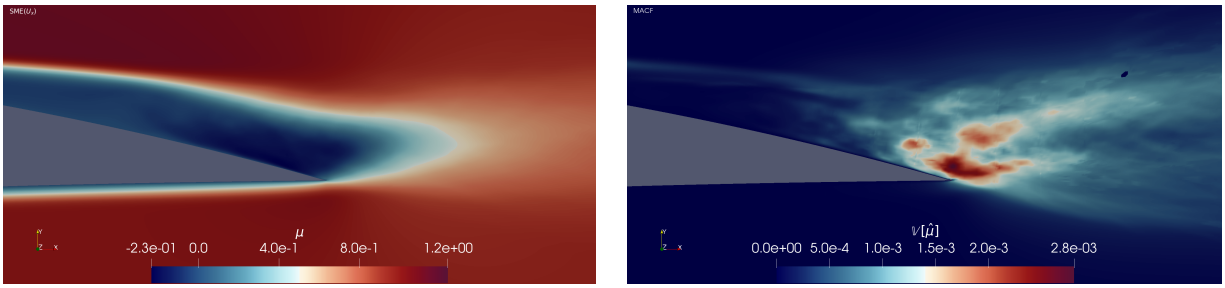
multiple computing nodes that are equipped with 128 physical cores and 256 GB of memory each. Based on previous experiences with ParaView Catalyst, we compiled a small edition of the package with only essential functionality that was necessary to execute the UQ framework excluding any rendering features. For the simulations, at all points of the 3D mesh, the updating ACF algorithm is applied every 20-th time step ($T_s = 20$) and with a maximal training lag $m_{\max} = 400$ resulting in 20 different training lags $\mathbf{m}_{\text{train}}$ to be cached. These settings can lead to accurate uncertainty estimates in most practical cases as shown in the previous sections. The computing time without data I/O and the maximal memory consumption throughout the simulation are measured for different number of MPI ranks, N_p, normalized by $N_{p,ref} = 128$. The time required to initialize the simulation and to write final results to disk, as well as transition to statistically stationary turbulence were not considered in the tests. For each test we ran six simulations, three runs with 1000 and three runs with 2000 time steps, based on which the computational performance measures with and without UQ were computed as the average time spent per each simulation time step. In addition, the stand-alone performance of the implementation of Eq. (3) in Python was investigated offline independent of any CFD code but for the same data matrix sizes provided by the CFD test case.

6.5.1. Analytical estimation of the memory consumption

A lower estimate for the memory consumption of the updating algorithm 1 can be derived by considering the sizes of the three arrays $M_{i,j}$, Γ^m_{ij} and the values of a time series that have to be buffered during runtime. For a single flow variable and N spatial points in the mesh, the memory consumption per rank M_r can then be estimated by,

$$M_r = 3N(m_{\max}/T_s)S_B\,, \tag{23}$$

with $m_{\max}$ being the maximum lag, T_s the sampling frequency and S_B the size of the floating point values, usually 8 bytes. In addition, there is a small memory overhead due to the scalar variables and Python objects that Eq. (23) does not account for. However, for all investigated configurations the overhead is in the order of 1 MB and can be neglected compared to the data arrays. Hence, the maximum lag as well as the sampling interval of the UQ algorithm have a major influence on the memory consumption. In simulation



(a) (b)

Figure 12: Contours of (a) the SME of streamwise velocity u and (b) associated variance estimated by the iMACF method for the NACA4412 wing use case.

cases with large local matrices, these two parameters can be reduced in order to save memory but up to a limit that the uncertainty estimates are still accurate, see Section 6.1.

6.5.2. Strong scaling

For the strong scaling tests of a fixed problem size and increasing number of MPI ranks, see Figure 14(a), we observe a super-linear scaling for Nek5000 simulations with and without including the updating UQ framework for all investigated numbers of MPI processors. This behavior can be explained by the very small size of the local matrices due to the large number of MPI ranks used. In these cases, the matrix sizes can be on the order of the processor's cache size leading to an increase in performance. For large local mesh sizes above around 10000 points per MPI rank, the entire in-situ framework including Nek5000, the catalyst interface and our UQ code, leads to only a small increase in the overall computing time (less than 5% compared to the UQ-excluded case). However, for smaller local meshes the framework does not scale linearly. At $N_p/N_{p,ref} = 32$ and around 7000 GLL points per rank, the in-situ UQ framework adds an additional computing time of approximately 11%. This is most likely due to intermediate steps required for interpolating the data at the GLL points and transferring the mesh from Nek5000 into VTK objects and then to Numpy arrays. Therefore, we expect that the performance can be further increased by implementing Eq. (16) directly into the CFD solver and working on the same data objects as provided there.

An analysis of the memory consumption in the strong scaling tests is shown in Figure 14(b). We observe an almost constant overhead of around 160 MB taken up by the Catalyst library with VTK that does not decrease with an increasing number of MPI ranks as the VTK library has to be loaded with each process. The updating iMACF algorithm 1 itself scales well up to large number of MPI ranks. As compared to the memory consumption of the pure Nek5000 simulations, the iMACF algorithm leads to a 20% increase in memory requirement (144.8 MB per rank) at $N_p/N_{p,ref} = 1$ that goes down to 3% (4.15 MB per rank) at $N_p/N_{p,ref} = 32$.

6.5.3. Weak scaling

In a second study, we evaluate the performance of the in-situ UQ framework for an increasing problem size with a constant number of 42000 GLL points per rank. In order to vary the problem size for these weak scaling tests, we gradually increase the order of the polynomials in each spatial dimension in the spectral elements from 5 to 17 alongside with increasing the number of MPI ranks from 54 to 1458. This increases the total number of GLL points from 2.27 million to 61.2 million for changing the polynomials order from 5 to 17. The computing time is increased by around 3% after adding the in-situ UQ framework, which is similar to what we observed in the strong scaling tests. Both memory consumption per rank and the additional computing time of the framework remain approximately constant over all investigated core counts. This agrees with the expectation, as the updating in-situ algorithm acts only on the local mesh and does not require any intense communication between ranks that would cause a drop in the performance.

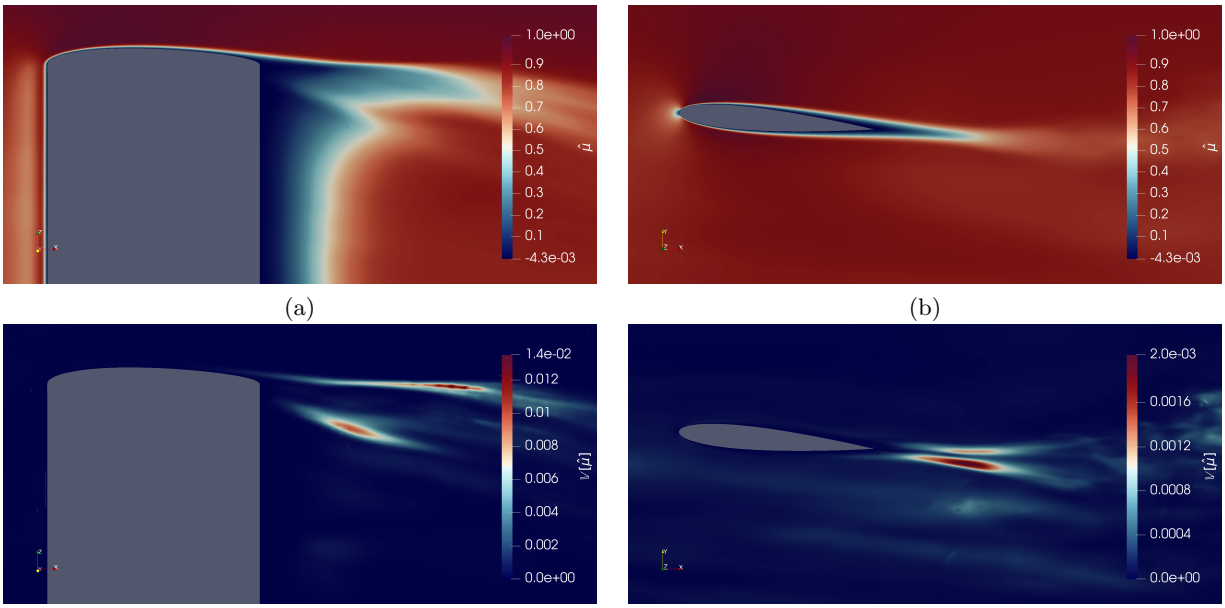


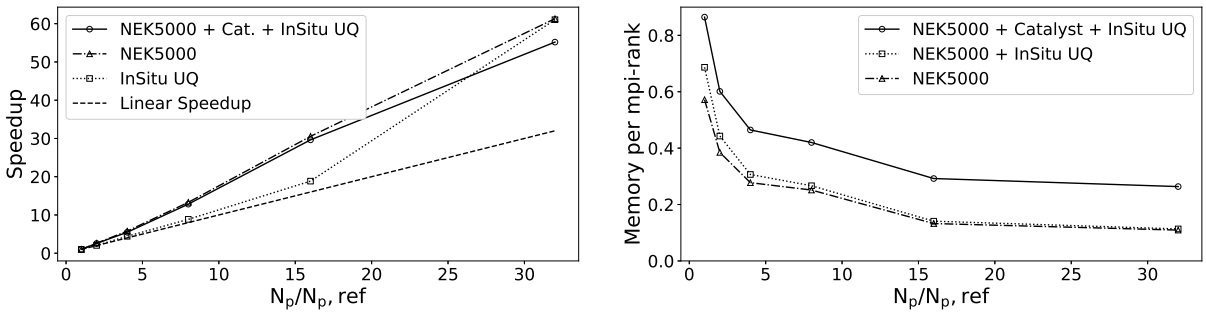
Figure 13: Contours of the SME of the streamwise velocity u of the rotor case at (a) $y = 0$ and (b) $z = 6$ planes. Corresponding variance of the SMEs obtained from the iMACF method with $M_{\max} = 20$ are plotted in (c) and (d), respectively.

7. Summary and Conclusions

This work introduces a novel framework for in-situ (online/streaming/updating) estimation of time-averaging uncertainties in turbulence statistics. This development can be of interest for large-scale simulations of turbulent flows, as it allows for monitoring such uncertainties on-the-fly without any need to store potentially large amounts of time series data on disk. A main characteristic of the turbulence time series (signals) considered here is that the samples are autocorrelated up to a generally unknown time lag.

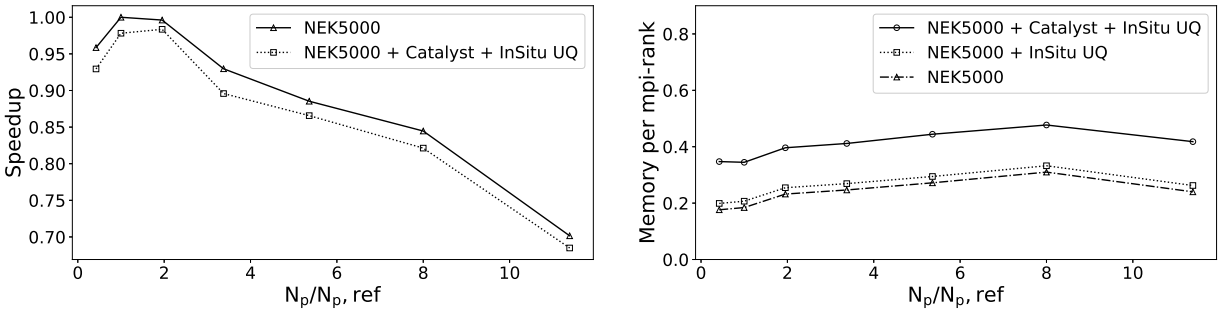
On the algorithmic side, the present work introduces a streaming formulation for the batch-based methods to estimate time-averaging uncertainties, see Section 3.2. To remove the natural shortcoming of such approaches, that is the bias in the estimates with respect to the batch size (a user's predefined parameter), we propose a new function, Eq. (6), to accurately model the autocorrelation function (ACF) of time series using a limited number of sample-estimated ACFs. The resulting modeled ACF is smooth and removes oscillations in the sample ACFs at high time lags. Furthermore, we propose an updating formula to evaluate the training sample-estimated ACFs. These two developments are then integrated into the iMACF method for in-situ estimation of time-averaging uncertainties, see Section 3.3, where only a minimal memory usage is required. A versatile and computationally efficient workflow based on VTK tools is designed and implemented to link an arbitrary CFD solver to the UQ module supplied with the proposed iMACF estimator, see Section 4, as well as the updating versions of the non-overlapping batch means (NOBM) and batch-means batch-correlation (BMBC) [27] methods. It is also possible to implement the in-situ Algorithm 1 (iMACF method) directly into a CFD solver without any interface.

The accuracy and validity of the MACF and iMACF estimators are thoroughly discussed in Sections 6.1 to 6.3 using the data from a first-order autoregressive model (ARM) and a fully resolved simulation of a turbulent channel flow. According to Figure 7, for statistically stationary time series of turbulent channel flow (equivalent to high-order ARMs) the proposed ACF model (6) results in much more accurate modeled ACFs compared to the existing function (5) used in literature. Moreover, the ACF modeled by function (6) is found to be robust with respect of the choice of the training sample-estimated ACFs, see Figure 6. The accuracy of the MACF method is validated by synthetic data, Figure 8, and compared to the power-law and



(a) (b)

Figure 14: Variation of the (a) speedup and (b) memory consumption (GB) with the number of cores in the strong scaling test. The NACA4412 wing use case is considered with $N_{p,ref} = 128$.



(a) (b)

Figure 15: Variation of the (a) speedup and (b) memory consumption (GB) with the number of cores in the weak scaling test. The NACA4412 wing use case is considered with $N_{p,ref} = 128$.

batch-based methods for turbulent channel flow data, see Figure 9. The detailed investigations represented in Figure 10 show a negligible influence of the hyperparameters of the iMACF method on the estimated uncertainties as compared to the offline MACF estimator.

In a final step, our new software is successfully applied to a number of more involved use cases including the turbulent flow over a circular cylinder and a wing with structured mesh, as well as a rotor simulated with adaptive mesh refinement. In all cases, accurate estimation of uncertainties was obtained by the iMACF method, without any bias that would exist upon using in-situ batch-based methods, see Figures 11 to 13. The detailed analyses in Section 6.5, including the weak and strong scaling tests, prove the efficiency of the framework from both computational time and memory usage aspects.

Although the focus of the present framework is on turbulent flow simulations, the methodology and framework are completely general, thus applicable to statistically stationary time series produced in any application, including laboratory experiments performed using e.g. hotwire anemometry or particle image velocimetry (PIV). Future extension of the present framework can be towards including other relevant in-situ data analysis techniques.

CRediT author statement

SR and **CG**: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing–Original Draft, Visualization. **AP**: Investigation, Writing–Original Draft. **JG**: Writing–Review & Editing, Supervision, Funding acquisition. **PS**: Conceptualization, Writing–Review & Editing, Funding acquisition.

Acknowledgments

This work has been supported by the EXCELLERAT project which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823691. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Germany, Italy, Slovenia, Spain, Sweden, and France under grant agreement No 101092621. The authors would like to thank Dr. Marco Atzori from The Polytechnic University of Milan, Italy, for providing the Nek5000 case for the wing simulation, and Dr. Antonio Memmolo from CINECA, Italy, for sharing the data of the "toy" rotor simulation. The computation of the rotor case was enabled by resources provided by the Swedish National Infrastructure for Computing (SNIC) at the PDC Center for High Performance Computing, KTH Royal Institute of Technology, partially funded by the Swedish Research Council through grant agreement no. 2018-05973. The simulations of the NACA4412 wing case were carried out on Hawk at the High Performance Computing Center Stuttgart (HLRS). Funding for Hawk was provided by Baden-Württemberg Ministry for Science, Research, and the Arts and the German Federal Ministry of Education and Research through the Gauss Centre for Supercomputing (GCS).

Appendix A. The Power-law Method to Model an ACF

A smooth model for ACF, to be plugged into Eq. (3), can be obtained by first fitting an autoregressive model (ARM) of order p to samples $\{x'_i\}_{i=1}^n$ where $x'_i = x_i - \hat{\mu}_x$ [34]:

$$x'_i = \sum_{j=1}^{p} \alpha_j x'_{i-j} + \epsilon_i \,. \tag{A.1}$$

The noise term is Gaussian, $\epsilon_i \sim \mathcal{N}(0, \sigma_d^2)$ where the standard deviation σ_d and the coefficients $\{\alpha_j\}_{j=1}^p$ can be obtained by various methods, see [34]. The starting point for deriving a smooth model for ACF is the Yule-Walker equation, see [34]:

$$\rho_k = \sum_{j=1}^{p} \alpha_j \rho_{k-j} \,, \quad k = 1, 2, \cdots, p \,, \tag{A.2}$$

where ρ_k is the autocorrelation of x at lag k. Plugging the ansatz $\rho_k = \lambda^k$ in this equation leads to an algebraic characteristic equation with roots $\{\lambda_k\}_{k=1}^p$ by which the ACF at lag k can be expanded as,

$$\rho_k = \sum_{i=1}^{p} c_i \lambda_i^k \,. \tag{A.3}$$

To compute the coefficients $\{c_i\}_{i=1}^p$ from a linear system of equations, a set of K_{ACF} sample-estimated ACF at lower lags (to avoid the issue of wiggles at high lags) are considered in Eq. (A.3). This results in a smooth model for the ACF at any lag $k \geq 0$. The utilization of modeled ACF in Eq. (3) is referred to as the ARM-based or power law uncertainty estimation method. The hyperparameters of this method are the ARM order p and the set of training sample-estimated ACFs with size K_{ACF}.

Appendix B. Reduction of Eq. (16) to Eq. (10)

By setting $m = 0$, i.e. lag zero in Eq. (16), we can recover Eq. (10). The main steps of the derivation are given below.

$$\begin{aligned}
\Gamma^0_{i,j} &= \Gamma^0_{i,j-1} - \Delta M_{i,j} \sum_{k=i}^{j-1} 2x_k + (j-i+1)M^2_{i,j} - (j-i)M^2_{i,j-1} + x_j^2 - 2x_j M_{i,j} \\
&= \Gamma^0_{i,j-1} - 2(j-i)\Delta M_{i,j} M_{i,j-1} + (j-i)M^2_{i,j} + (M_{i,j} - x_j)^2 - (j-i)M^2_{i,j-1} \\
&= \Gamma^0_{i,j-1} - 2(j-i)\Delta M_{i,j} M_{i,j-1} + (j-i)(M_{i,j-1} + \Delta M_{i,j})^2 + (j-i)^2 \Delta M^2_{i,j} - (j-i)M^2_{i,j-1} \\
&= \Gamma^0_{i,j-1} + (j-i)(j-i+1)\Delta M^2_{i,j} \,,
\end{aligned} \tag{B.1}$$

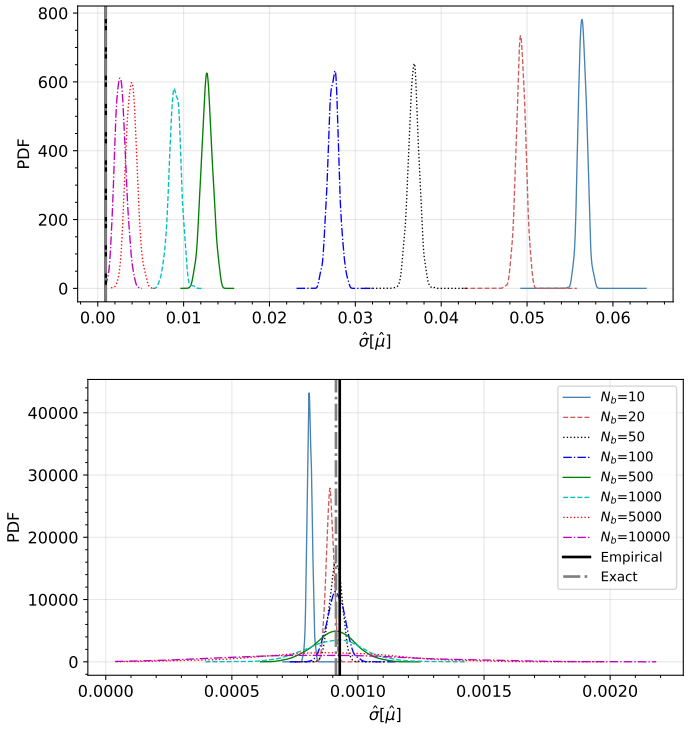


Figure C.16: Impact of the batch size on the estimated $\hat{\sigma}(\hat{\mu})$ for the synthetic samples generated by the ARM(1) defined in Eq. (17) for (top) NOBM and (bottom) BMBC methods. The PDFs are obtained by 1000 repetitions of independent simulations with the number of samples n in each simulation being equal to 10^5.

where $\Gamma^0_{i,j-1} = S_{i,j-1}$.

Appendix C. Impact of the Batch Size on the Batch-based Uncertainty Estimators

The batch-based methods are the most frequently used approaches for estimating time-averaging uncertainties in autocorrelated turbulence time series. However, the batch size introduces a bias in such estimated uncertainties [36]. This is shown here by adopting the procedure described in Section 6.2 with the samples generated from the first-order ARM defined in Eq. (17). Figure C.16 represents the PDF of the SME's uncertainties obtained by the non-overlapping Batch Means (NOBM) and the Batch-Means Batch-Correlation (BMBC) methods proposed in Ref. [27]. In both plots, the empirical and exact uncertainties in the SME are also shown. The bias introduced by the batch size is clearly more evident for the NOBM method, but it is interesting that even for ARM(1), the BMBC method is still affected. Another important observation is that for the NOBM method, the confidence in the estimated $\hat{\sigma}(\hat{\mu})$ is not changing significantly with the variation of the batch size, but for the BMBC method, the confidence in the estimations is reduced as the batch size increases. The latter is due to the reduction of the number of batches as the batch size increases while the total number of available samples is fixed. This issue is the main barrier to the use of batch-based methods in an in-situ way, noting that no valid estimates can be made for the optimal batch size for the time series prior to the simulations. Furthermore, any preset batch size may not be applicable for all quantities of interest and the spatial points in a region of interest, noting the dependence of the time series and associated ACF to these factors.

References

[1] M. Atzori, W. Köpp, W. Chien, D. Massaro, F. Mallor, A. Peplinski, M. Rezaei, N. Jansson, S. Markidis, R. Vinuesa, E. Laure, P. Schlatter, and T. Weinkauf. In-situ visualization of large-scale turbulence simulations in Nek5000 with paraview catalyst. *The Journal of Supercomputing*, 78, 02 2022. doi: 10.1007/s11227-021-03990-3.

[2] A. C. Bauer, B. Geveci, and W. Schroeder. The ParaView Catalyst User's Guide, 2019. https://www.mn.uio.no/astro/english/services/it/help/visualization/paraview/paraviewcatalystguide-5.8.1.pdf.

[3] C. Burstedde, L. Wilcox, and O. Ghattas. `p4est`: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. *SIAM J. Sci. Comput.*, 33(3):1103–1133, 2011. doi: 10.1137/100791634.

[4] T. F. Chan, G. H. Golub, and R. J. LeVeque. Algorithms for computing the sample variance: Analysis and recommendations. *The American Statistician*, 37(3):242, Aug. 1983. doi: 10.2307/2683386. URL https://doi.org/10.2307/2683386.

[5] H. Choi and P. Moin. Grid-point requirements for large eddy simulation: Chapman's estimates revisited. *Physics of Fluids*, 24(1):011702, 2012. doi: 10.1063/1.3676783. URL https://doi.org/10.1063/1.3676783.

[6] S. Deck, F. Gand, V. Brunet, and S. Ben Khelil. High-fidelity simulations of unsteady civil aircraft aerodynamics: stakes and perspectives. application of zonal detached eddy simulation. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 372(2022):20130325, 2014. doi: 10.1098/rsta.2013.0325. URL https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2013.0325.

[7] G. K. El Khoury, P. Schlatter, A. Noorani, P. F. Fischer, G. Brethouwer, and A. V. Johansson. Direct numerical simulation of turbulent pipe flows at moderately high Reynolds numbers. *Flow Turbulence Combust.*, 91:475–495, 2013.

[8] P. F. Fischer, G. W. Kruse, and F. Loth. Spectral element methods for transitional flows in complex geometries. *J. Sci. Comput.*, 17(1-4):81–98, Dec. 2002. ISSN 0885-7474. doi: 10.1023/A:1015188211796. URL http://dx.doi.org/10.1023/A:1015188211796.

[9] P. F. Fischer, J. W. Lottes, and S. G. Kerkemeier. Nek5000 Web page, 2008. http://nek5000.mcs.anl.gov.

[10] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

[11] J. Jiménez. Near-wall turbulence. *Physics of Fluids*, 25(10):101302, 2013. doi: 10.1063/1.4824988. URL http://dx.doi.org/10.1063/1.4824988.

[12] Y. Ju, A. Perez, S. Markidis, P. Schlatter, and E. Laure. Understanding the impact of synchronous, asynchronous, and hybrid in-situ techniques in computational fluid dynamics applications. In *2022 IEEE 18th International Conference on e-Science (e-Science)*. IEEE, Oct. 2022. doi: 10.1109/escience55777.2022.00043. URL https://doi.org/10.1109/escience55777.2022.00043.

[13] A. Khan, H. Sim, S. Vazhkudai, A. Butt, and K. Youngjae. An analysis of system balance and architectural trends based on top500 supercomputers. pages 11–22, 01 2021. doi: 10.1145/3432261.3432263.

[14] D. H. Lenschow, J. Mann, and L. Kristensen. How long is long enough when measuring fluxes and other turbulence statistics? *Journal of Atmospheric and Oceanic Technology*, 11(3):661–673, June 1994. doi: 10.1175/1520-0426(1994)011⟨0661:hlilew⟩2.0.co;2. URL https://doi.org/10.1175/1520-0426(1994)011<0661:hlilew>2.0.co;2.

[15] F. Liang, R. Shi, and Q. Mo. A split-and-merge approach for singular value decomposition of large-scale matrices. *Statistics and Its Interface*, 9:453–459, 01 2016. doi: 10.4310/SII.2016.v9.n4.a5.

[16] Y. Maday, C. Mavriplis, and A. T. Patera. Nonconforming mortar element methods - Application to spectral discretizations. In *Domain Decomposition Methods*, pages 392–418, 1989.

[17] C. Mavriplis. *Proceedings of the Eighth GAMM-Conference on Numerical Methods in Fluid Mechanics*, chapter A posteriori error estimators for adaptive spectral element techniques, pages 333–342. Vieweg+Teubner Verlag, Wiesbaden, 1990. ISBN 978-3-663-13975-1. doi: 10.1007/978-3-663-13975-1_34. URL http://dx.doi.org/10.1007/978-3-663-13975-1_34.

[18] E. Merzari, A. Obabko, P. Fischer, and M. Aufiero. Wall resolved large eddy simulation of reactor core flows with the spectral element method. *Nuclear Engineering and Design*, 364, 2020.

[19] J. Meyers, B. Geurts, and P. Sagaut, editors. *Quality and Reliability of Large-Eddy Simulations*. Springer, Netherlands, 2010.

[20] N. Offermans, O. Marin, M. Schanen, J. Gong, P. Fischer, P. Schlatter, A. Obabko, A. Peplinski, M. Hutchinson, and E. Merzari. On the strong scaling of the spectral element solver Nek5000 on petascale systems. In *Proceedings of the Exascale Applications and Software Conference 2016*, pages 1–10, 2016.

[21] N. Offermans, A. Peplinski, O. Marin, and P. Schlatter. Adaptive mesh refinement for steady flows in Nek5000. *Computers & Fluids*, 197:104352, 2020. ISSN 0045-7930. doi: https://doi.org/10.1016/j.compfluid.2019.104352. URL http://www.sciencedirect.com/science/article/pii/S0045793019303111.

[22] N. Offermans, D. Massaro, A. Peplinski, and P. Schlatter. Error-driven adaptive mesh refinement for unsteady turbulent flows in spectral-element simulations. *Computers & Fluids*, 251:105736, 2023. ISSN 0045-7930. doi: https://doi.org/10.1016/j.compfluid.2022.105736. URL https://www.sciencedirect.com/science/article/pii/S0045793022003280.

[23] T. Oliver, N. Malaya, R. Ulerich, and R. Moser. Estimating uncertainties in statistics computed from direct numerical simulation. *Physics of Fluids*, 26, 02 2014. doi: 10.1063/1.4866813.

[24] A. T. Patera. A spectral element method for fluid dynamics: laminar flow in a channel expansion. *Journal of computational Physics*, 54(3):468–488, 1984.

[25] S. Rezaeiravesh, R. Vinuesa, and P. Schlatter. UQit: A Python package for uncertainty quantification (UQ) in com-

putational fluid dynamics (CFD). *Journal of Open Source Software*, 6(60):2871, 2021. doi: 10.21105/joss.02871. URL https://doi.org/10.21105/joss.02871.

[26] S. Rezaeiravesh, R. Vinuesa, and P. Schlatter. An uncertainty-quantification framework for assessing accuracy, sensitivity, and robustness in computational fluid dynamics. *Journal of Computational Science*, 62:101688, 2022. ISSN 1877-7503. doi: https://doi.org/10.1016/j.jocs.2022.101688. URL https://www.sciencedirect.com/science/article/pii/S1877750322000941.

[27] S. Russo and P. Luchini. A fast algorithm for the estimation of statistical error in DNS (or experimental) time averages. *Journal of Computational Physics*, 347:328 – 340, 2017. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2017.07.005. URL http://www.sciencedirect.com/science/article/pii/S0021999117305077.

[28] P. Sagaut, S. Deck, and M. Terracol. *Multiscale and Multiresolution Approaches in Turbulence: LES, DES and Hybrid RANS/LES Methods : Applications and Guidelines*. Imperial College Press, 2013. ISBN 9781848169876. URL https://books.google.se/books?id=MzW6CgAAQBAJ.

[29] S. T. Salesky, M. Chamecki, and N. L. Dias. Estimating the random error in eddy-covariance based fluxes and other turbulence statistics: The filtering method. *Boundary-Layer Meteorology*, 144(1):113–135, Mar. 2012. doi: 10.1007/s10546-012-9710-0. URL https://doi.org/10.1007/s10546-012-9710-0.

[30] S. Seabold and J. Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

[31] J. P. Slotnick, A. Khodadoust, J. J. Alonso, D. L. Darmofal, W. D. Gropp, E. A. Lurie, and D. J. Mavriplis. CFD vision 2030 study: A path to revolutionary computational aerosciences, 2014.

[32] R. C. Smith. *Uncertainty Quantification: Theory, Implementation, and Applications*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2013. ISBN 161197321X, 9781611973211.

[33] R. Vinuesa, P. Negi, M. Atzori, A. Hanifi, D. Henningson, and P. Schlatter. Turbulent boundary layers around wing sections up to rec=1,000,000. *International Journal of Heat and Fluid Flow*, 72:86–99, 2018. ISSN 0142-727X. doi: https://doi.org/10.1016/j.ijheatfluidflow.2018.04.017. URL https://www.sciencedirect.com/science/article/pii/S0142727X17311426.

[34] W. Wei. *Time Series Analysis: Univariate and Multivariate Methods*. Pearson Addison Wesley, 2006. ISBN 9780321322166. URL https://books.google.se/books?id=aYOQAQAAIAAJ.

[35] B. P. Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962. doi: 10.1080/00401706.1962.10490022. URL https://www.tandfonline.com/doi/abs/10.1080/00401706.1962.10490022.

[36] D. Xavier, S. Rezaeiravesh, R. Vinuesa, and P. Schlatter. On the use of batch-means estimators for quantifying uncertainties in time averages of turbulence simulations. *To be submitted*, 2023.