Towards stochastic methods in CFD for engineering applications

Riccardo Tosi

Ph.D. thesis in Civil Engineering

Universitat Politècnica de Catalunya

Departament d'Enginyeria Civil i Ambiental

Supervisors: Prof. Riccardo Rossi
Dr. Jordi Pons-Prats

Barcelona, November 2021

I'll try and try again! I won't ever give up!
Look out across the moor – there's another omen!
And I'll remember it, poppa! There's always another rainbow!
Scrooge McDuck, The Life and Times of Scrooge McDuck
Don Rosa

Abstract

Recent developments of high performance computing capabilities allow solving modern science problems employing sophisticated computational techniques. However, it is necessary to ensure the efficiency of state of the art computational methods to fully take advantage of modern technology capabilities. In this thesis we propose uncertainty quantification and high performance computing strategies to solve fluid dynamics systems characterized by uncertain conditions and unknown parameters. We verify that such techniques allow us to take decisions faster and ensure the reliability of simulation results.

Different sources of uncertainties can be relevant in computational fluid dynamics applications. For example, we consider the shape and time variability of boundary conditions, as well as the randomness of external forces acting on the system. From a practical point of view, one has to estimate statistics of the flow, and a failure probability convergence criterion must be satisfied by the statistical estimator of interest to assess reliability. We use hierarchical Monte Carlo methods as uncertainty quantification strategy to solve stochastic systems. Such algorithms present three levels of parallelism: over levels, over realizations per level, and on the solution of each realization. We propose an improvement by adding a new level of parallelism, between batches, where each batch has its independent hierarchy. These new methods are called asynchronous hierarchical Monte Carlo, and we demonstrate that such techniques take full advantage of concurrency capabilities of modern high performance computing environments, while preserving the same reliability of state of the art methods. Moreover, we focus on reducing the wall clock time required to compute statistical estimators of chaotic incompressible flows. Our approach consists in replacing a single long-term simulation with an ensemble of multiple independent realizations, which are run in parallel with different initial conditions. The error analysis of the statistical estimator leads to the identification of two error contributions: the initialization bias and the statistical error. We propose an approach to systematically detect the burn-in time to minimize the initialization bias, accompanied by strategies to reduce the simulation cost. Finally, we propose an integration of Monte Carlo and ensemble averaging methods for reducing the wall clock time required for computing statistical estimators of time-dependent stochastic turbulent flows. A single long-term Monte Carlo realization is replaced by an ensemble of multiple independent realizations, each characterized by the same random event and different initial conditions. We consider different systems, relevant in the computational fluid dynamics engineering field, as realistic wind flowing around high-rise buildings or compressible potential flow problems. By solving such numerical examples, we demonstrate the accuracy, efficiency, and effectiveness of our proposals.

Key words: uncertainty quantification · Monte Carlo · multilevel Monte Carlo · asynchronous algorithms · ensemble averaging · statistical analysis · computational fluid dynamics · turbulent flows · high performance computing · distributed computing.

Resumen

Los desarrollos relacionados con la computación de alto rendimiento de las últimas décadas permiten resolver problemas científicos actuales, utilizando métodos computacionales sofisticados. Sin embargo, es necesario asegurarse de la eficiencia de los métodos computacionales modernos, con el fin de explotar al máximo las capacidades tecnológicas. En esta tesis proponemos diferentes métodos, relacionados con la cuantificación de incertidumbres y el cálculo de alto rendimiento, con el fin de minimizar el tiempo de computación necesario para resolver las simulaciones y garantizar una alta fiabilidad. En concreto, resolvemos sistemas de dinámica de fluidos caracterizados por incertidumbres.
En el campo de la dinámica de fluidos computacional existen diferentes tipos de incertidumbres. Nosotros consideramos, por ejemplo, la forma y la evolución en el tiempo de las condiciones de frontera, así como la aleatoriedad de las fuerzas externas que actúan sobre el sistema. Desde un punto de vista práctico, es necesario estimar valores estadísticos del flujo del fluido, cumpliendo los criterios de convergencia para garantizar la fiabilidad del método. Para cuantificar el efecto de las incertidumbres utilizamos métodos de Monte Carlo jerárquicos, también llamados *hierarchical Monte Carlo methods*. Estas estrategias tienen tres niveles de paralelización: entre los niveles de la jerarquía, entre los eventos de cada nivel y durante la resolución del evento. Proponemos agregar un nuevo nivel de paralelización, entre *batches*, en el cual cada *batch* es independiente de los demás y tiene su propia jerarquía, compuesta por niveles y eventos distribuidos en diferentes niveles. Definimos estos nuevos algoritmos como métodos de Monte Carlo asíncronos y jerárquicos, cuyos nombres equivalentes en inglés son *asynchronous hierarchical Monte Carlo methods*. También nos enfocamos en reducir el tiempo de computación necesario para calcular estimadores estadísticos de flujos de fluidos caóticos e incompresibles. Nuestro método consiste en reemplazar una única simulación de dinámica de fluidos, caracterizada por una ventana de tiempo prolongada, por el promedio de un conjunto de simulaciones independientes, caracterizadas por diferentes condiciones iniciales y una

ventana de tiempo menor. Este conjunto de simulaciones se puede ejecutar en paralelo en superordenadores, reduciendo el tiempo de computación. El método de promedio de conjuntos se conoce como *ensemble averaging*. Analizando las diferentes contribuciones del error del estimador estadístico, identificamos dos términos: el error debido a las condiciones iniciales y el error estadístico. En esta tesis proponemos un método que minimiza el error debido a las condiciones iniciales, y en paralelo sugerimos varias estrategias para reducir el coste computacional de la simulación. Finalmente, proponemos una integración del método de Monte Carlo y del método de *ensemble averaging*, cuyo objetivo es reducir el tiempo de computación requerido para calcular estimadores estadísticos de problemas de dinámica de fluidos dependientes del tiempo, caóticos y estocásticos. Reemplazamos cada realización de Monte Carlo por un conjunto de realizaciones independientes, cada una caracterizada por el mismo evento aleatorio y diferentes condiciones iniciales. Consideramos y resolvemos diferentes sistemas físicos, todos relevantes en el campo de la dinámica de fluidos computacional, como problemas de flujo del viento alrededor de rascacielos o problemas de flujo potencial. Demostramos la precisión, eficiencia y efectividad de nuestras propuestas resolviendo estos ejemplos numéricos.

Palabras clave: cuantificación de incertidumbre · Monte Carlo · Monte Carlo multinivel · algoritmos asincrónicos · promedio de un conjunto · análisis estadística · dinámica de fluidos computacional · flujos turbulentos · computación de alto rendimiento · computación distribuida.

Abstract

Gli sviluppi del calcolo ad alte prestazioni degli ultimi decenni permettono di risolvere problemi scientifici di grande attualità, utilizzando sofisticati metodi computazionali. È però necessario assicurarsi dell'efficienza di questi metodi, in modo da ottimizzare l'uso delle odierne conoscenze tecnologiche. A tal fine, in questa tesi proponiamo diversi metodi, tutti inerenti ai temi di quantificazione di incertezze e calcolo ad alte prestazioni. L'obiettivo è minimizzare il tempo necessario per risolvere le simulazioni e garantire alta affidabilità. Nello specifico, utilizziamo queste strategie per risolvere sistemi fluidodinamici caratterizzati da incertezze in macchine ad alte prestazioni.

Nel campo della fluidodinamica computazionale esistono diverse tipologie di incertezze. In questo lavoro consideriamo, ad esempio, il valore e l'evoluzione temporale delle condizioni di contorno, così come l'aleatorietà delle forze esterne che agiscono sul sistema fisico. Dal punto di vista pratico, è necessario calcolare una stima delle variabili statistiche del flusso del fluido, soddisfacendo criteri di convergenza, i quali garantiscono l'accuratezza del metodo. Per quantificare l'effetto delle incertezze sul sistema utilizziamo metodi gerarchici di Monte Carlo, detti anche *hierarchical Monte Carlo methods*. Queste strategie presentano tre livelli di parallelizzazione: tra i livelli della gerarchia, tra gli eventi di ciascun livello e durante la risoluzione del singolo evento. Proponiamo di aggiungere un nuovo livello di parallelizzazione, tra gruppi (*batches*), in cui ogni *batch* sia indipendente dagli altri ed abbia una propria gerarchia, composta da livelli e da eventi distribuiti su diversi livelli. Definiamo questi nuovi algoritmi come metodi asincroni e gerarchici di Monte Carlo, il cui corrispondente in inglese è *asynchronous hierarchical Monte Carlo methods*. Ci focalizziamo inoltre sulla riduzione del tempo di calcolo necessario per stimare variabili statistiche di flussi caotici ed incomprimibili. Il nostro metodo consiste nel sostituire un'unica simulazione fluidodinamica, caratterizzata da un lungo arco temporale, con il valore medio di un insieme di simulazioni indipendenti, caratterizzate da diverse condizioni iniziali ed un arco temporale minore. Questo insieme di simulazioni può essere eseguito in parallelo in un supercomputer, riducendo il tempo di calcolo. Questo metodo è noto come media di un insieme o, in inglese, *ensemble averaging*. Calcolando la stima di variabili statistiche, commettiamo due errori: l'errore dovuto alle condizioni iniziali e l'errore statistico. In questa tesi proponiamo un metodo per minimizzare l'errore dovuto alle condizioni iniziali, ed in parallelo suggeriamo diverse strategie per ridurre il costo computazionale della simulazione. Infine, proponiamo un'integrazione del metodo di Monte Carlo e del metodo di *ensemble averaging*, il cui obiettivo è ridurre il tempo di calcolo necessario per stimare variabili statistiche di problemi di fluidodinamica dipendenti dal tempo, caotici e stocastici. Ogni realizzazione di Monte Carlo è sostituita da un insieme di simulazioni indipendenti, ciascuna caratterizzata dallo stesso evento casuale, da differenti condizioni iniziali e da un arco temporale minore. Consideriamo e risolviamo differenti sistemi fisici, tutti rilevanti nel campo della fluidodinamica computazionale, come per esempio problemi di flusso del vento attorno a grattacieli, o sistemi di flusso potenziale. Dimostriamo l'accuratezza, l'efficienza e l'efficacia delle nostre proposte, risolvendo questi esempi numerici.

Parole chiave: quantificazione di incertezze · Monte Carlo · Monte Carlo multilivello · algoritmi asincroni · media di un insieme · analisi statistica · fluidodinamica computazionale · flussi turbolenti · calcolo ad alte prestazioni · calcolo distribuito.

Acknowledgements

First of all, I want to thank the ExaQUte (EXAscale Quantification of Uncertainties for Technology and Science Simulation) project and the Spanish Ministry of Economy and Competitiveness, which financially supported this doctorate through the grant agreement No 800898 and the "Severo Ochoa Programme for Centres of Excellence in R&D" (CEX2018-000797-S), respectively. I acknowledge the computer resources at MareNostrum and the technical support provided by Barcelona Supercomputing Center (FI-2019-1-0041, IM-2019-2-0008, IM-2019-3-0007, IM-2020-1-0016, IM-2020-2-0030, IM-2021-2-0007). The Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ (ID:90140) is also acknowledged.

My deepest gratitude goes to my supervisor, Prof. Riccardo Rossi, and my co-supervisor, Dr. Jordi Pons Prats. This work would not have been the same without the endless support of Prof. Riccardo Rossi, who taught me that it is always possible to find a solution to any engineering problem, even when everything is not working as it should or as expected. This happened during this thesis work, especially during the second year. I, therefore, thank Prof. Riccardo Rossi, who transmitted to me the passion for research and helped me grow both professionally and academically. I want to thank Dr. Jordi Pons Prats. The many conversations we had, the ideas we shared and our weekly meetings were an unmatched help in the accomplishment of this thesis. A special mention is for Prof. Javier Principe, with whom I had many important conversations during the last year of the doctorate.

The whole Kratos team deserves to be acknowledged. A special thank you goes to Marc Núñez, my office partner since the first day of my doctorate. We shared many moments during these three years, both professionally and personally. Thank you also to Carlos Roig (a.k.a. Charlie), you were always available to answer my infinite coding questions. I also want to thank Ramon Amela, Quentin Ayoul-Guilmard, Brendan Keith, Sundar Ganesh and Anopp Kodakkal, who supported the creation and development of the XMC library, with code development, weekly discussions, and sharing of ideas.

A special mention is for the many friends and colleagues I met at university during these years. Thank you to Kim, Gabriel, Raul, Agustina, Uxue, Nicola, Rubén, Massimiliano, Vicente, Sebastian, Edu, Lu, Iván and many more. We shared many coffees, lunches, vermuts, *pizze*, and (hopefully soon) *piadine*.

Contents

List of Figures

List of Tables

Acronyms

AMR	adaptive mesh refinement
CAARC . . .	Commonwealth Advisory Aeronautical Council
CDF	cumulative distribution function
CFD	computational fluid dynamics
CFL	Courant-Friedrichs-Lewy
CMLMC . . .	continuation multilevel Monte Carlo
CPU	central processing unit
CVaR	Conditional Value at Risk
DE	discretization error
e.g.	exempli gratia
FE	finite element
HPC	high performance computing
i.e.	id est
Kratos	Kratos Multiphysics
LES	large eddy simulation
MC	Monte Carlo
MLMC	multilevel Monte Carlo

MPI	message passing interface
NS	Navier-Stokes
OpenMP . .	open multi-processing
PDF	probability density function
PMF	probability mass function
QoI	quantity of interest
Re	Reynolds number
SC	spatially correlated
SE	statistical error
SU	spatially uncorrelated
UQ	uncertainty quantification
VaR	Value at Risk
VMS	variational multiscale

Chapter 1

Introduction

1.1 Importance of estimating uncertainties

Since the beginning of time, humans have always tried to describe the surrounding nature. Nowadays, it is possible to describe complex physical systems using sophisticated mathematical models. However, several questions are still standing, and their associated research fields are active. Are these models accurate? Can we obtain the same amount of information faster? How can we describe our system if some data is not known or, better said, is uncertain?

Uncertainty refers to states, or systems, of which we have limited knowledge, with unknown or even multiple outcomes [68]. The field of science that studies how to characterize and reduce uncertainties is called uncertainty quantification (UQ). Uncertainties arise in environments that are *stochastic* or that can be only partially described by an observer. Uncertainties are an important factor in many different fields, spacing from statistics and economics to physics and engineering.

It is common to predict outcomes of uncertain systems using *probability*. As commented by Carlo Rovelli, when all of the details of a system are not available, we cannot know what will happen in the future. Probability is the tool we use to predict their output[1].

One of the most important achievements of the last century is the development of computational methods, which allow describing systems through mathematical models. Such developments are parallel to the development of high performance computing (HPC) capabilities of last decades. Nowadays we assume that the results of a single simulation

[1]C. Rovelli, *Helgoland*, Adelphi, Milano, 2020, p. 41.

run on HPC machines are accurate enough for engineering purposes. The interest of academia and industries is therefore moving into new directions. New topics of interest are, for example, the quantification of uncertainties on systems characterized by unknown data, the exploration of design alternatives for optimization purposes, or the development of computational methods that minimize the wall clock time required to solve simulations, to take decisions faster.

A large number of uncertain components are usually involved in the description of complex engineering systems since both geometry features and operational conditions are of difficult calibration and identification. It is therefore crucial for an engineer to quantify how these uncertainties affect the system they are dealing with, and this is exactly the purpose of this thesis.

This is also the objective of the European project ExaQUte (EXAscale Quantification of Uncertainties for TEchnology and science simulation), that supports this work. The ExaQUte project aims at "constructing a framework to enable uncertainty quantification and optimization under uncertainties in complex engineering problems, exploiting computational simulations on forthcoming exascale systems"[2].

1.2 Uncertainty quantification

We describe and quantify uncertainties within a probabilistic framework. This implies uncertainties can be considered as *random variables*. Therefore, given a system characterized by randomness, we can propagate uncertainties into the computational model of the system to assess their effect on the problem. Usually, some statistics of an output quantity of interest (QoI) of the system are estimated to assess the impact of uncertainties on the computational model. This operation is called forward UQ, and, in this work, we refer to it simply with UQ.

Different methods are available in the literature to perform UQ. Many of them, as stochastic Galerkin, stochastic collocation or polynomial chaos methods [57, 87, 138], suffer the *curse of dimensionality*, which means the convergence rate degrades as the number of random variables grows. In the most general scenario, for complex engineering problems, we may expect to deal with many uncertainties, thus with a stochastic space arbitrarily big. For this reason, in this work we decide to use Monte Carlo (MC) sampling-based approaches [59], that do not suffer the curse of dimensionality since their

[2]ExaQUte Consortium. (2021). ExaQUte. Retrieved from `http://exaqute.eu/`.

(a) Thai Airways Boeing 747-400 HS-TGB airfoil [97].
(b) 30 St Mary Axe in London, a modern high-rise building [9].

Figure 1.1: Examples of systems we consider in this thesis work.

convergence rate is independent of the number of random variables. These methods rely on the idea of solving different *realizations* of the same problem, each characterized by different random variable values. MC methods also are *non-intrusive*, which means the solver can be considered to be a black box. This is an important feature since it simplifies the integration of the MC methods and the developments of our work with other existing software. Moreover, the MC methods present the property that estimated statistics of a QoI converge to the *exact statistics* as the number of realizations tends to infinity, independently from the dimension of the stochastic space.

The main drawback of the standard MC method is its slow convergence rate, which implies a high computational cost and a high wall clock time are required to solve the problem. To improve this bottleneck, other methods have been derived from standard MC, as multilevel Monte Carlo (MLMC) [62, 64, 63, 58, 59] or continuation multilevel Monte Carlo (CMLMC) [38]. Both approaches exploit a *hierarchy* of *levels* of increasing accuracy to solve the problem under consideration. Using a large number of coarse realizations and only a few fine samples reduce the overall computational cost with respect to standard MC, where all realizations are run on the finest level. We remark that the name hierarchical MC methods comes from using a hierarchy of approximations to solve the system and we use it in this work to refer to the class of MC methods.

In literature, many applications of the MC methods already exist. These have been successfully applied to hyperbolic conservation laws with stochastic input data [103, 102], to elliptic partial differential equations [34], in engineering applications [101, 111, 23] and in software [26]. However, using MLMC for solving time-dependent partial differential equations with random parameters is rather uncommon. We refer for example to [122, 54], where stochastic differential equations are solved by extending the MLMC method.

The systems we aim at solving are computational fluid dynamics (CFD) problems. It is known that the simulation of highly turbulent flows represents a well-established challenge in CFD, with predictions becoming more difficult as the Reynolds number (Re) increases. This situation is explained by Kolmogorov's theory which establishes that turbulent flows are characterized by *multiple temporal and spatial scales*, with an energy transfer cascade from larger eddies to smaller ones [112]. According to the theory, the ratio between the largest and smallest length scales is proportional to $\mathrm{Re}^{3/4}$, while the ratio between the timescales is proportional to $\mathrm{Re}^{1/2}$. When multiple spatial scales are involved, issues may arise for discretizing the spatial domain, since different regions require different accuracy. Even though it is standard for multi-level MC methods (MC methods with more than one level, as MLMC and CMLMC) to use meshes with uniform discretization, adaptive mesh refinement (AMR) procedures based on error estimation techniques [1, 3, 2, 55] can be preferred over uniform meshes to describe multiple spatial scales. AMR permits to have a finer resolution where needed, and a coarser discretization elsewhere, thus saving computational resources.

We aim at exploring different strategies for integrating AMR with hierarchical MC algorithms. Few attempts are present in the literature, and we refer for example to [65, 51]. Moreover, in this work, we observe that the usability of standard MLMC and of CMLMC methods depends strongly on whether or not the underlying problem is time-dependent and chaotic in nature.

Finally, we comment that engineers normally take advantage of risk measures to take decisions [115]. Such quantities can for example be computed on top of statistics estimated by means of UQ analyses.

1.3 Distributed computation

Figure 1.2: The MareNostrum 4 supercomputer, located at the Barcelona Supercomputing Center [93].

In this work, we aim at developing an UQ framework for running on HPC systems, exploiting *concurrency* capabilities of modern supercomputers and of next-generation exascale supercomputers, that will reach hundreds of thousands of cores [7] and will follow a "mega-node, kilo-core, giga-hertz rule" [82].

Hierarchical MC algorithms are known to be highly parallelizable, since they present three levels of parallelism: over levels, over realizations per level and at solver level [48]. However, an efficient implementation of hierarchical MC algorithms for running on HPC systems is challenging. In fact, different aspects must be taken into account, as the scalability of the solver or the fact that tasks may be heterogeneous, which means their runtime may easily vary.

To solve such issues, a *dynamic* scheduling approach is exploited for running on supercomputers, since it provides higher adaptability and is faster than a *static* scheduling [48, 128].

Additionally, running in a distributed environment introduces *synchronization points* that deteriorate the computational efficiency of the algorithm. For this reason, in this thesis we introduce a new class of *asynchronous* hierarchical MC algorithms, which are specifically designed to run on modern supercomputers, maximizing the computational efficiency and exploiting distributed computations.

1.4 Ensemble averaging

When solving turbulent CFD problems, one has to estimate statistics of the flow, e.g. mean or variance quantities. Such estimations typically require very long simulations which include the initial transient dynamics, required for the flow to develop, followed by the effective dynamics, required for the estimator to converge. Unfortunately, despite decades of hardware improvements, such simulations require prohibitive runtimes. While the use of HPC systems may reduce these runtimes, practical limits exist on the achievable speedup for a given problem size. The most important feature controlling the runtime is that time evolution in a single simulation is intrinsically sequential [90].[3]

In order to control the bias associated to the initial conditions, the estimation of statistics of a turbulent flow entails collecting data starts only at some point in time after the flow has developed [85, 83], i.e. once the solution has been drawn to the attractor [120]. We refer to the discarded initial time interval as the *burn-in time* and the remainder as the *effective time.*

We explore the possibility of reducing the time to solution for solving CFD problems by exploiting *ensemble*-based approaches, that consist of estimating statistics by averaging over numerous independent simulations. This technique has been investigated in the literature in two different settings. In one setting [84], the focus is on reducing the wall clock time on constant hardware resources[4]. Approaches to this problem typically consist of solving linear systems with multiple right-hand sides [76, 75]. In the other setting, which our work considers, the focus is on exploiting the concurrency capabilities of HPC

[3]A potential solution could be parallel-in-time methods [56], which received much attention in the last years exactly due to their potential in providing a solution for the latter problem. Unfortunately, their application does not seem to be viable in chaotic problems [136], that is the class of problems we are interested in.

[4]For example, in [84] the author focuses "on attempt to speedup the simulations by increasing the efficiency of computations on the same hardware resources, discussing an idea of simultaneous modeling of multiple independent flow realizations".

systems [90]. Such approaches can be seen to target the direction of next-generation exascale computers. In addition, in this thesis, we extend ensemble-based approaches by presenting strategies for reducing the overall computational cost of the burn-in phase.

We are also interested in solving generic stochastic turbulent flow problems by exploiting ensemble averaging to minimize the runtime. Therefore, two possibilities can be considered for the integration of UQ and ensemble-based methods. One scenario is described in [89], where the authors propose an ensemble-based MC method that makes use of multiple right-hand sides to solve stochastic second-order parabolic partial differential equations. However, such an approach is intrusive, in the sense that it requires changes in software that are not designed to solve partial differential equations with multiple right-hand sides. In our work we propose another scenario, that consists of a non-intrusive integration of UQ and ensemble-based methods and exploits concurrency capabilities of modern HPC systems.

1.5 Aim

The aim of this thesis is to perform efficient and accurate uncertainty quantification of CFD problems. To reach this goal, the following objectives are set.

- Extend hierarchical MC methods to efficiently run on HPC environments and a new class of asynchronous hierarchical MC algorithms is introduced.
- Integrate hierarchical MC methods with AMR strategies for performing UQ.
- Introduce different strategies for minimizing the overall computational cost and the time to solution of ensemble-based methods.
- Integrate ensemble-based and hierarchical MC methods for solving time-dependent stochastic problems.

To achieve such objectives, the development of high performance and modular software in Python and C++ is required. We remark that all the implementations of this work must remain as open source software. In order to fulfil this requirement, the UQ framework is developed within the Kratos Multiphysics (Kratos) open source software [44, 43] and the XMC open source software [13].

In parallel, other goals are set. For example, we need to ensure that our software developments are compatible with open multi-processing (OpenMP) parallelism and with message passing interface (MPI) parallelism and we need to run simulations on multiple supercomputers.

1.6 Scientific production

In this section, we present all of the scientific production of this thesis work.

Publications

- In the article [128] we present the asynchronous hierarchical MC methods, we verify their computational efficiency when running on HPC systems and we perform a strong scalability test of our implementation.
- In the conference proceeding [131] we briefly apply the asynchronous MC method to solve problems of wind engineering interest.
- We comment that we have an article in preparation, which will cover the topics presented in chapters 8 and 9.

Conference presentations

- In the conference presentation [129] we first present the asynchronous hierarchical MC methods and some preliminary validation tests, run on HPC systems.
- In the conference presentation [81] we present shape-optimization problems with uncertainty in the data parameters. Two-dimensional benchmark problems are considered and solved.
- The conference presentation [130] is related to the conference proceeding [131], and we talk about the application of the asynchronous MC method to solve wind engineering problems on HPC systems.
- In the conference presentation [119] the statistical ensemble averaging framework is briefly presented and applied to solve problems of wind engineering interest on supercomputers.
- In the conference presentation [134] the ensemble-based MC method is discussed and applied to solve wind engineering problems on HPC systems.

Software Since the starting of this thesis, we have contributed to developing Kratos [95] and XMC [13].

- Kratos "is a framework for building parallel, multi-disciplinary simulation software, aiming at modularity, extensibility, and high performance".
- XMC "is a Python library for parallel, adaptive, hierarchical Monte Carlo algorithms, aiming at reliability, modularity, extensibility and high performance".

ExaQUte reports Within the ExaQUte project, we worked on different reports and we present here the main content of these deliverables, highlighting their connection with the objectives of this thesis.

- In [127] the first version of the UQ framework, developed within the Kratos software, is released.
- In [4] a first version of XMC is publicly released. Part of the UQ framework is moved from Kratos to XMC.
- In [10] the UQ framework containing asynchronous hierarchical MC methods is publicly released. The integration between Kratos, XMC and the remeshing software Mmg [45] is complete. Unit tests checking their integration are run on Kratos continuous integration.
- In [11] the integration of the MC method and ensemble averaging is publicly released.
- In [5] we profile the UQ framework by running different benchmarks on HPC environments using PyCOMPSs [88, 16, 126] as underlying programming model and dynamic scheduler of the tasks involved in the executions.
- In [14] we comment that the MLMC method can be used successfully for low Re flows when combined with AMR strategies. However, the hypotheses for optimal MLMC performance are found to not be satisfied at high turbulent Re despite the use of AMR strategies.
- In [15] we report on the use of the MLMC method for time-dependent problems. For time-dependent non-chaotic systems, optimal MLMC hypotheses are found to

be satisfied. For time-dependent chaotic cases with high Re the MLMC hypotheses are not satisfied.

- In [22] the UQ workflow is tested against a challenging wind engineering problem, that is wind flowing past a high-rise building.
- In [78] we present a simple and efficient strategy for AMR and a posteriori error estimation for the transient incompressible Navier-Stokes (NS) equations.
- In [132] we comment on the ensemble averaging method, its associated statistical framework and its applications for solving chaotic time-dependent fluid dynamics problems.
- In [133] we comment on the calibration of the ensemble-based MC method for wind engineering problems and on the identification of the most relevant parameters when solving stochastic chaotic time-dependent fluid dynamics problems.
- In [52] we profile the UQ framework (specifically the asynchronous MC method) by solving a high-rise building problem on HPC systems. We use PyCOMPSs [88, 16, 126] and Quake [33] as underlying programming models and Kratos as MPI parallel solver.
- In [12] we comment on the application of hierarchical MC methods to solve wind engineering problems, namely wind flowing past a high-rise twisted building. Specifically, we focus on the applicability of the multifidelity Monte Carlo method [109] to solve such a problem.

1.7 Contents

The remainder of the thesis is structured as follows. In chapter 2 we describe the CFD method and the wind model we use. In chapter 3 we introduce the probabilistic framework and we present different techniques to efficiently estimate statistics. In chapter 4 we discuss state of the art hierarchical MC methods together with considerations on their execution on HPC systems. In chapter 5 we introduce the asynchronous hierarchical MC methods and we verify the computational efficiency of these methods. In chapter 6 we introduce AMR strategies and we propose our integration of AMR and MLMC. In chapter 7 we apply hierarchical MC method to solve stochastic problems of wind engineering interest. In chapter 8 we introduce the statistical ensemble averaging framework, together with techniques to accelerate the burn-in phase. We compare ensemble averaging and standard time averaging by solving problems of wind engineering interest. In chapter 9 we present the ensemble-based MC method and we calibrate such a method for the class of wind engineering problems considered in this thesis. Concluding remarks close the thesis in chapter 10.

Chapter 2

Computational fluid dynamics foundations

In this chapter, we describe the implicit large eddy simulation (LES) model we use for simulating incompressible fluid flows. The NS equations describing fluid systems are introduced and summarized in section 2.1. The NS equations are numerically solved by applying the variational multiscale (VMS) method [69, 70], and their numerical formulation is presented in section 2.2. The final discrete problem is discussed in section 2.3. Moreover, we describe in section 2.4 the wind model we apply for describing wind flows.

We remark that the probabilistic framework we introduce in next chapters treats the solver as a black box. This implies that systems, models and methods different from the ones discussed in this chapter can be employed as well, as in the case of section 5.3.1. The CFD model we describe in this chapter is used in chapters 5 and 7–9.

We acknowledge that the implementation of the NS equations described in sections 2.1–2.3 and of the wind model of section 2.4 within the Kratos software has been done by colleagues working in the Kratos software and the ExaQUte project. We refer to [95, 79, 80] for details.

The content of this chapter is taken from a manuscript in preparation and is adapted wherever needed.

2.1 The Navier-Stokes problem

We restrict ourselves to incompressible flows, which are described by the incompressible NS equations and cover a wide range of applications in CFD. The strong form of such a problem consists of finding a velocity field $\boldsymbol{u}$ and a pressure field p, defined for a time window $[0,T]$ in a bounded domain $D \subseteq \mathbb{R}^d$, where $d = 2,3$ is the domain space dimensions. The system of equations reads

$$\begin{aligned} \partial_t \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} - \nu \Delta \boldsymbol{u} + \nabla p &= \boldsymbol{f} \qquad \text{in } [0,T] \times D \\ \nabla \cdot \boldsymbol{u} &= 0 \qquad \text{in } [0,T] \times D, \end{aligned} \tag{2.1}$$

where $\boldsymbol{f}$ is the force vector, ν the kinematic viscosity and, as usual, we denote vectors and tensors using bold characters. These equations must be complemented with appropriate boundary conditions, which can be of Dirichlet type, i.e. $\boldsymbol{u} = \boldsymbol{u}_g$ applied on a boundary Γ_{D}, or of Neumann type, i.e. $(-p\mathbf{I} + \nu \nabla \boldsymbol{u}) \cdot \boldsymbol{n} = \boldsymbol{t}_N$ applied on a boundary Γ_{N}, for all $t \in [0,T]$. In this chapter we consider $\partial D = \Gamma_{\mathrm{D}}$ and $\boldsymbol{u}_g = 0$ to simplify the notation, but in general $\Gamma_{\mathrm{D}} \cup \Gamma_{\mathrm{N}} = \partial D$ and $\Gamma_{\mathrm{D}} \cap \Gamma_{\mathrm{N}} = \emptyset$, that is the case of the numerical experiments of following chapters. Equation (2.1) must be supplemented with appropriate initial conditions. For example, in section 8.1.3 we propose two strategies to generate initial conditions.

As usual, we denote by $L^p(D)$, $1 \leq p < \infty$, the spaces of functions whose p-th power is Lebesgue integrable in D. The space of functions whose first-order distributional derivatives are in $L^2(D)$ and have zero trace on ∂D is denoted by $H_0^1(D)$ and its topological dual by $H^{-1}(D)$. We write $(\cdot,\cdot)$ to denote the integral over D of the product of any two functions f and g, whenever it makes sense. When the integral is computed in a region $\omega \subseteq D$, we will denote it with $(\cdot,\cdot)_\omega$. Given a Banach space X, $L^p(0,T;X)$ denotes the space of functions whose X-norm is in $L^p(0,T)$ whereas $\mathcal{D}'(0,T;X)$ denotes the space of distributions in time with values in X.

The weak form of the NS problem consists of finding $\boldsymbol{u} \in L^2(0,T;V)$ and $p \in \mathcal{D}'(0,T;Q)$ such that [39]

$$\begin{aligned} (\partial_t \boldsymbol{u}, \boldsymbol{v}) + (\boldsymbol{u} \cdot \nabla \boldsymbol{u}, \boldsymbol{v}) + \nu(\nabla \boldsymbol{u}, \nabla \boldsymbol{v}) - (p, \nabla \cdot \boldsymbol{v}) &= (\boldsymbol{f}, \boldsymbol{v}) \qquad \text{for all } \boldsymbol{v} \in V \\ (q, \nabla \cdot \boldsymbol{u}) &= 0 \qquad \text{for all } q \in Q, \end{aligned} \tag{2.2}$$

where $V = H_0^1(D)^d$ and $Q = L_0^2(D) := L^2(D)/\mathbb{R}$ (L^2 functions with zero mean).

2.2 The variational multiscale method

The discrete form of equation (2.2) is obtained using linear triangles and tetrahedra elements in the framework of the VMS method, which incorporates LES concepts in the numerics, giving an implicit LES method for the simulation of turbulent incompressible flows. The VMS method was originally introduced in [69, 70] as a framework for the development of stabilization methods, which are designed to overcome the two main problems of the numerical approximation of equation (2.2). The first one is the compatibility required between the velocity and pressure spaces which need to satisfy an *inf-sup condition* [25] to guarantee the stability of the approximation. The second one is the lack of robustness of the Galerkin method in the advection-dominated regime. There are many VMS methods and we refer for example to [36] for details.

The starting point of VMS formulations is a splitting of the solution space as $V = V_h \oplus \widetilde{V}$, into a finite element (FE) space V_h and a space of subgrid scales $\widetilde{V}$. The FE space V_h is built on top of a partition $\mathcal{T}_h$ of the domain D and is used to represent resolvable scales. In this way, a function $\boldsymbol{u} \in V$ is decomposed as $\boldsymbol{u} = \boldsymbol{u}_h + \widetilde{\boldsymbol{u}}$. The same splitting can be considered for the pressure space although this is not necessary to develop stable methods and the simplest approach is to consider $\widetilde{p} = 0$, that is what we assume. Alternatively, a model for the pressure subscale depending on the velocity divergence is commonly used, see e.g. [40, 74].

The VMS decomposition of the test function $\boldsymbol{v}$ in equation (2.2) gives rise to an equation for the resolved scales (tested by $\boldsymbol{v}_h$) and an equation for the fine scales (tested by $\widetilde{\boldsymbol{v}}$). However, these two equations are coupled and, because the space $\widetilde{V}$ is infinite-dimensional, some modeling assumptions for the fine scale equation are required to close the system. This modeling step is the algebraic approximation of the differential operator acting on the fine scales, and the terms involving subscales are integrated by parts within each element $K \in \mathcal{T}_h$. Using the notation

$$(\cdot,\cdot)_h = \sum_{K\in\mathcal{T}_h} (\cdot,\cdot)_K \quad \text{and} \quad (\cdot,\cdot)_{\partial h} = \sum_{K\in\mathcal{T}_h} (\cdot,\cdot)_{\partial K} \tag{2.3}$$

we obtain

$$\begin{aligned}
&(\partial_t \boldsymbol{u}_h, \boldsymbol{v}_h) + (\boldsymbol{u}_* \cdot \nabla \boldsymbol{u}_h, \boldsymbol{v}_h) + \nu(\nabla \boldsymbol{u}_h, \nabla \boldsymbol{v}_h) - (p_h, \nabla \cdot \boldsymbol{v}_h) + (q_h, \nabla \cdot \boldsymbol{u}_h) \\
&\quad + (\partial_t \widetilde{\boldsymbol{u}}, \boldsymbol{v}_h) - (\widetilde{\boldsymbol{u}}, \boldsymbol{u}_* \cdot \nabla \boldsymbol{v}_h + \nu \Delta \boldsymbol{v}_h + \nabla q_h)_h \\
&\quad + (\widetilde{\boldsymbol{u}}, \nu \boldsymbol{n} \cdot \nabla \boldsymbol{v}_h + q_h \boldsymbol{n})_{\partial h} = (\boldsymbol{f}, \boldsymbol{v}_h)
\end{aligned} \tag{2.4}$$

and

$$\begin{aligned}(\partial_t\widetilde{\boldsymbol{u}},\widetilde{\boldsymbol{v}}) &+ (\boldsymbol{u}_*\cdot\nabla\widetilde{\boldsymbol{u}} - \nu\Delta\widetilde{\boldsymbol{u}},\widetilde{\boldsymbol{v}})_h + (\nu\boldsymbol{n}\cdot\nabla\widetilde{\boldsymbol{u}},\widetilde{\boldsymbol{v}})_{\partial h}\\ &+ (\partial_t\boldsymbol{u}_h + \boldsymbol{u}_*\cdot\nabla\boldsymbol{u}_h - \nu\Delta\boldsymbol{u}_h + \nabla p_h,\widetilde{\boldsymbol{v}})_h\\ &+ (\nu\boldsymbol{n}\cdot\nabla\boldsymbol{u}_h - p_h\boldsymbol{n},\widetilde{\boldsymbol{v}})_{\partial h} = (\boldsymbol{f},\widetilde{\boldsymbol{v}}).\end{aligned} \tag{2.5}$$

These discrete variational equations must hold for all test functions $\boldsymbol{v}_h \in V_h$, $q_h \in Q_h$ and $\widetilde{\boldsymbol{v}} \in \widetilde{V}$ and the choice of $\boldsymbol{u}_*$, that is discussed below. Apart from taking the pressure subscale to be zero, no approximations have been made to arrive to equations (2.4) and (2.5) and therefore these equations cannot be solved. Because the space $\widetilde{V}$ is still infinite dimensional and equations (2.4) and (2.5) are coupled, a modeling step is required to close the system. A common approximation made in most VMS methods is

$$(\boldsymbol{u}_*\cdot\nabla\widetilde{\boldsymbol{u}},\widetilde{\boldsymbol{v}}) + \nu(\nabla\widetilde{\boldsymbol{u}},\nabla\widetilde{\boldsymbol{v}}) \approx (\tau^{-1}\widetilde{\boldsymbol{u}},\widetilde{\boldsymbol{v}}), \tag{2.6}$$

where τ is a piecewise constant function, computed within each element $K \in \mathcal{T}_h$ as

$$\tau_K^{-1} = \frac{c_1\nu}{h_K^2} + \frac{c_2\|\boldsymbol{u}_h + \widetilde{\boldsymbol{u}}\|_K}{h_K}. \tag{2.7}$$

Here, h_K is a characteristic length of K, c_1 and c_2 are algorithmic constants that depend only on the degree of the finite element approximation being used, and $\|\cdot\|_K$ is some norm defined on each element, e.g. the $L^2(K)$-norm. Equation (2.7) can be motivated also by a heuristic Fourier analysis argument [35], although the important point is its asymptotic behavior in terms of h_K, ν and $\|\boldsymbol{u}_h + \widetilde{\boldsymbol{u}}\|_K$. Another usual approximation is to neglect boundary terms in equation (2.5), which actually vanish for the exact solution. Sometimes, the stronger assumption $\widetilde{\boldsymbol{u}} = 0$ on ∂K is made, which can also be exploited to develop two level approximations, e.g. the residual free bubbles. The crucial point is that equation (2.5) is split into elementwise equations that can be solved locally, i.e. without a global assembly. After this approximation, the fine scale equation to solve at each element becomes

$$\partial_t\widetilde{\boldsymbol{u}} + \tau_K^{-1}\widetilde{\boldsymbol{u}} = \mathcal{P}(\mathbf{R}) \tag{2.8}$$

where

$$\mathbf{R} = \boldsymbol{f} - (\partial_t\boldsymbol{u}_h + \boldsymbol{u}_*\cdot\nabla\boldsymbol{u}_h + \nabla p_h - \nu\Delta\boldsymbol{u}_h) \tag{2.9}$$

and $\mathcal{P}$ is the projection onto the space of subscales discussed below. After this approximation, the system can be solved with appropriate time integration schemes.

Some further modeling choices lead to different VMS models.

Static/Dynamic subscales From the VMS decomposition it follows that $\partial_t \boldsymbol{u} = \partial_t \boldsymbol{u}_h + \partial_t \widetilde{\boldsymbol{u}}$. Considering *dynamic subscales*, introduced in [35, 37], has some advantages such as a correct behavior of time integration schemes and better accuracy. In particular, stability and convergence for the Stokes problem can be proved without any restriction on the time step size and the stabilization parameters on which the formulation depends. The typical approach, however, is the use of *quasistatic subscales* to neglect $\partial_t \widetilde{\boldsymbol{u}}$.

Linear/Nonlinear subscales Applying the VMS decomposition to the nonlinear convective term, four different contributions are obtained on each equation, that is, $\boldsymbol{u} \cdot \nabla \boldsymbol{u} = (\boldsymbol{u}_h + \widetilde{\boldsymbol{u}}) \cdot \nabla(\boldsymbol{u}_h + \widetilde{\boldsymbol{u}})$. After the approximation in equation (2.6) it is possible to keep all the contributions, as proposed in [35, 37]. A simpler alternative is to perform the approximation $\boldsymbol{u} \cdot \nabla \boldsymbol{u} \approx \boldsymbol{u}_h \cdot \nabla(\boldsymbol{u}_h + \widetilde{\boldsymbol{u}})$, which is enough to have numerical stability.

The space of subscales The choice of a space for the approximation of the subscales defines a projector $\mathcal{P}$ to be used in the fine scale equation. One option is to choose $\widetilde{V}$ as the space of the residual, that is to simply take $\mathcal{P} = \mathbb{I}$ (the identity). We refer to this space of subscales as the *algebraic subscales*. Another possibility is to consider the space of the subscales orthogonal to the FE space, that is, to take $\mathcal{P} := \Pi_h^{\perp} = \mathbf{I} - \Pi_h$, where Π_h is the projection onto the FE space [35].

A complete assessment of these modeling choices can be found in [39]. In this thesis work we use static, linear, orthogonal subscales. Using nonlinear and/or dynamic subscales requires tracking them along the iterative and time integration loops, with the consequent increase in memory demands and computational cost (the simplest option is to store the subscales at the integration points). Although using dynamic, nonlinear orthogonal subscales provides better accuracy, these subscales also imply a higher computational cost. The evaluation of this problem-dependent trade-off is outside the scope of this thesis.

However, even if it is simpler to consider algebraic subscales, orthogonal subgrid scales enjoy a number of important properties that are worth having, such as stability without restrictions on the time step size [37], a clear scale separation in the energy transfers and the possibility of predicting backscatter with a stable numerical method [113] and convergence towards weak solutions [17]. After the introduction of orthogonal subgrid scales, different projection-based methods appeared [36, section 4.1]; their difference is the number of terms involving fine scales which are kept in the resolved scale equation. The method we use in this work retains only the advective and pressure terms and projects them separately, and it is usually referred to as term-by-term orthogonal subgrid scales.

With all these modeling choices the final semi-discrete problem to be solved consists of finding $\boldsymbol{u}_h \in V_h$ and $p_h \in Q_h$ such that

$$\begin{aligned}(\partial_t \boldsymbol{u}_h, \boldsymbol{v}_h) + (\boldsymbol{u}_h \cdot \nabla \boldsymbol{u}_h, \boldsymbol{v}_h) + \nu(\nabla \boldsymbol{u}_h, \nabla \boldsymbol{v}_h) - (p_h, \nabla \cdot \boldsymbol{v}_h) + (q_h, \nabla \cdot \boldsymbol{u}_h) \\ + (\boldsymbol{u}_h \cdot \nabla \boldsymbol{v}_h, \tau \mathcal{P}(\boldsymbol{u}_h \cdot \nabla \boldsymbol{u}_h))_h + (\nabla q_h, \tau \mathcal{P}(\nabla p_h))_h = (\boldsymbol{f}, \boldsymbol{v}_h),\end{aligned} \tag{2.10}$$

for any $\boldsymbol{v}_h \in V_h$ and $q_h \in Q_h$.

2.3 Final discrete problem

Equation (2.10) is integrated in time using a second-order backward differentiation formula scheme. The projections onto the FE space are handled explicitly, that is, given $\boldsymbol{u}_h^n$ we first obtain the projection of the convective term η_h (which satisfies $(q_h, \eta_h) = (q_h, \boldsymbol{u}_h^n \cdot \nabla \boldsymbol{u}_h^n)$ for any $q_h \in L^2$) and the projection of the pressure ξ_h (which satisfies $(q_h, \xi_h) = (\nabla q_h, \nabla p_h^n)$ for any $q_h \in L^2$). We use linear finite elements for the velocity, the pressure, and the projections.

The final system is solved iteratively by a predictor-corrector scheme with a block preconditioner which permits us to separate the computation of the velocity and the pressure variable. It is a variant of the classical fractional step method [31, 32] obtained algebraically [53]. This algebraic view opens the door to other options, like performing an iterative correction, eventually converging to the monolithic solution.

2.4 Wind modeling

In this thesis work, we are interested in solving problems of wind flows around high-rise buildings. Such buildings reside entirely within the atmospheric boundary layer; a layer of Earth's atmosphere, extending vertically from its surface, which is characterized by constant shear stress in the vertical direction [77]. This region is generally recognized to be *neutrally stable* at high wind speeds. That is, the buoyancy forces due to temperature gradients are negligible in comparison to surface-driven friction forces.

As is typical in the wind engineering community, we model the natural wind effects in the atmospheric boundary layer by decomposing the incoming velocity field, $\boldsymbol{u} = \overline{\boldsymbol{u}} + \boldsymbol{u}'$, into its stationary *mean profile* $\overline{\boldsymbol{u}}$ and its unsteady *turbulent fluctuations* $\boldsymbol{u}'$. The mean wind profile is a contribution to the velocity field which only changes gradually over the span of several hours or days. It is generally considered constant with respect to the scale of most numerical simulations. On the other hand, the turbulent fluctuations introduce short-term wind gusts with a time span of seconds or minutes. This term induces temporary states of maximum overall wind loading and can induce resonant effects in large structures if their structural eigenfrequencies coincide with frequencies in the gust-induced wind load pattern.

Depending on the local terrain, the variety of obstacles affecting ground friction can vastly change. For instance, consider that different friction forces will arise from flow across grasses, forests, open water, or urban canopies. In our studies, we consider cases in which the local terrain type can be characterized by a fixed or a varying *roughness height* parameter $z_0 > 0$. Intervals of validity for this parameter, for various terrain categories, can be found in numerous engineering code books; e.g., [73]. In addition, one must specify the *friction velocity* u_*, which can be derived from the shear stress on the ground τ_0 by the simple formula $\tau_0 = \rho u_*^2$.

Let D denote a section of the atmospheric boundary layer lying above a flat section of Earth's surface, parameterized by the Cartesian coordinates (x, y, z). Let $\boldsymbol{e} \in \mathbb{R}^3$ be a unit normal vector which denotes the mean wind direction, $\overline{\boldsymbol{u}} \parallel \boldsymbol{e}$. Under the assumptions of neutral stability, homogeneous roughness, the mean velocity $\overline{\boldsymbol{u}}$ is function of the vertical coordinate ($\overline{\boldsymbol{u}} = f(z)$, with $f(\cdot)$ being a generic function useful to highlight dependencies) and is often modeled by the following logarithmic profile [125] when $z > z_0$:

$$\overline{\boldsymbol{u}} = \frac{u_*}{\kappa} \ln\left(\frac{z}{z_0}\right)\boldsymbol{e}, \tag{2.11}$$

where $\kappa \approx 0.41$ is the von Karmán constant.

In this thesis work, we assume that $\boldsymbol{e} = (1, 0, 0)$, that corresponds to an equal probability that the incoming wind will arrive from a fixed horizontal direction. In some environments, this is a poor assumption because geographic features may make the incoming wind arriving from any horizontal direction. In addition, we assume that the three mean profile parameters u_*, z_0, and $\boldsymbol{e}$ are independent. In specific application scenarios, this may also be a poor modeling assumption.

Modeling turbulent fluctuations with physical wind gust statistics is significantly more challenging than modeling the mean profile. Numerous techniques have been proposed in the engineering literature to tackle this issue and we refer the interested reader to the review article [123] for an overview. In this study, we choose to model the velocity fluctuations using the stochastic atmospheric boundary layer turbulence model proposed in [91, 92, 118]; hereafter referred to as the *Mann model*.

Chapter 3

Probability and statistics foundations

In this chapter, we start by introducing probability, which allows treating uncertainties as random variables. The reference we rely upon is [117]. Therefore, a focus on discrete, continuous, and joint random variables is provided. We continue introducing the most important statistics that we consider in this thesis. Then, the problem of forwarding propagation of uncertainties is discussed, together with associated risk measures computation. Finally, we focus on accurate and computationally efficient estimation of statistics, which is fundamental for the target algorithms we aim at building.

3.1 Probability framework

In this work, we assume that it is possible to treat uncertain parameters as random variables. A random variable is characterized by its *probability space* $(\Omega, \mathcal{A}, \mathbb{P})$, where $\Omega \subseteq \mathbb{R}^n$. We assume the existence of such a probability space for all random quantities without loss of generality.

A complete probability space is characterized by the triplet $(\Omega, \mathcal{A}, \mathbb{P})$, where

- Ω is the *sample space*,
- $\mathcal{A}$ is the *σ-algebra*,
- $\mathbb{P}$ is the *probability measure*.

The sample space Ω contains all the possible values the uncertain parameter can assume. The σ-algebra collects the so-called events, which are subsets of Ω. The probability measure is a function $\mathbb{P} : \mathcal{A} \to [0,1]$ returning the probability of each event of the σ-algebra.

We define now the σ-algebra and the probability measure. $\mathcal{A}$ is a σ-algebra if it satisfies the following properties:

- $\mathcal{A} \neq \varnothing$, with $\varnothing$ being the empty set,
- given an event $X \in \mathcal{A}$, then its complement $\Omega \setminus X \in \mathcal{A}$,
- given two events $X \in \mathcal{A}$ and $Y \in \mathcal{A}$, their union $X \cup Y \in \mathcal{A}$.

A probability measure is a function $\mathbb{P} : \mathcal{A} \to [0,1]$ that satisfies the following properties:

- $\forall \mathrm{A} \in \mathcal{A},\ 0 \leq \mathbb{P}[A] \leq 1$,
- $\mathbb{P}[\Omega] = 1$ and $\mathbb{P}[\varnothing] = 0$,
- for all families of mutually disjoint events $A_1, A_2, \ldots, A_N$, i.e. events that cannot occur at the same time, it holds $\mathbb{P}\left[\cup_{n=1}^{N} A_n\right] = \sum_{n=1}^{N} \mathbb{P}[A_n]$.

Other properties follow from the axioms defined above. One of these is the monotonicity property, which reads: given two events $X \in \mathcal{A}$ and $Y \in \mathcal{A}$, $A \subseteq B$, $\mathbb{P}[A] \leq \mathbb{P}[B]$.

Even though random variables can be identified by their sample space, σ-algebra and probability measure, typically other quantities are used when working with random variables, as for example the cumulative distribution function (CDF).

3.2 Discrete random variables

Before discussing continuous random variables, which are the unknown parameters we deal with in this work, we prefer to first introduce *discrete random variables*. This random variable can assume a finite or countable infinite number of possible values.

We introduce now the concepts of CDF and of probability mass function (PMF). The CDF of a discrete random variable X is a function $F_X : \mathbb{R} \to [0,1]$, defined as

$$F_X(x) = \mathbb{P}[X \leq x],\ x \in \mathbb{R}. \tag{3.1}$$

The PMF of a discrete random variable X is a function $p_X : \mathbb{R} \to [0,1]$, defined as

$$p_X(x_i) = \mathbb{P}[X = x_i],\ x \in \mathbb{R}. \tag{3.2}$$

$p_X(x_i)$ is the probability of having $X = x_i$. The sum of the probabilities associated to all possible values x_i is 1.

Let $X = \{x_1, x_2, \dots, x_N\}$ be a discrete random variable, the expected value of X is defined as

$$\mathbb{E}[X] = \sum_{n=1}^{N} x_n \, \mathbb{P}[X = x_n]. \tag{3.3}$$

Another statistical value we want to introduce is the P^{th} central moment, which is defined as $\mathbb{E}[(X - \mathbb{E}[X])^P]$. The second central moment ($P = 2$) is the variance, which is an indicator of the dispersion of events around the expected value. We denote the variance as

$$\mathbb{V}[X] = \sigma^2[X] = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2 , \tag{3.4}$$

where $\sigma[X]$ is the standard deviation of X.

3.3 Continuous random variables

We introduce now *continuous random variables*, which can assume an infinite number of values. A random variable X is continuous if it exists a function $\varrho : \mathbb{R} \to [0, \infty)$ such that

$$\mathbb{P}[X \in B] = \int_B \varrho(x) \, \mathrm{d}x, \tag{3.5}$$

for any measurable set B of real numbers. ϱ is the probability density function (PDF) of the random variable.

The CDF of a continuous random variable X is a function $F_X : \mathbb{R} \to [0, 1]$ and reads

$$F_X(x) = \int_{-\infty}^{x} \varrho(x) \, \mathrm{d}x. \tag{3.6}$$

The expected value of a continuous random variable X is

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} x\varrho(x) \, \mathrm{d}x. \tag{3.7}$$

Central moments of order P of continuous random variables are defined as for discrete random variables. The definition of variance for a continuous random variable X follows from the definition of the expected value, and reads

$$\begin{aligned}\mathbb{V}[X] = \sigma^2[X] = \mathbb{E}[(X-\mathbb{E}[X])^2] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2 \\ = \int_{-\infty}^{\infty}(x-\mathbb{E}[X])^2\varrho(x)\,\mathrm{d}x = \int_{-\infty}^{\infty}x^2\varrho(x)\,\mathrm{d}x - \left(\int_{-\infty}^{\infty}x\varrho(x)\,\mathrm{d}x\right)^2.\end{aligned} \tag{3.8}$$

It is worth remarking once and for all that in this thesis we work with continuous random variables. For this reason, unless otherwise specified, the random variables are considered by default to be continuous.

3.4 Jointly distributed random variables

In our applications we normally deal with more than one random variable at once. For this reason, it is important to introduce jointly distributed random variables, or random vectors. Hereafter we consider only two random variables; however, it is easy to generalize for more random variables.

Let X and Y be two random variables. We first define their joint CDF, that reads

$$F_{X,Y}(a,b) = \mathbb{P}[X \leq a, Y \leq b], \; a,b \in \mathbb{R}. \tag{3.9}$$

Then, let X and Y be continuous random variables and A and B be two sets of real numbers. X and Y are jointly continuous if

$$\mathbb{P}[X \in A, Y \in B] = \int_B\int_A \varrho_{X,Y}(x,y)\,\mathrm{d}x\,\mathrm{d}y \tag{3.10}$$

holds, where the non-negative function $\varrho_{X,Y}(x,y)$ is the joint PDF of X and Y.

Finally, we introduce the concept of *independent* random variables, which are variables that do not affect each other. More formally, X and Y are independent if for any two sets A and B of real numbers, the following holds

$$\mathbb{P}[X \in A, Y \in B] = \mathbb{P}[X \in A]\,\mathbb{P}[Y \in B]. \tag{3.11}$$

It follows that

$$F_{X,Y}(a,b) = F_X(a)F_Y(b), \; a,b \in \mathbb{R}, \tag{3.12}$$

and, if X and Y are jointly continuous,

$$\varrho_{X,Y}(x,y) = \varrho_X(x)\varrho_Y(y), \tag{3.13}$$

where $\varrho_X(x)$ and $\varrho_Y(y)$ are the PDF of X and Y, respectively.

3.5 Properties of principal statistical values

We present now the main properties of the principal statistical values we deal with in this work, that are the expected value, the variance and the covariance.
Let X, Y be two random variables defined on $(\Omega, \mathcal{A}, \mathbb{P})$. If $|\mathbb{E}[X]| < \infty$, it follows that

- if $Y < X$ then $\mathbb{E}[Y] < \mathbb{E}[X]$,
- $|\mathbb{E}[X]| \leq \mathbb{E}[|X|]$,
- $\mathbb{E}[aX + bY] = a\,\mathbb{E}[X] + b\,\mathbb{E}[Y]$.

The covariance of two random variables X and Y is defined as

$$\mathrm{co}\mathbb{V}[X,Y] = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]. \tag{3.14}$$

Some of the most important properties of the variance and of the covariance operators are

- $\mathbb{V}[X] > 0$,
- $\mathbb{V}[aX] = a^2\,\mathbb{V}[X]$,
- $\mathbb{V}[aX + bY] = a^2\,\mathbb{V}[X] + b^2\,\mathbb{V}[Y] + 2ab\,\mathrm{co}\mathbb{V}[X,Y]$,
- $\mathrm{co}\mathbb{V}[X,X] = \mathbb{V}[X]$.

More generally, for N random variables $X_1, \ldots, X_N$,

$$\mathbb{V}[\sum_{n=1}^{N} X_n] = \sum_{n=1}^{N} \mathbb{V}[X_n] + \sum_{\substack{i,j=1 \\ i\neq j}}^{N} \mathrm{co}\mathbb{V}[X_i, Y_j]. \tag{3.15}$$

It is possible to estimate the correlation between two random variables X and Y by computing the cross-correlation coefficient, which is defined as

$$\rho(X,Y) = \frac{\mathrm{co}\mathbb{V}[X,Y]}{\sqrt{\mathbb{V}[X]\,\mathbb{V}[Y]}}. \tag{3.16}$$

The cross-correlation coefficient takes values in $[-1,1]$. The random variables are perfectly uncorrelated if $\rho(X,Y)=0$, while the variables are perfectly correlated if $\rho(X,Y)=1$, and perfectly uncorrelated if $\rho(X,Y)=-1$.

3.6 Propagation of uncertainties

UQ is the field of science which studies how uncertainties propagate within a system where some parameters are not stochastic. This implies that all output observables of such a system are random variables. We aim at estimating statistics of these quantities.
Let us denote the unknown input values of the system with the generic term w, which may be a scalar or a vector. The solution of the system u is therefore function this unknown input

$$u(w) = \mathcal{S}(w), \tag{3.17}$$

where $\mathcal{S}$ represents the system (or solver) and only the dependency on w is made explicit. Of course, other dependencies as space $\boldsymbol{x}$ or time t can be present. We are interested in computing quantities which depend on the solution of the system, that we denote as $\mathrm{Q}(w) \equiv \mathrm{Q}(u(w))$. A quantity like this is called quantity of interest.
An example of uncertainty propagation is shown in figure 3.1, where the uncertainty w is propagated to the quantity $\mathrm{Q}(w)$ in the system $\mathcal{S}$.

3.7 Decision models

One of the main purposes of UQ is to provide the tools for making decisions on top of statistical values of the QoI. Many different decision models (or risk measures) are available in the literature, and we refer for example to [114, 115] and references therein for details. A decision model is a functional $\mathcal{R} : X \to \mathcal{R}(X)$ that assigns a number $\mathcal{R}(X)$ to a random variable X, to quantify the random variable associated risk. Some of the decision models presented in [115] are reported next.

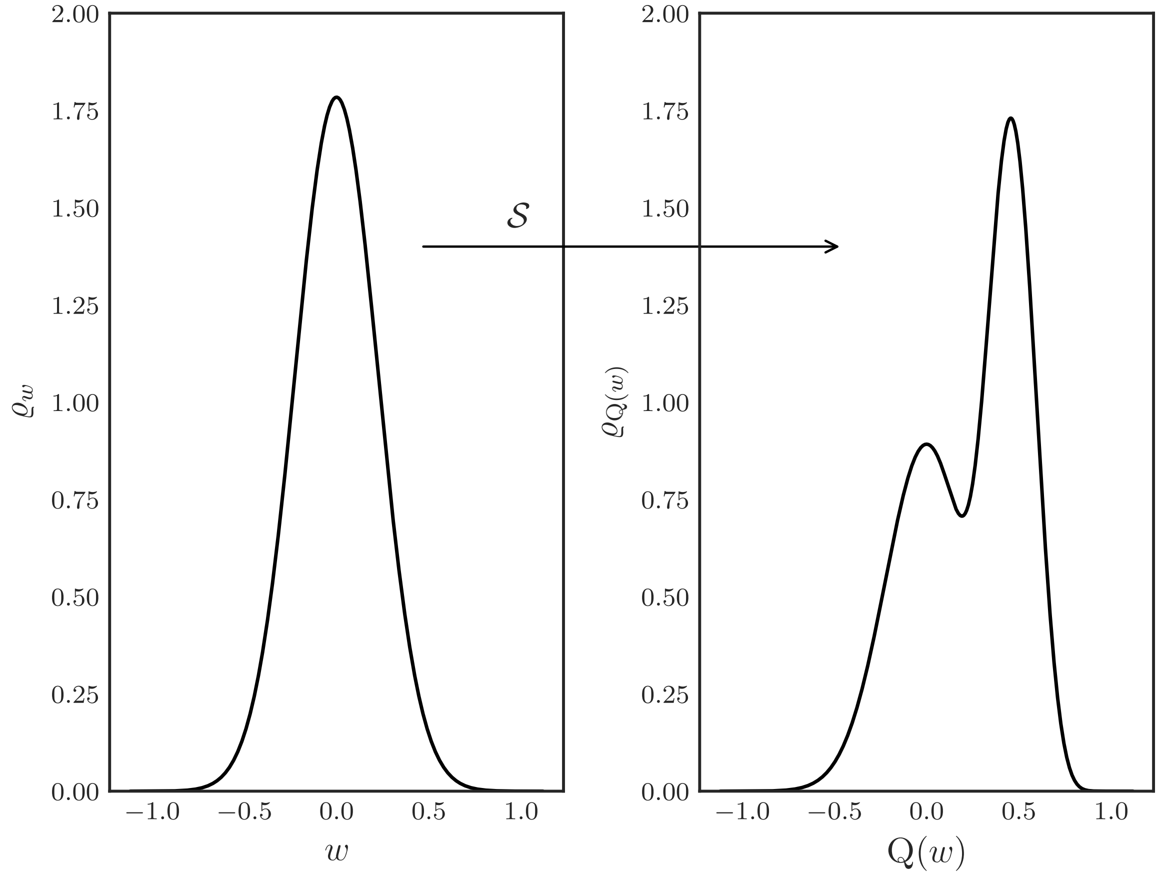


Figure 3.1: Propagation of uncertainties between the random input parameter w and the output quantity $\mathrm{Q}(w)$ via the system $\mathcal{S}$. The probability density functions are plotted.

Expected value The expected value risk measure is $\mathcal{R}(X) = \mathbb{E}[X]$. It is not sensitive to high values, nor to oscillations, since they are damped in the computation of the expected value.

Worst case scenario The decision model $\mathcal{R}(X) = \sup X$ only considers the highest value of X, ignoring the PDF distribution. Such a criterion may be too restrictive, since for distributions as the normal or the exponential $\sup X = \infty$ [115].

Safety margin This risk measure adds safety margins to the expected value, and reads $\mathcal{R}(X) = \mathbb{E}[X] \pm c\sigma[X]$, where $c > 0$ is a constant and $\sigma[X]$ the standard deviation of the random variable. This risk measure is symmetric, since both $\mathbb{E}[X]$ and $\sigma[X]$ are symmetric operators. For this reason, $\mathcal{R}(X)$ may be insensitive to high values in the tail [115].

Failure probability The decision model evaluates $\mathbb{P}[X > \bar{X}]$, that is the probability that X is larger than a critical value $\bar{X}$. This risk measure may present the issue that two random variables can have the same probability, but different distributions, in particular in the upper tail of the PDF [115].

Value at Risk The Value at Risk (VaR) risk measure is $\mathcal{R}(X) = \text{VaR}_\alpha[X]$, where $\alpha \in (0,1)$ is a probability and $\text{VaR}_\alpha[X]$ is the smallest x such that $F_X^{-1}(x)$ is not smaller than α. When the CDF $F_X(x)$ is strictly increasing, it is true that $\text{VaR}_\alpha[X] = F_X^{-1}(x)$ [115]. The name quantile is widely used as well in literature to refer to the VaR value.

Conditional Value at Risk The Conditional Value at Risk (CVaR) risk measure $\mathcal{R}(X) = \text{CVaR}_\alpha[X]$ is particularly interesting when the PDF is not symmetric, since it measures the weight of the tail of the probability density function. Given $\alpha \in (0,1)$, the CVaR of X at probability α is defined as

$$\text{CVaR}_\alpha[X] = \frac{1}{1-\alpha}\int_\alpha^1 \text{VaR}_\beta[X] d\beta. \tag{3.18}$$

When the CDF of X has no discontinuities, the CVaR can be computed as the conditional expectation [114]

$$\text{CVaR}_\alpha[X] = \mathbb{E}[X|X > \text{VaR}_\alpha[X]]. \tag{3.19}$$

In other words, CVaR_α splits the area beneath the PDF curve between $[\text{VaR}_\alpha[X], +\infty]$ into two balancing parts. An alternative name which can be found in the literature for the CVaR is superquantile.

To ease the understanding of the multiple risk measures presented, we consider a standard Gaussian (or normal) distribution with mean 0 and variance 1, which reads $X \sim \mathcal{N}(\mu, \sigma^2) = \mathcal{N}(0,1)$. The PDF of such a random variable is $\frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}}$, and its CDF is $\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{+\infty} e^{-\frac{t^2}{2}}\,\mathrm{d}t$. Let $c = 1$, $\bar{X} = 1.0$ and $\alpha = 0.86$. Results for all considered risk measures are reported in table 3.1. We present in figures 3.2 and 3.3 the CDF and the PDF of the standard normal random variable X. Such plots help understanding

	$\mathbb{E}[X]$	$\sup X$	$\mathbb{E}[X] \pm c\sigma[X]$	$\mathbb{P}[X > \bar{X}]$	$\text{VaR}_\alpha[X]$	$\text{CVaR}_\alpha[X]$
$\mathcal{R}(X)$	0.0	$+\infty$	0 ± 1	0.16	1.0	1.53

Table 3.1: Risk measures for a standard normal random variable.

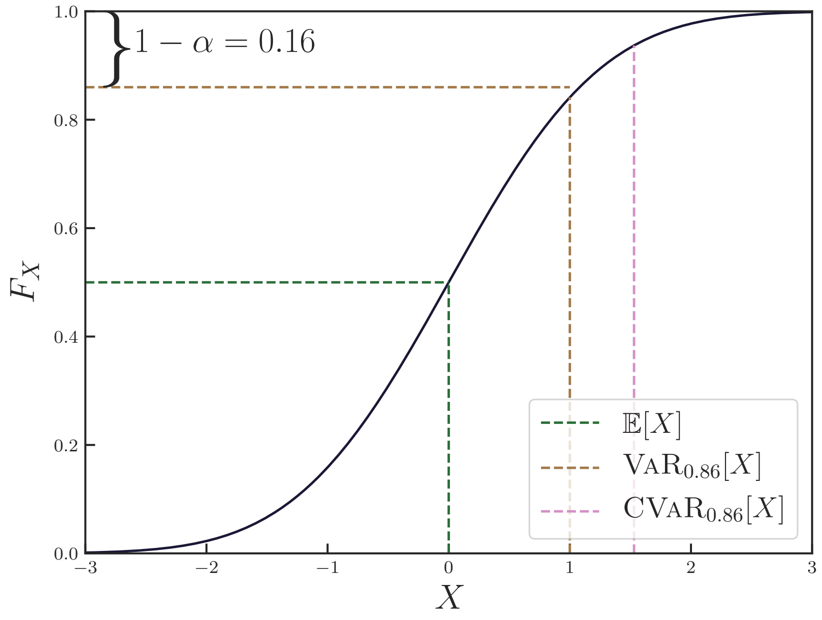


Figure 3.2: Expected value, VaR and CVaR, with $\alpha = 0.16$, of a standard normal random variable. Statistics are computed from the CDF of the random variable.

the meaning of the some statistical values, and consequently of related risk measures. For example, by looking at figure 3.3 it is clear that the probability that $X > 1$ is 0.16. Moreover, we can observe that the CVaR is a conditional expectation, since $\text{CVaR}_\alpha[X]$ splits the area between $[\text{VaR}_\alpha[X], +\infty]$ and the PDF into equal parts.

3.8 Estimation of moments

The exact statistics of a random variable are known only when the PDF or the CDF are known. This is, of course, not always possible, especially for complex problems, whose output quantities are the output of some system function, as shown in figure 3.1. In this section we focus on the estimation, i.e. approximation, of raw and central moments. Raw moments are defined as

$$\mathbb{E}[X^P] \tag{3.20}$$

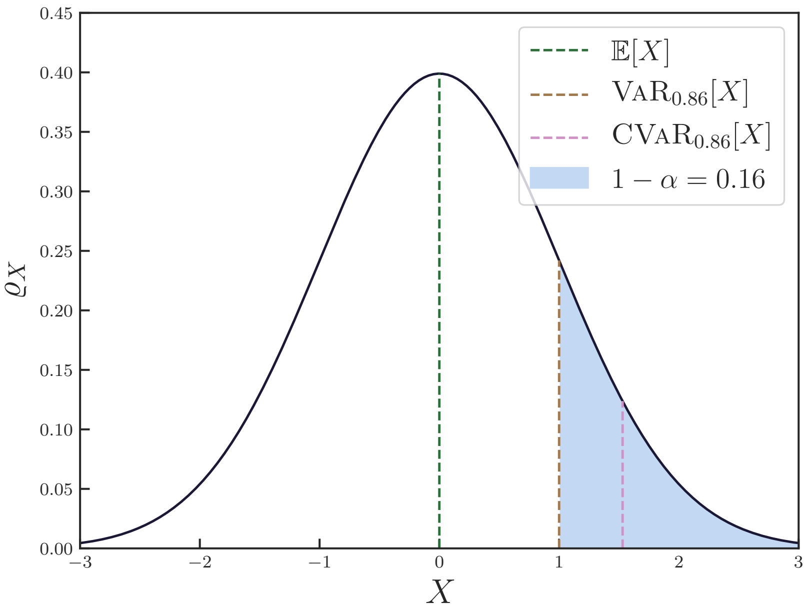


Figure 3.3: Expected value, VaR and CVaR, with $\alpha = 0.16$, of a standard normal random variable. Statistics are computed from the PDF of the random variable.

and central moments as

$$\mathbb{E}[(X - \mathbb{E}[X])^P], \tag{3.21}$$

where X is a random variable and P is the order. We remark that the vast majority of traditional applications of descriptive statistics usually require the estimation of expected value, variance, skewness and kurtosis, and not of higher-order moments, as commented in [20, 108].

Let N be the number of independent and identically distributed realizations of a random variable X. The simplest way to approximate the expected value and the variance of X is by computing the sample average (or sample mean), which reads

$$\mathbb{E}[X] \approx \mathrm{E}^N[X] = \frac{1}{N}\sum_{n=1}^{N} X_n, \tag{3.22}$$

and the sample variance, defined as

$$\mathbb{V}[X] \approx \mathrm{V}^N[X] = \frac{1}{N-1}\sum_{n=1}^{N}(X_n - \mathrm{E}^N[X])^2. \tag{3.23}$$

It follows that the covariance between two random variables X and Y can be approximated by the sample covariance, which reads

$$\mathrm{co}\mathbb{V}[X,Y] \approx \mathrm{coV}^N[X,Y] = \frac{1}{N-1}\sum_{n=1}^{N}(X_n - \mathrm{E}^N[X])(Y_n - \mathrm{E}^N[Y]). \tag{3.24}$$

Sample formulas to estimate higher-order central moments are not reported here for the sake of simplicity.

It is straightforward to understand the bottleneck of these estimations for large-scale computations. First, the sample mean is computed only once all the N realizations are available. Then, sample central moments can be estimated. This approach is highly inefficient and can easily become impractical in a large-scale distributed environment, since the cost of accessing each data $n \in [1, N]$ twice massively dominates the computation cost [20]. Additionally, such an approach also requires storing in memory and/or saving to file the system results, and this can easily become a bottleneck for complex problems. For this reason, updating statistical estimators on the fly and accessing memory only once is the preferred strategies, since it reduces the overall time to solution, increasing the computational efficiency of the algorithm.

3.8.1 Estimation by updating difference from the current mean

The first formula to estimate the variance on the fly by passing a new realization and accessing memory once is introduced in [137]. Let $\mathrm{E}^{N-1}[X]$ be the sample average at the previous step, $\mathcal{M}_2^{N-1}$ the sum of squares of differences at the previous step and X_N the new realization. We first update the sample average with the new realization

$$\mathrm{E}^N[X] = \mathrm{E}^{N-1}[X] + \frac{X_N - \mathrm{E}^{N-1}[X]}{N} \tag{3.25}$$

and then update the sum of squares of differences from the current mean

$$\mathcal{M}_2^N = \mathcal{M}_2^{N-1} + (X_N - \mathrm{E}^{N-1}[X])(X_N - \mathrm{E}^N[X]), \tag{3.26}$$

where $\mathcal{M}$ presents order 2 and set N. The sample variance can then be computed as $\mathrm{V}^N[X] = \frac{\mathcal{M}_2^N}{N-1}$.

A generalization is proposed in [29, 30], where the authors extend above equations to update the variance estimate with any number of new realizations. Specifically, let N

be the number of samples at the previous step and let M be the new realizations. We first update the sample average

$$\mathrm{E}^{N+M}[X] = \mathrm{E}^{N}[X] + M\frac{\mathrm{E}^{M}[X] - \mathrm{E}^{N}[X]}{N+M} \tag{3.27}$$

and then the sum of squares of differences from the current mean

$$\mathcal{M}_2^{N+M} = \mathcal{M}_2^{N} + \mathcal{M}_2^{M} + NM\frac{(\mathrm{E}^{M}[X] - \mathrm{E}^{N}[X])^2}{N+M}. \tag{3.28}$$

The extension to any arbitrary order $P > 1$ is proposed in [20]. One can compute the difference from the current mean for any power P, namely $\mathcal{M}_P^{N+M} = \sum_{i=1}^{N+M}(X_i - \mathrm{E}^{N+M}[X])^P$. The formula to update it is

$$\begin{aligned}\mathcal{M}_P^{N+M} &= \mathcal{M}_P^{N} + \mathcal{M}_P^{M}\\ &+ \sum_{p=1}^{P-2}\binom{p}{P}\left[\left(-\frac{M}{N+M}\right)^p \mathcal{M}_{P-p}^{N} + \left(\frac{N}{N+M}\right)^p \mathcal{M}_{P-p}^{M}\right](\mathrm{E}^{M}[X] - \mathrm{E}^{N}[X])^p\\ &+ \left(\frac{NM}{N+M}(\mathrm{E}^{M}[X] - \mathrm{E}^{N}[X])\right)^P\left[\frac{1}{M^{P-1}} - \left(\frac{-1}{N}\right)^{P-1}\right],\end{aligned} \tag{3.29}$$

where $\binom{p}{P}$ is the binomial coefficient.

Moreover, in [20] the authors propose a formula to update on the fly co-moments (which involve more than one random variable), as the covariance. Such formulas access memory once, thus being suitable for large-scale scenarios. Let X and Y be two random variables, and let N and M be two subsets of realizations. We report here the formula to estimate the covariance. First, we compute

$$\mathcal{C}_2^{N+M} = \mathcal{C}_2^{N} + \mathcal{C}_2^{M} + NM\frac{(\mathrm{E}^{M}[X] - \mathrm{E}^{N}[X])(\mathrm{E}^{M}[Y] - \mathrm{E}^{N}[Y])}{N+M}, \tag{3.30}$$

and then the unbiased covariance estimate as $\frac{\mathcal{C}_2^{N+M}}{N+M-1}$. We refer to [20, 108] for higher-order co-moments estimation.

We remark that above results for the incremental estimation of moments and co-moments of arbitrary order are extended in [108], where arbitrary weights are introduced. Moreover, the framework proposed in this section 3.8.1 is proven to be "numerically stable and presents near-optimal linear scalability and speed-up properties", as commented by the authors in [20, section 6].

3.8.2 Estimation by updating power sums

Another possibility to estimate central moments is by updating power sums on the fly and then computing the so-called h-statistics, which are unbiased estimators with minimal variance, compared to other unbiased central moment estimators [50, 60, 86]. Using unbiased estimators is particularly convenient when working with hierarchical MC methods, as multilevel Monte Carlo or multifidelity Monte Carlo, since telescopic sums of estimators can be easily written by using h-statistics [110, 86]. Moreover, power sums can be exploited to estimate also the sample average, that is the main raw moment we are interested in.

In this work, we follow the same approach of [110, 86], where sample-based central moment estimators are computed directly from power sums.

Let X be a random variable and N its number of realizations. The power sum of order P of X is defined as

$$S_P^N[X] = \sum_{n=1}^{N} (X_n)^P. \tag{3.31}$$

Power sums allow for the computation of h-statistics, which are estimators of central moments. Therefore, the central moment of order P is estimated by the h-statistic of order P, which is expressed in terms of the power sums and of the number of realizations. In other words

$$h_P^N[X] = f(S_p^N[X], N), \; p \in [1, P], \tag{3.32}$$

where $f(\cdot)$ is a generic function operator, useful for highlighting function dependencies. In the following, we may use the shorthand notation $S_P \equiv S_P^N[X]$ and $h_P \equiv h_P^N[X]$.

It follows that we can approximate the expected value and the variance of X as [110, 86]

$$\begin{aligned} \mathbb{E}[X] \approx \mathrm{E}^N[X] &= \frac{S_1}{N} \\ \mathbb{V}[X] \approx h_2^N[X] &= \frac{NS_2 - S_1^2}{(N-1)N}. \end{aligned} \tag{3.33}$$

For the sake of completeness, we report as well the equations for the estimation of central moments of order 3 and 4 [86]

$$\begin{aligned} h_3 &= \frac{N^2 S_3 - 3NS_2S_1 + 2S_1^3}{(N-2)(N-1)N} \\ h_4 &= \frac{(-4N^2+8N-12)S_3S_1 + (N^3-2N^2+3N)S_4 + 6NS_2S_1^2 + (9-6N)S_2^2 - 3S_1^4}{(N-3)(N-2)(N-1)N}. \end{aligned} \tag{3.34}$$

Above estimation of the central moments through h-statistics present an associated statistical error (SE), that is the variance (or the standard deviation) of the moment estimator, which decreases as the number of realizations N grows. For example, the variance of expected value and variance estimators read [86]

$$\begin{aligned} \mathbb{V}[\mathrm{E}^N[X]] &= \frac{\sigma^2[X]}{N} \\ \mathbb{V}[h_2^N[X]] &= \frac{\mathbb{E}[(X-\mathbb{E}[X])^4]}{N} - \frac{\mathbb{E}[(X-\mathbb{E}[X])^2]^2(N-3)}{(N-1)N}, \end{aligned} \tag{3.35}$$

where we recall that $\mathbb{E}[(X-\mathbb{E}[X])^P]$ denotes the central moment of order P. As commented in [86, section 2.1], "the naive approach of replacing central moments $\mathbb{E}[(X-\mathbb{E}[X])^P]$ with their unbiased estimate $h_P^N[X]$ will not result in an unbiased estimator for $\mathbb{V}[h_P^N[X]]$, since $\mathbb{V}[h_P^N[X]]$ depends non-linearly on the central moments". We therefore follow the approach proposed in [110, 86] to estimate the variances of equation (3.35), that consists of "not only substituting $h_P^N[X]$ for $\mathbb{E}[(X-\mathbb{E}[X])^P]$ but also introducing an additional multiplicative coefficient for each substitution". We therefore obtain the following unbiased estimators [86]

$$\begin{aligned} \mathbb{V}[\mathrm{E}^N[X]] &\approx \frac{h_2}{N} \\ \mathbb{V}[h_2^N[X]] &\approx \frac{N((N-1)^2NS_4 - (N^2-3)S_2^2) + (6-4N)S_1^4}{(N-3)(N-2)(N-1)^2N^2} \\ &+ \frac{4N(2N-3)S_2S_1^2 - 4(N-1)^2NS_3S_1}{(N-3)(N-2)(N-1)^2N^2}. \end{aligned} \tag{3.36}$$

We refer to [86, section 2.1] for details about unbiased variance estimators of higher-order central moments, which are $\mathbb{V}[h_P^N[X]]$, $P > 2$.

We remark that expressing h-statistics in terms of power sums can lead to numerically unstable formulas, for p large. A possible solution could be expressing h-statistics in terms of centered power sums $S_P^N[X-\mathbb{E}[X]] = \sum_{n=1}^N (X_n - \mathbb{E}[X])^P$, somehow similarly to what is discussed in section 3.8.1. However, we do not focus on such a topic in this thesis work.

3.8.3 Estimation of combined moments by updating combined power sums

We focus now on the estimation of *combined* moments, which are moments of time-dependent quantities. The simplest way to compute such combined moments is by storing the whole historical data for all realizations and then computing statistics from the available information. However, this is not computationally viable. For example, it would imply storing the whole time history of the pressure on every point on the skin surface of a building. This becomes inviable when we have hundreds of thousands of points to evaluate and thousands of time steps. We present here a tentative extension of the equations presented in section 3.8.2 for the estimation of combined moments. Our idea consists of updating at each time step the power sums of the random variable, and then computing h-statistics with the available combined power sums.

Let $X(t)$ be a process random variable, N its number of realizations, and M the number of time steps discretizing the time window. The update of power sums is done as

$$S_P^{N,M}[X] = \sum_{n=1}^{N} \sum_{m=1}^{M} (X_{n,m})^P. \tag{3.37}$$

An example of update of combined power sums is shown in figure 3.4, where a generic realization $n \in [1, N]$ of a process is represented. The time window of this process is $[-6, 6]$ and information is collected for 30 time steps, thus $M = 30$. Of course, such an update must be repeated for all N realizations. We remark that, by updating power sums at each time step, all fluctuations and peaks that may appear during the time history of a single realization affect the moment estimation.

Expected value and variance of $X(t)$ are estimated as

$$\begin{aligned} \mathbb{E}[X] &\approx \mathrm{E}^{N,M}[X] = \frac{S_1}{NM} \\ \mathbb{V}[X] &\approx h_2^{N,M}[X] = \frac{NMS_2 - S_1^2}{(NM)^2}, \end{aligned} \tag{3.38}$$

where $S_P \equiv S_P^{N,M}[X]$.

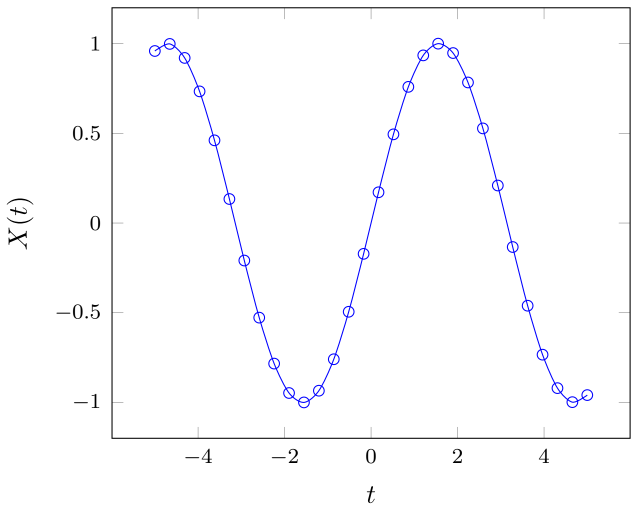


Figure 3.4: Process $X(t)$ plotted as function of time. Each circle represents a sample updating combined power sums.

Although a formal proof of minimal variance for $h_2^{N,M}[X]$ could not be reached, numerical examples have shown its consistency compared to the standard computation of statistics from data stored to file.

A simple validation of the proposed approach is presented in table 3.2, where statistics of observables of section 7.1 are estimated using both the combined power sums approach and the NumPy Python library [61]. We observe that the difference we obtain is minimal, thus giving consistency to the method we propose.

X	$\frac{\left\|\mathrm{E}^{N,M}[X]-\mathrm{E}^{N,M,\text{ref.}}[X]\right\|}{\frac{\mathrm{E}^{N,M}[X]+\mathrm{E}^{N,M,\text{ref.}}[X]}{2}}$	$\frac{\left\|h_2^{N,M}[X]-\mathrm{V}^{N,M,\text{ref.}}[X]\right\|}{\frac{h_2^{N,M}[X]+\mathrm{V}^{N,M,\text{ref.}}[X]}{2}}$
F_d	$6.9 \cdot 10^{-15}$	$5.3 \cdot 10^{-13}$
M_b	$6.9 \cdot 10^{-15}$	$5.3 \cdot 10^{-13}$

Table 3.2: Relative difference of expected value estimation and of variance estimation, computed using the combined power sums approach and the NumPy Python library [61], that is the reference solution in this comparison. $N = 100$ and $M = 10000$ are considered. F_d is the drag force and M_b the base moment, and the data used for the comparison are taken from the problem of section 7.1.

Chapter 4

Hierarchical Monte Carlo methods

In this chapter, we start reviewing the current state of the art of hierarchical MC methods in sections 4.1–4.4. We focus on the standard MC method and on other algorithms that enhance standard MC, as MLMC or CMLMC. These methods solve stochastic problems by propagating uncertainties within the system and allow performing a statistical analysis of an output QoI, which can be both a scalar or a field. Then, the integration of hierarchical MC algorithms with the strategies presented in section 3.8 to compute statistical moments on the fly is discussed in section 4.5. We continue introducing a convergence criterion and analyzing it in the context of both single-level (MC) and multi-level (MLMC, CMLMC) methods in section 4.6. In sections 4.7 and 4.8 we present the algorithms of the main methods discussed in the chapter, together with considerations on their execution on HPC systems. We conclude the chapter validating the implemented framework in section 4.9.

Let us introduce some preliminary notation. We refer to the solution of the problem under consideration with $u = f(w)$, where w is the random variable of the system and f denotes a generic function and is useful to highlight function dependencies. The physical problems considered are solved exploiting the FE solver Kratos. This implies that a discretized domain D_H is exploited, where D is the problem domain and H is a discretization parameter of the domain. An example of a discretization parameter is for example the minimal size.

The content of this chapter is taken from the preprint of our work [128] and is adapted and integrated with additional content wherever needed.

4.1 The Monte Carlo method

The MC method is the reference technique for the stochastic analysis of multi-physics problems with uncertainties in the data parameters. As previously mentioned, it gives origin to a wide class of different algorithms, whose main idea is to repeat many times the realization with uncorrelated random variable w, and to estimate the desired statistics of the QoI a posteriori.

We are interested in estimating the expected value of a generic scalar quantity of interest $\mathrm{Q} = f(u(w))$, that is $\mathbb{E}[\mathrm{Q}]$. This means that the expected value estimator must satisfy some criterion for achieving convergence. However, we remark that other statistical estimators are computed to assess convergence, as we will see next.

The MC potential lies in its basic property of convergence to the exact statistics as the number of samples N tends to infinity, independently of the dimensionality of the stochastic space. It also presents the advantage of considering the solver as a black box, since it is non-intrusive and directly applicable to any simulation code, making it suitable for any kind of problem.

Considering $\mathrm{Q} \approx \mathrm{Q}_H$, where Q_H is the approximation on the discretized domain D_H, the MC expected value estimator of the output quantity of interest is

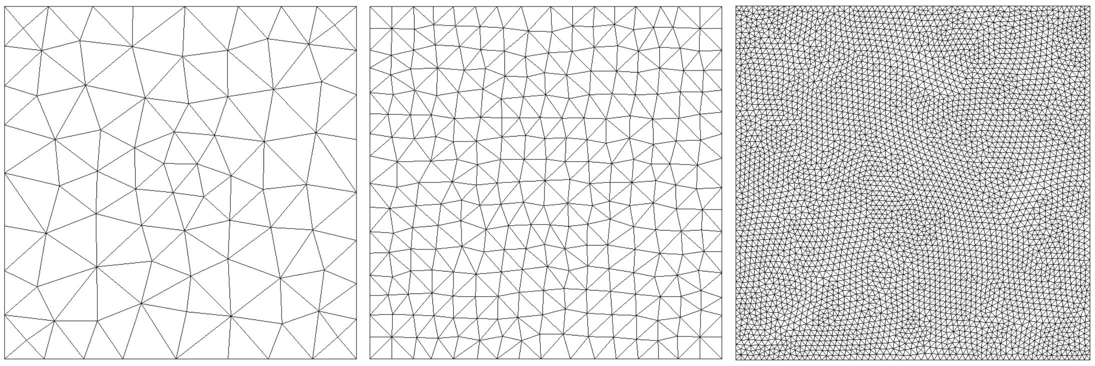
$$\mathbb{E}[\mathrm{Q}] \approx \mathrm{E}^{\mathrm{MC}}[\mathrm{Q}_H] = \frac{1}{N}\sum_{n=1}^{N} \mathrm{Q}_H(w^{(n)}), \tag{4.1}$$

where $\mathrm{Q}_H(w^{(n)})$, $n = 1, \ldots, N$, are N independent, identically distributed values computed on D_H.

The MC estimation accuracy of the expectation can be evaluated through the mean square error, that reads

$$e^2_{\mathrm{MC}} = \mathbb{E}[(\mathrm{E}^{\mathrm{MC}}[\mathrm{Q}_H] - \mathbb{E}[\mathrm{Q}])^2] = (\mathbb{E}[\mathrm{Q}_H - \mathrm{Q}])^2 + \frac{\mathbb{V}[\mathrm{Q}_H]}{N}, \tag{4.2}$$

where $\mathbb{E}[\mathrm{Q}]$ is the true expected value of the QoI and $\mathbb{V}[\mathrm{Q}] = \mathbb{E}[\mathrm{Q}^2] - \mathbb{E}[\mathrm{Q}]^2$ is the variance of the QoI. The term $(\mathbb{E}[\mathrm{Q}_H - \mathrm{Q}])$ is the discretization error (DE), it is independent of the statistics of the QoI and only depends on the accuracy of the domain we are exploiting to solve the problem [111, section 2]. On the other hand, $\frac{\mathbb{V}[\mathrm{Q}_H]}{N}$ is the squared SE, which decreases as long as the number of samples grows, and is an indicator of the variance of the expected value estimator [111, section 2].



(a) Mesh discretization of level 0. (b) Mesh discretization of level 1. (c) Mesh discretization of level 2.

Figure 4.1: Hierarchy of three levels.

The main drawback of the MC method is its computational cost, which makes it very expensive for the stochastic analysis of industrial problems with complex geometries. In fact, e^2_{MC} decreases proportionally to N^{-1}. To try to overcome this limitation, different algorithms have been derived from standard MC, as MLMC and CMLMC.

4.2 The multilevel Monte Carlo method

The MLMC method was first introduced in [62, 64, 58]. MLMC reduces the computational cost of solving stochastic problems by standard MC method through a variance reduction technique.

The main idea of the MLMC algorithm is to draw MC instances simultaneously on a sequence of levels of increasing accuracy, and consequently of increasing computational cost. Many MC samples are run on the coarsest levels, in order to capture the statistical variability, and only a few are run on the finest ones, to reduce the so-called discretization error. This way, the computational effort relies mainly on the coarsest levels, while for standard MC the computational effort is all concentrated on the finest. Therefore, this leads to notable computational savings.

The standard procedure to build the hierarchy of levels is to perform uniform refinement in space. By defining the mesh parameter H equal to the reciprocal of the mesh size, H grows as long as the level increases, i.e. $H_0 < H_1 < \ldots < H_L$, where L is the maximum number of levels the current simulation may reach. An example of hierarchy of three levels with uniform refinement is shown in figure 4.1.

The linearity of the expectation operator allows writing the mean of the QoI as a telescopic sum of expectations. Therefore, the MLMC expected value of the QoI reads

$$\begin{aligned}\mathbb{E}[\mathrm{Q}] \approx \mathrm{E}^{\mathrm{MLMC}}[\mathrm{Q}_H] &= \mathrm{E}^{\mathrm{MC}}[\mathrm{Q}_{H_0}] + \sum_{l=1}^{L} \mathrm{E}^{\mathrm{MC}}[\mathrm{Q}_{H_l} - \mathrm{Q}_{H_{l-1}}] \\ &= \sum_{l=0}^{L} \mathrm{E}^{\mathrm{MC}}[Y_l] = \sum_{l=0}^{L} \frac{1}{N_l} \sum_{n=1}^{N_l} Y_l(w^{(n,l)}),\end{aligned} \tag{4.3}$$

where $Y_l = \mathrm{Q}_{H_l} - \mathrm{Q}_{H_{l-1}}$ and $Y_0 = \mathrm{Q}_{H_0}$. For the sake of simplicity, the dependence on the random variable w is made explicit only in the last expression, from which we observe that the two quantities Q_{H_l} and $\mathrm{Q}_{H_{l-1}}$ are computed using the same random variable realization $w^{(n,l)}$.

Analogously to the MC algorithm, the mean square error of the MLMC expectation estimator is the sum of a discretization error and a statistical error:

$$e^2_{\mathrm{MLMC}} = \mathbb{E}[(\mathrm{E}^{\mathrm{MLMC}}[\mathrm{Q}_H] - \mathbb{E}[\mathrm{Q}])^2] = (\mathbb{E}[\mathrm{Q}_H - \mathrm{Q}])^2 + \sum_{l=0}^{L} \frac{\mathbb{V}[Y_l]}{N_l}, \tag{4.4}$$

where the first term represents the squared DE and the latter the squared SE. We can observe matching equations (4.2) and (4.4) that the only difference in the mean square error evaluation is the SE contribute. In fact, in equation (4.2) the variance of Q_H is computed, while in equation (4.4) we consider the difference Y_l on two consecutive levels, whose variance is consistently smaller with respect to the one of Q_H.

Since the MLMC method works with a hierarchy of levels, it is important to calibrate well the number of realizations per level, to avoid wasting unnecessary resources on the finest levels. For this reason, adaptive strategies can be employed. However, as we comment in section 4.3, a bad hierarchy calibration can easily lead to oversampling, resulting in a wrong resources usage. To avoid such an issue, other algorithms, as the CMLMC method, can be employed.

4.3 Adaptivity

Three important considerations lie at the basis of both MC and MLMC algorithms (see [111] and references therein).

- The cost of computing one realization Q_{H_l} grows exponentially with the level accuracy H_l. This means $\mathrm{C}[\mathrm{Q}_{H_l}] \leq c_\gamma H_l^\gamma$, where C is the computational cost and $c_\gamma, \gamma > 0$.

- The discretization error $\left|\mathbb{E}[\mathrm{Q}_{H_l} - \mathrm{Q}]\right|$ decreases exponentially as H_l grows. In other words, $\left|\mathbb{E}[\mathrm{Q}_{H_l} - \mathrm{Q}]\right| \leq c_\alpha H_l^{-\alpha}$, where $c_\alpha, \alpha > 0$.
-
  - MC: $\mathbb{V}[\mathrm{Q}_H]$ constant with respect to H,
  - MLMC: $\mathbb{V}[Y_l]$ decreases exponentially as the level grows. Therefore, $\mathbb{V}[Y_l] \leq c_\beta H_l^{-\beta}$, where $c_\beta, \beta > 0$ and $\alpha \geq \min\{\beta, \gamma\}$.

If the above hypotheses are satisfied, it is known [58] that MLMC reduces the computational cost required to achieve the same mean square error of MC, or the same failure probability criterion, described next in section 4.6.

The hierarchy, that is the number of levels and the number of realizations per level, of hierarchical MC methods can be determined both non-adaptively[1], thus defined prior to the start of the simulation, or adaptively, computed on top of some statistics. Two examples of non-adaptive hierarchies are:

- keep the number of levels L and of realizations per level $N_l,\ l = 0, \dots, L$ constant at each MC/MLMC iteration,
- increase the number of levels by one and double the number of realizations per level at each MC/MLMC iteration.

On the other hand, the evaluation of the computational cost associated to each level, the discretization error and the variance of the QoI allows estimating the optimal adaptive hierarchy for reaching the desired accuracy of the statistical estimator. We refer for example to [111, sections 2.1 and 2.2] for details about how to compute adaptively hierarchies for MC and MLMC.

A screening phase is needed in order to calibrate adaptively the hierarchy of the simulation. This setup takes place before the execution of the main algorithm, and normally few samples per level are run. Nevertheless, this setup is an expensive and challenging phase, since, if wrongly calibrated, it may lead to oversampling or to bad hierarchy estimations. An alternative approach is proposed in [38] and is presented in section 4.4.

[1]In the preprint of our work [128], we use the expression *deterministic* to refer to non-adaptive hierarchies. However, a reviewer suggested replacing deterministic with *non-adaptive*, and since we believe this change increases the clarity of the text, we keep such a suggestion.

It consists of updating on the fly the adaptive hierarchy of the simulation, exploiting a decreasing sequence of tolerance. However, an initial screening phase is still needed.

4.4 The continuation multilevel Monte Carlo method

In [38], the authors introduce the CMLMC algorithm, whose idea is to update the optimal hierarchy on the fly, to decrease the risk of oversampling. The idea of CMLMC is to run, after the initial screening phase, $\mathcal{I}$ MLMC iterations and to use a decreasing sequence of tolerances $\varepsilon_0 > \varepsilon_1 > \ldots > \varepsilon_{\mathcal{I}}$. This implies updating on the fly the optimal hierarchy, increasing its accuracy and decreasing the risk of oversampling. We refer to [38, 111] for further details.

However, as we will see later in section 4.7 and chapter 5, the main problem of CMLMC and of the other approaches discussed above (MC and MLMC), in the context of supercomputers, is their intrinsically serial nature.

4.5 Computation of moments

Computing statistics of QoIs is a crucial step for having high efficiency frameworks. Different techniques are possible to update central and raw moments on the fly, and the two possibilities analyzed are:

- online update of differences from current mean (see section 3.8.1),
- online update of power sums (see section 3.8.2).

For the sake of simplicity, we define the central moment estimator of order $P > 1$, estimated with N samples, as $\mu_P^N \equiv \mu_P^N[\mathrm{Q}] \approx \mathbb{E}[(\mathrm{Q} - \mathbb{E}[\mathrm{Q}])^P]$.

4.5.1 Online update of central moments

The first possibility is to update the differences from the current mean with one-pass formulas and then to compute the central moments, where one-pass means that we update by adding one value per time. We refer to section 3.8.1 and references therein cited. We report here the dependencies for computing moments with one-pass formulas for an arbitrary order P:

$$\mu_P^N = f(\mu_p^{N-1}, \mathrm{Q}(w^{(i)}), N, P),\ p \in [1, P], \tag{4.5}$$

where $f(\cdot)$ is a generic function useful to highlight dependencies. Update formulas with arbitrary set decomposition exist, and the dependencies for an arbitrary order P read as

$$\mu_P^{N_A+N_B} = f(\mu_p^{N_A}, \mu_p^{N_B}, N_A, N_B, P),\ p \in [1, P], \tag{4.6}$$

where N_A, N_B are the sizes of the two sets.

4.5.2 Online computation of power sums

A second possibility is to update power sums with one-pass strategies, where a power sum of order p is defined as $S_P^N \equiv S_P^N[\mathrm{Q}] = \sum_{i=1}^N \mathrm{Q}(w^{(i)})^P$. We refer to section 3.8.2 and references therein for details. From the power sum definition definition, we can derive the dependencies of updating power sums. Given any arbitrary order P, such dependencies are

$$S_P^N = f(S_P^{N-1}, \mathrm{Q}(w^{(i)}), N, P). \tag{4.7}$$

Power sums allow computing h-statistics, which are unbiased estimators with minimal variance of central moments. The h-statistics, for an arbitrary order P, is just function of the power sums and of the number of samples N, therefore

$$h_P^N[\mathrm{Q}] = f(S_p^N, N),\ p \in [1, P]. \tag{4.8}$$

4.6 Stopping criteria

Convergence is said to be accomplished if the estimator of the expected value ($\mathrm{E}^{\mathrm{MC}}[\mathrm{Q}_H]$ or $\mathrm{E}^{\mathrm{MLMC}}[\mathrm{Q}_H]$) achieves a desired tolerance ε, with respect to the true estimator $\mathbb{E}[\mathrm{Q}]$, with a confidence $(1-\phi)$. In other words, we define a failure probability criterion (often denoted simply as failure probability), and we want it to be met with a certain level of confidence. The failure probability criterion for the MC expected value estimator is defined as

$$\mathbb{P}\left[\left|\mathrm{E}^{\mathrm{MC}}[\mathrm{Q}_H] - \mathbb{E}[\mathrm{Q}]\right| \geq \varepsilon\right] \leq \phi, \quad \phi \ll 1, \tag{4.9}$$

where $\varepsilon > 0$ and $(1-\phi) \in (0,1)$. For obtaining the failure probability criterion of the MLMC expected value estimator, it is sufficient to replace $\mathrm{E}^{\mathrm{MC}}[\mathrm{Q}_H]$ with $\mathrm{E}^{\mathrm{MLMC}}[\mathrm{Q}_H]$.

The total error can be bounded, with confidence $(1-\phi)$, by the sum of the discretization error and the statistical error, where the latter is multiplied by a confidence factor [38, 111]:

$$\begin{aligned}\tau &= |\mathbb{E}[\mathrm{Q}_H] - \mathbb{E}[\mathrm{Q}]| \\ &\le |\mathbb{E}[\mathrm{Q} - \mathrm{Q}_H]| + |\mathbb{E}[\mathrm{Q}_H] - \mathbb{E}[\mathrm{Q}_H]| \\ &\le |\mathbb{E}[\mathrm{Q}] - \mathbb{E}[\mathrm{Q}_H]| + \mathcal{C}_\phi \sqrt{\mathbb{V}[\mathbb{E}[\mathrm{Q}_H]]},\end{aligned} \tag{4.10}$$

where $\mathcal{C}_\phi = \Phi^{-1}(1-\frac{\phi}{2})$ and Φ is the cumulative distribution function of the standard normal distribution $\mathcal{N}(0,1)$.

The variance of the MLMC expected value estimator is

$$\mathbb{V}[\mathrm{E}^{\mathrm{MLMC}}[\mathrm{Q}_H]] = \sum_{l=0}^{L} \frac{\mathbb{V}[Y_l]}{N_l}, \tag{4.11}$$

while for MC it simplifies to

$$\mathbb{V}[\mathrm{E}^{\mathrm{MC}}[\mathrm{Q}_H]] = \frac{\mathbb{V}[\mathrm{Q}_H]}{N}. \tag{4.12}$$

In both cases, these variances have to be estimated. One can use the sample variance or other estimators, as the h-statistics described above.

Therefore, the estimate of the total error is $\tau = \mathrm{DE} + \mathcal{C}_\phi \mathrm{SE}$, and the convergence criterion is

$$\tau \le \varepsilon. \tag{4.13}$$

While for MLMC it is possible to approximate the DE as $\mathrm{E}^{\mathrm{MC}}[\mathrm{Q}_{H_L} - \mathrm{Q}_{H_{L-1}}]$ [111], for MC the same approximation is not possible. Then, in this case, we will consider just the SE for assessing the convergence criterion, which will read

$$\mathcal{C}_\phi \mathrm{SE} \le \varepsilon. \tag{4.14}$$

4.7 Algorithms

We report MC and MLMC algorithms. To simplify comparisons with chapter 5, we exploit h-statistics to estimate central moments.

We define $S_{l,p}$ as the power sum of level l and power p, where $p \in [1, P]$, P is the maximum order we need and the number of realizations N is omitted. Similarly, the h-statistic of level l and power p is defined as $h_{l,p}$. Since for MC there is just one level, the dependency on l is omitted. The number of iterations it is updated each time a convergence check is performed. The number of levels, of samples per level and the mesh parameters are represented by L, N and H, respectively. The left horizontal arrow denotes the update or the computation of the left value as a function of the right values.

Non-adaptive and adaptive MC algorithms are described in algorithm 1, while non-adaptive and adaptive MLMC are reported in algorithm 2. Concerning the adaptive version, the initial screening phase is represented by the first iteration inside the *while* loop. Afterward, central moments are computed and then it is possible to estimate the optimal hierarchy. As mentioned in section 5.3, upper and lower thresholds may be applied in order to be able to control the number of levels and of samples per level estimated. This allows us to better check and compare the results obtained from different algorithms.

Algorithm 1 MC

```
$N, H$ initial hierarchy
**while** convergence is not True **do**
    **if** non-adaptive **then**
        $N \longleftarrow N, it$
    **else if** adaptive **then**
        $N \longleftarrow N, it, h_p,\ p \in [1, P]$
    **end if**
    **for** $n = 0 : N$ **do**
        $\mathrm{Q}_H^{(n)} \longleftarrow \text{solver}(w^{(n)})$
        $S_p^n \longleftarrow S_p^{n-1}, \mathrm{Q}_H^{(n)}, n, p,\ p \in [1, P]$
    **end for**
    $h_p \longleftarrow S_p, N,\ p \in [1, P]$
    convergence $\longleftarrow$ equation (4.9)
    $it = it + 1$
**end while**
```

Algorithm 2 MLMC

```
$L, N, H$ initial hierarchy
while convergence is not True do
    if non-adaptive then
        $L \longleftarrow L, it$
        $N \longleftarrow N, it$
    else if adaptive then
        $L \longleftarrow L, it, h_{l,p},\ l \in [0, L],\ p \in [1, P]$
        $N \longleftarrow N, it, h_{l,p},\ l \in [0, L],\ p \in [1, P]$
    end if
    for $l = 0 : L$ do
        for $n = 0 : N_l$ do
            $\mathrm{Q}_{H_l}^{(n)} \longleftarrow \mathrm{solver}(w^{(n,l)})$
            $\mathrm{Q}_{H_{l-1}}^{(n)} \longleftarrow \mathrm{solver}(w^{(n,l)})$
            $Y_{H_l}^{(n)} = \mathrm{Q}_{H_l}^{(n)} - \mathrm{Q}_{H_{l-1}}^{(n)}$
            $S_{l,p}^{n} \longleftarrow S_{l,p}^{n-1}, Y_{H_l}^{(n)}, n, p,\ p \in [1, P]$
        end for
    end for
    $h_{l,p} \longleftarrow S_{l,p}, N_l,\ l \in [0, L],\ p \in [1, P]$
    convergence $\longleftarrow$ equation (4.9)
    $it = it + 1$
end while
```

4.8 Distributed computing

Hierarchical MC methods are highly parallelizable. In standard approaches, three different layers of parallelism are available: between levels, between samples per level, and on each realization at solver level. In order to exploit such parallelisms when running in distributed environments, different scheduling strategies can be adopted, depending on the problem under consideration. In [48], the authors propose and discuss different *static* scheduling approaches for running MLMC on supercomputers. The hierarchy is defined before the execution starts and optimizes the computational efficiency and the time to solution. Although efficient, this procedure is problem-dependent and only allows to run one iteration, without a check of the convergence criterion on the fly. In addition, it does not take into account the fact that the sampling in MC algorithms is always stochastic, so the execution time of each realization is random. Hence, a static planning *cannot* adapt the scheduling, as the execution goes, in order to optimize the usage of the available resources. In fact, as discussed in [48], *dynamic* scheduling suites best for simulations whose runtime may vary, resulting to be faster than their static counterparts.

Therefore, in [128] we propose using a dynamic scheduling approach for both single-level and multi-level methods, as problem independent as possible, which optimally adapts to the length of each execution. Our dynamic approach re-evaluates the scheduling each time a task[2] execution finishes, while the static scheduling approach introduced in [48] defines the scheduling once, when all tasks are launched. Even though this last approach can successfully handle heterogeneity in the task duration and adapt the scheduling on it, it cannot react when the actual duration differs with respect to the precomputed estimation. Instead, with our dynamic approach, the scheduling decisions are taken on the fly, providing higher adaptability. Moreover, dynamic scheduling is preferred when the workload can vary depending on the partial results of the computation, and therefore not predictable statically.

Knowing that MLMC has more levels and MC just has one, the amount of central processing units (CPUs) used by the solver at each level should be tuned accordingly to the scalability limits of the solver for the considered problem size. As a general rule, since scalability improves as the problem grows in size, the amount of assigned resources should be as high as possible, while keeping a reasonable efficiency. In addition, it must be taken into account that the amount of CPUs set for each MLMC level should be defined in such a way that the available memory per processor is enough to perform a typical execution of that level. This way, it is possible to keep a good efficiency while reducing as much as possible the imbalance between the different levels. Concerning the overall performance, the hierarchy size is the most impacting factor, and it needs to be tuned accordingly to the problem under consideration.

Standard algorithms require the presence of synchronization points. For this reason, we refer to them as *synchronous algorithms*, also to remark the difference with asynchronous algorithms, introduced in chapter 5. Hence, synchronous algorithms need to wait until a certain parameter is computed in order to resume the execution. At this exact point, the full machine is idle. This is highly inefficient when running on supercomputers, since it may occur that the machine remains idle for long periods of time. This fact is particularly important in UQ, where the runtime of each realization depends on some random data, and may result in the whole algorithm waiting for a single realization to end before going on.

[2]A task is a method or a function that is run on a HPC system.

We also present the dependency graph of one MC iteration in figure 4.2. The circles denote tasks, that are the execution units that are sent to worker nodes to be executed in parallel. Task executions are uniquely identified through a number, while the arrows mean that the following task needs to wait for the previous one to finish before being launched, since it requires some data produced by the previous task.

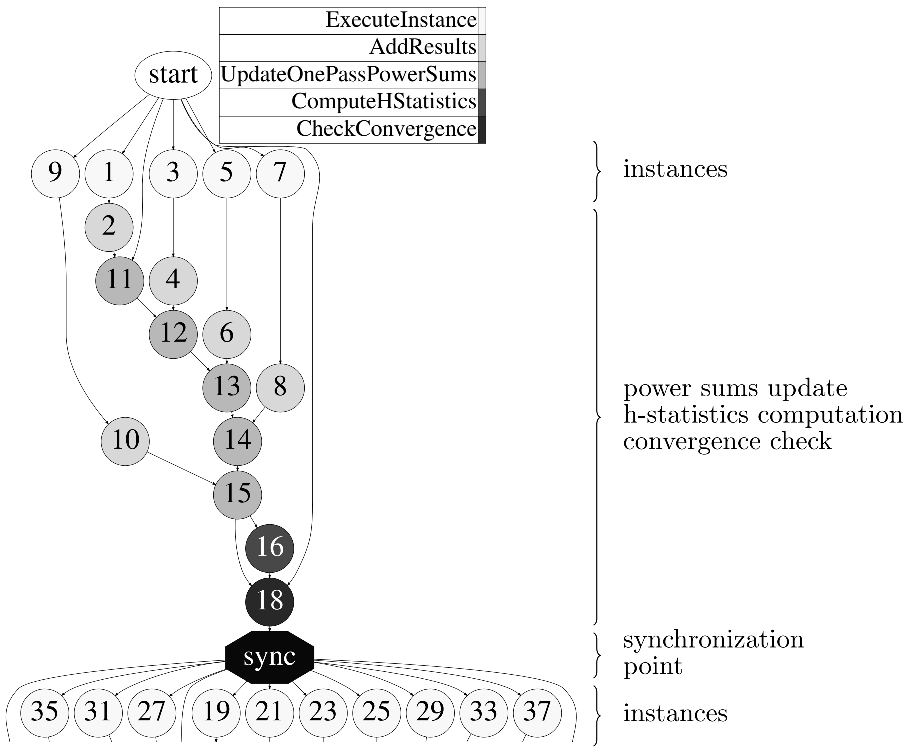


Figure 4.2: Graph connections of synchronous algorithm dependencies. The first iteration is represented. Each gray tonality identifies a different action: *ExecuteInstance* the simulation task, *AddResults* and *UpdateOnePassPowerSums* the power sums update, *ComputeHStatistics* the h-statistics computation and *CheckConvergence* the convergence check. Each circle represents a different task and is uniquely identified by a number.

It is important to realize how the dependencies shown in figure 4.2 contain an unavoidable synchronization point. In fact, we observe the requirement of waiting for the whole single hierarchy to finish before performing the statistical analysis of the QoI. Therefore, to compute the convergence (task 18), we need to wait for all the realizations to finish, and this may be highly inefficient if, for example, we are waiting for a single execution to end that is taking longer due to its random variable input value.

Figure 4.3 represents the theoretical trace of a synchronous algorithm execution, i.e.

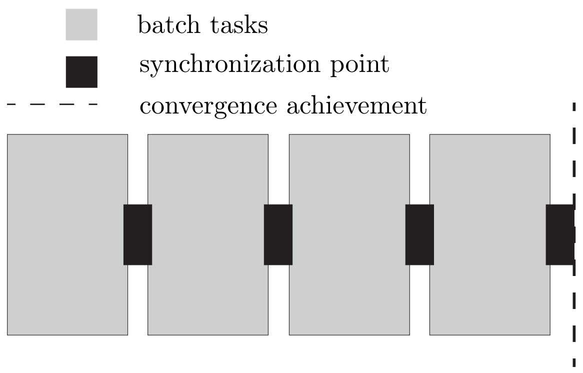


Figure 4.3: Theoretical trace of the synchronous framework. The gray rectangles represent all the different tasks, from the execution of the instances (*ExecuteInstance*) to the check of the convergence (*CheckConvergence*), which implies a synchronization point. The black rectangles are the synchronization points. The dashed vertical line denotes the achievement of the algorithm convergence after four iterations.

shows how well we are filling up the machine. The bottleneck of synchronous algorithms is clearly visible: at each iteration there is a period in which the machine is empty, therefore not executing any computation, and this moment corresponds to the synchronization point towards which all tasks need to converge at the end of each iteration. In the case of figure 4.3 we have four iterations, then four periods in which only a small part of the machine is busy. As previously mentioned, these idle times can be very costly, since we cannot know a priori their time length. As a consequence, to gain computational efficiency, the idea is to minimize as much as possible the idle times, by developing an asynchronous framework, introduced in chapter 5. The dependency graph of MLMC follows as a consequence, and for the sake of simplicity we choose to report only the MC dependency graph.

4.9 Validation

To validate our implementation of synchronous, state of the art, hierarchical MC methods, we solve a stochastic two-dimensional diffusion problem with the MC and the MLMC methods. The problem is taken from [86, section 5.2] and reads

$$-\Delta\phi = f \quad \text{in } D, \tag{4.15}$$

where $D = [0, 1] \times [0, 1]$ is the domain, $f = -432\zeta(x^2 + y^2 - x - y)$ and $\zeta \sim \mathrm{Beta}(2, 6)$ is a random variable described by the Beta distribution. The system is integrated with homogeneous Dirichlet boundary conditions in the boundaries. The solution of the problem for $\zeta = 0.25$ is represented in figure 4.4. The QoI we consider is the integral of

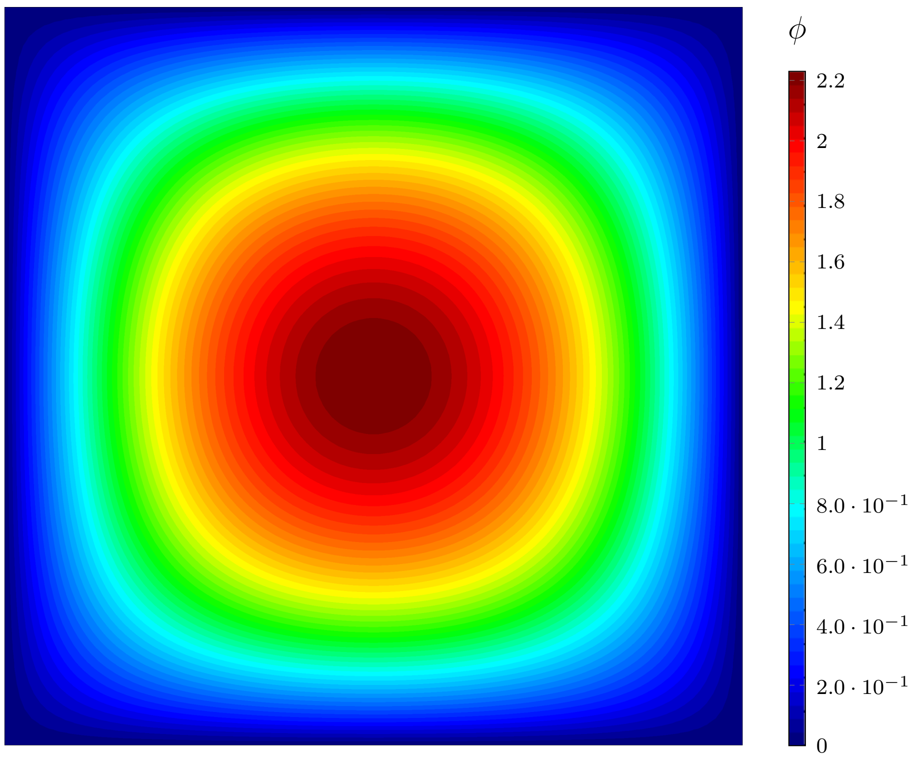


Figure 4.4: Domain and values of the solution variable ϕ of the diffusion problem.

the solution ϕ over the whole domain, that is

$$\mathrm{Q} = \int_D \phi \, \mathrm{d}x \, \mathrm{d}y, \tag{4.16}$$

whose analytical value is 6ζ. Such a problem is particularly interesting because it is possible to analytically estimate the statistical moments of the QoI. We aim at estimating the expected value of Q, whose correct value is 1.5.

We start analyzing the available mesh discretizations and observing how the DE and the variance decay as levels increase. We consider 8 levels with uniform discretization. We start from a mesh size of 0.337 m for the first level, and the mesh size decreases as 2^{-l} as the level l grows. The DE is computed as $\left|\mathrm{E}^N[\mathrm{Q}_{H_l}] - 1.5\right|$, with N large enough, and its decay can be observed in figure 4.5. The variance is estimated with the h-statistics estimator, again with enough realizations N, and its decay is plotted in figure 4.6. It

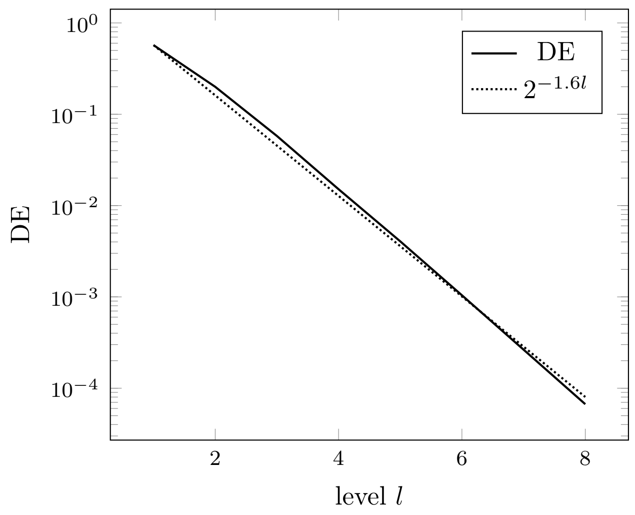


Figure 4.5: DE decay of the diffusion problem.

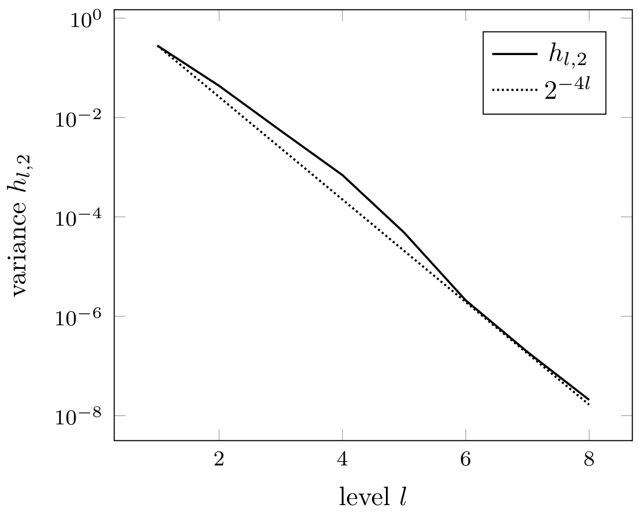


Figure 4.6: Variance decay of the diffusion problem. The variances are estimated using the h-statistics estimator, that is $h_{l,p} \equiv h^N_{l,p}[\mathrm{Q}]$.

is interesting to observe the following points about MLMC hypotheses (see section 4.3, [58, complexity theorem 3.1] and [111, section 2.2]).

- Both convergence rate exponents ($\alpha = 1.6$ and $\beta = 4.0$) are greater than 0.
- The computational cost of each levels grows with rate $2^{1.405l}$, therefore $\gamma = 1.405$.

- The constraint $\alpha \geq \min\{\beta, \gamma\}$ holds.
- $\beta > \gamma$ implies that the computation effort is primarily on the coarsest levels.

Therefore, the convergence rate exponents we obtain are in agreement with MLMC hypotheses and it follows that the MLMC method is more convenient than the MC method for solving this stochastic problem.

We can then solve the problem using the MC method. A tolerance ε of 0.13 and a confidence $1-\phi$ of 0.99 are considered. The convergence criterion we use is equation (4.13). The problem is run on the finest available mesh. For satisfying the failure probability criterion, $N = 320$ realizations on the finest mesh are run. The overall computational cost of solving the problem with the MC method is 1404 s.

Then, we solve the problem using the MLMC method. The same tolerance, confidence and convergence criterion of the MC case are considered. For satisfying the failure probability criterion, $N = (500, 200, 100, 75, 50, 25, 25, 25)$ realizations per level are run, where members of N are sorted in ascending order. The overall computational cost of solving the problem with the MLMC method is 148 s, thus showing the superiority of MLMC over MC for solving this problem.

Chapter 5

Asynchronous hierarchical Monte Carlo methods

In this chapter, we introduce and develop *asynchronous hierarchical MC methods*, which are designed to fully exploit concurrency capabilities of modern HPC systems. The proposed framework allows bypassing the expensive screening phase, preserving statistical reliability and avoiding the presence of classical synchronization points, thus improving the overall computational efficiency. In section 5.1 we present the asynchronous framework, focusing on its workflow and on its differences with respect to standard, synchronous algorithms of chapter 4. Some computational considerations on the update of statistics are presented in section 5.1.1. In section 5.2 the asynchronous MC and the asynchronous MLMC algorithms are described. Results comparing synchronous algorithms of chapter 4 and asynchronous algorithms of chapter 5 are presented in section 5.3. Scalability tests of the implemented asynchronous methods are shown in section 5.4. Concluding remarks end the chapter in section 5.5.

The content of this chapter is taken from the preprint of our publication [128] and is adapted and integrated wherever needed.

5.1 The asynchronous framework

As we have seen in chapter 4, a single slow simulation, for example due to particular random variable values or to system malfunctions, can simply lead to a huge hardware usage inefficiency, caused by a high percentage of the available resources being idle during a long time. We present here an asynchronous framework that can be applied to hierarchical MC algorithms, allowing for a continuous feed of the machine, when running in a distributed environment.

The main idea is to add a new level of parallelism, between *batches*, where each batch is defined by its own hierarchy, and to fill the idle resources while finishing a batch computation, just in case the convergence is not achieved. This way, part of the work needed for a hypothetical new batch will be already started when checking the convergence. It is clear that, when converging, we will always be discarding some work already done under the non-convergence hypothesis. Nevertheless, this methodology allows having a much better usage of the resources. This can be interpreted as a sort of pre-fetching, which consists of optimistically performing computational work that may turn out to be useful. Pre-fetching has been already applied with success to other Monte Carlo methods, as shown in [27, 8].

The update of the statistics of all QoIs and the convergence check of the algorithm are performed on the fly. As it is shown in section 4.5, we consider two possible ways of computing the central moments, respectively equations (4.5) and (4.8). The strategy we follow is to update the central moments using the second option, and the advantage of this choice is that in both equations (4.7) and (4.8) there is no dependency on the central moment values of previous steps.

The idea we follow is to work in batches, each one with its own hierarchy, hereafter called batch hierarchy. Each batch updates its *local power sums*, which afterward add their contribution to the *global power sums*. In order to preserve the correctness of the statistical analysis, it is important to avoid introducing bias, i.e. to consider only the fastest samples of our problem. Then, the update from local (of a single batch) to global takes place only after all the simulations of a single batch finish, and the batches are considered in the same order that they are spawned. Therefore, we are able to update the statistics and to check the convergence of the scheme, while in the meanwhile other batches are running. This way, we avoid synchronization points that can leave idling the hardware. Each time convergence is not achieved, a new batch is launched in order to guarantee that the resources are kept busy. On the contrary, if converged, the remaining analyses are unnecessary and can be stopped, and all the executions of the incomplete batches are discarded, in order to avoid the bias of considering only the fastest solutions of a batch.

We see the described behavior in figure 5.1. Here we start the simulation by running

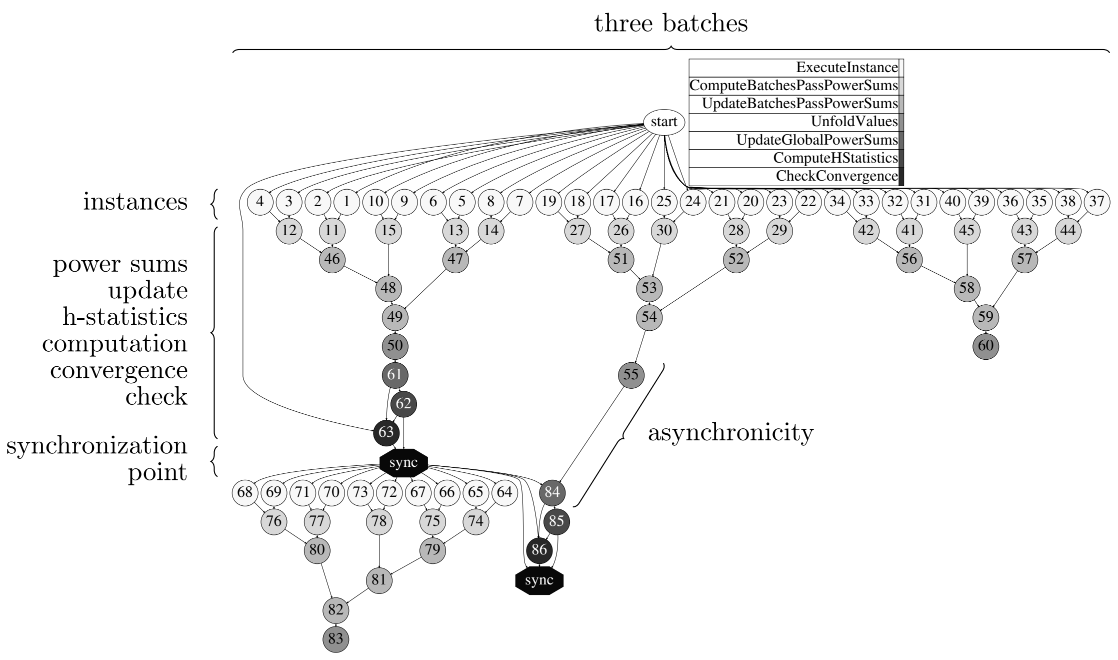


Figure 5.1: Graph connections of synchronous algorithm dependencies. Each gray tonality identifies a different action: *ExecuteInstance* the simulation task, *ComputeBatchesPassPowerSums* and *UpdateBatchesPassPowerSums* the in-batch power sums update, *UpdateGlobalPowerSums* the global power sums update, *ComputeHStatistics* the h-statistics computation and *CheckConvergence* the convergence check. Each circle represents a different task and is uniquely identified by a number.

three batches: tasks $1 - 10$, $16 - 25$ and $31 - 40$ stand for batch number one, two and three, respectively. Focusing now only on the first batch, we observe that its instances update on the fly the local power sums, and all the contributions are taken into account to check the convergence. Then, there is the synchronization point of the first batch, but, differently from what happens in the synchronous framework, this runs in parallel with other batches. In fact, the local power sums of the second batch (task 55) are directly updating the global power sums of the successive iteration (task 84), guaranteeing asynchronicity. Therefore, the difference between figures 4.2 and 5.1 is clear, and we see how this framework allows having more executions running while the statistical analysis of a single batch hierarchy is being computed.

As before, we can appreciate how the machine is being filled up through a trace in figure 5.2. With the current framework, the parallelism between the single-batch synchronization point and other batches can be observed. Moreover, we acknowledge that all the tasks that would have been executed after the convergence achievement are stopped,

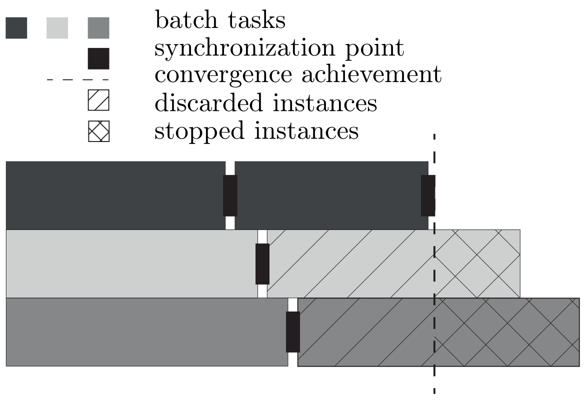


Figure 5.2: Theoretical trace of the asynchronous framework. Different gray colors of the rectangles represent different bacthes. Each batch contains all the tasks from the execution of the instances (*ExecuteInstance*) to the check of the convergence (*CheckConvergence*). The black rectangles are the synchronization points. The dashed vertical line denotes the achievement of the algorithm convergence after four iterations. The two types of parallel diagonal lines indicate that the instances of the unfinished batches are discarded (left with respect to the dashed line) or stopped (right with respect to the dashed line).

while the ones already computed and belonging to unfinished batches are discarded, thus preserving the efficiency and the accuracy of the framework.

This asynchronous approach adds one new level of parallelism to the MC family, which now are:

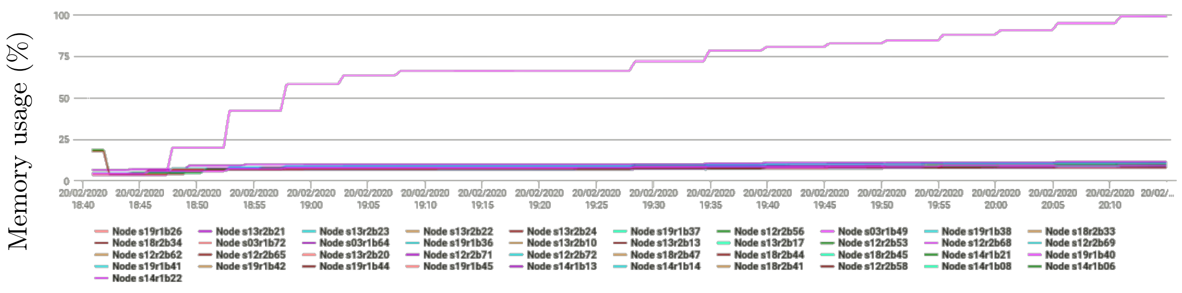
- between batches,
- between levels,
- between samples per level,
- on the simulation of each sample.

The advantages of having many small batches running give efficiency and reliability to the method we propose. The former is guaranteed since the expensive screening phase is no more needed and the number of samples is not exceeding much the optimal hierarchy to achieve convergence, considering it is directly related to the batch size hierarchy, which is non-adaptively updated. Reliability is ensured by the stopping criterion conditions and the fact that unbiased estimators are exploited to estimate central moments.

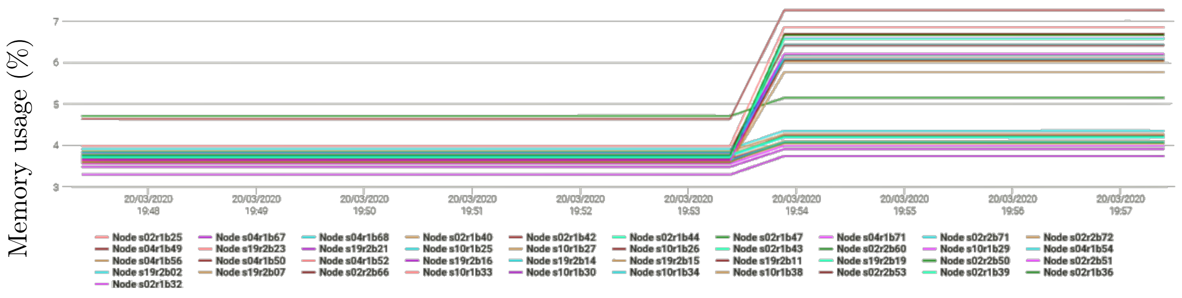
5.1.1 Parallel in-batch analysis

Considering that the convergence parameters are computed from a sum that is both associative and commutative, it is possible to compute the central moments following a tree scheme. To fully exploit the power sums potentiality, the in-batch and in-level update of the power sums $S_{b,l,p}$ are performed in parallel, as we observe in figure 5.1, in tasks $11-15, 46-49$. This way, we increase the parallelism and we reduce substantially the sequentially dependent code.

We remark that the tasks updating power sums can be grouped together, to reduce the overall number of tasks and the number of parameters that the programming model needs to handle at runtime. This permits large reductions of memory usage when many realizations and many QoIs are considered. For instance, figure 5.3a shows that the memory used by the simulation exceeds the maximum memory threshold (100%) and the job is therefore stopped. On the other hand, the same simulation is successfully run when tasks are grouped, as we see in figure 5.3b, where the memory usage is kept low.



(a) Memory usage without grouping of in-batch and in-level update tasks. The simulation is run on the MareNostrum 4 HPC system, using 40 worker nodes (1920 cores).



(b) Memory usage with grouping of in-batch and in-level update tasks. The simulation is run on the MareNostrum 4 HPC system, using 40 worker nodes (1920 cores).

Figure 5.3: Memory usage with and without grouping of tasks.

5.2 Asynchronous algorithms

The update of the number of batches B, of the number of levels L and of the number of samples per level N are only functions of the iteration counter it and of their previous values. We define $S_{b,l,p}$ as the local power sum of batch b, level l and power p, where $p \in [1, P]$, P is the maximum order we need and the number of realizations N is omitted. The global power sum of level l and order p is defined as $S_{G,l,p}$. Analogously, the h-statistic of level l and power p is denoted with $h_{l,p}$. The left horizontal arrow denotes the update or the computation of the left value as a function of the right values.

We present in algorithm 3 the asynchronous MC algorithm, where we omit to specify level $l = 0$.

Algorithm 3 Asynchronous MC

```
$B, N, H$ initial hierarchy
while convergence is not True do
    $B \longleftarrow B, it$
    $N \longleftarrow N, it$
    for $b = 0 : B$ do
        for $n = 0 : N_b$ do
            $\mathrm{Q}_H^{(n)} \longleftarrow \mathrm{solver}(w^{(n)})$
            $S_{b,p}^{n} \longleftarrow S_{b,p}^{n-1}, \mathrm{Q}_H^{(n)}, n, p,\ p \in [1, P]$
        end for
        $N = N + N_b$
        $S_{G,p} = S_{G,p} + S_{b,p},\ p \in [1, P]$
    end for
    $h_p \longleftarrow S_{G,p}, N,\ p \in [1, P]$
    convergence $\longleftarrow$ equation (4.9)
    $it = it + 1$
end while
```

MLMC follows the same idea, as shown in algorithm 4.

5.3 Comparisons between synchronous and asynchronous methods

In order to verify the accuracy, the efficiency and the computational improvements of the proposed methods, two different test cases are solved. The first, presented in sec-

Algorithm 4 Asynchronous MLMC

```
$B, L, N, H$ initial hierarchy
while convergence is not True do
    $B \longleftarrow B, it$
    $L \longleftarrow L, it$
    $N \longleftarrow N, it$
    for $b = 0 : B$ do
        for $l = 0 : L_b$ do
            for $n = 0 : N_{b,l}$ do
                $\mathrm{Q}_{H_l}^{(n)} \longleftarrow \mathrm{solver}(w^{(n,l)})$
                $\mathrm{Q}_{H_{l-1}}^{(n)} \longleftarrow \mathrm{solver}(w^{(n,l)})$
                $Y_{H_l}^{(n)} = \mathrm{Q}_{H_l}^{(n)} - \mathrm{Q}_{H_{l-1}}^{(n)}$
                $S_{b,l,p}^{n} \longleftarrow S_{b,l,p}^{n-1}, Y_{H_l}^{(n)}, n, p,\ p \in [1, P]$
            end for
            $N_l = N_l + N_{b,l}$
            $S_{G,l,p} = S_{G,l,p} + S_{b,l,p},\ p \in [1, P]$
        end for
    end for
    $h_{l,p} \longleftarrow S_{G,l,p}, N,\ l \in [0, L],\ p \in [1, P]$
    convergence $\longleftarrow$ equation (4.9)
    $it = it + 1$
end while
```

tion 5.3.1, describes the two-dimensional steady-state compressible flow of a fluid around an airfoil NACA 0012. Successively, in section 5.3.2, a two-dimensional analysis simulating a high viscosity fluid flow past a square obstacle is analyzed. Both examples are characterized by the presence of random parameters, which lead to the necessity of analyzing and studying their uncertainty propagation across the system and their influence on the QoI of the problem.

The analysis is carried out comparing the computational cost and the time to solution needed by each algorithm for satisfying the convergence criterion. The asynchronous framework is compared against two synchronous approaches: the standard non-adaptive method and one replicating the adaptive behavior, which minimizes the number of iterations. For the two non-adaptive approaches, the hierarchy update is non-adaptive, while for the adaptive method the number of iterations is reduced, and the update of the number of levels and realizations per level is controlled by lower and upper thresholds, which balance the hierarchy of the execution to properly compare the computational efforts. In all these cases the initial number of levels L and the amount of samples per level N_l is the same, and is the result of an a priori strategy which optimizes the batch execution. For MLMC, this is integrated with a MLMC estimate based on precomputed variance and computational cost estimates.

For solving the CFD problems, we use Kratos [44, 43] as FE solver software, XMC [13] as hierarchical MC library and PyCOMPSs [16, 88, 126] as programming model for distributed computing. The integration of these software has been an important part of this thesis work, as documented in [127, 10, 11, 5, 4, 22].

The analyses were run on MareNostrum 4. This supercomputer has 11.15 Petaflops of peak performance, which consists of 3456 compute nodes equipped with two Intel R Xeon Platinum 8160 (24 cores at 2.1 GHz each) processors. The analyses are performed exploiting 16 worker nodes, which account for 768 cores. Moreover, for the latter and more challenging problem, a scalability test is provided.

In the tables presenting the results, B identifies the initial number of batches, defined by the initial hierarchy, it the number of iterations, that is the amount of convergence checks executed, N the total number of realizations computed per level at the end of the execution, $\mathrm{E}^{\mathrm{MC}}[\mathrm{Q}_H]$ or $\mathrm{E}^{\mathrm{MLMC}}[\mathrm{Q}_H]$ the expected value estimation, SE the statistical error, time to solution the total time the simulation required to finish, measured in seconds, and C the computational cost of the algorithm run, expressed in CPU hours. The considered algorithms are summarized in table 5.1.

algorithm	abbreviation
asynchronous non-adaptive MC	AMC
synchronous non-adaptive MC	SMC
synchronous adaptive MC	SAMC
asynchronous non-adaptive MLMC	AMLMC
synchronous non-adaptive MLMC	SMLMC
synchronous adaptive MLMC	SAMLMC

Table 5.1: Summary of considered algorithms.

Even though the aim of this work is not to show the superiority of MLMC over MC, some considerations regarding this topic are provided. We remark that a DE estimation is not available when running the MC method standalone (if an analytical solution is not available), and for this reason we exploit equation (4.14) for assessing MC convergence. On the other hand, we use equation (4.13) to assess MLMC convergence. For this reason, the tolerance ε presents different values when solving the stochastic problems with MC or with MLMC.

5.3.1 Compressible potential flow problem

Governing equations

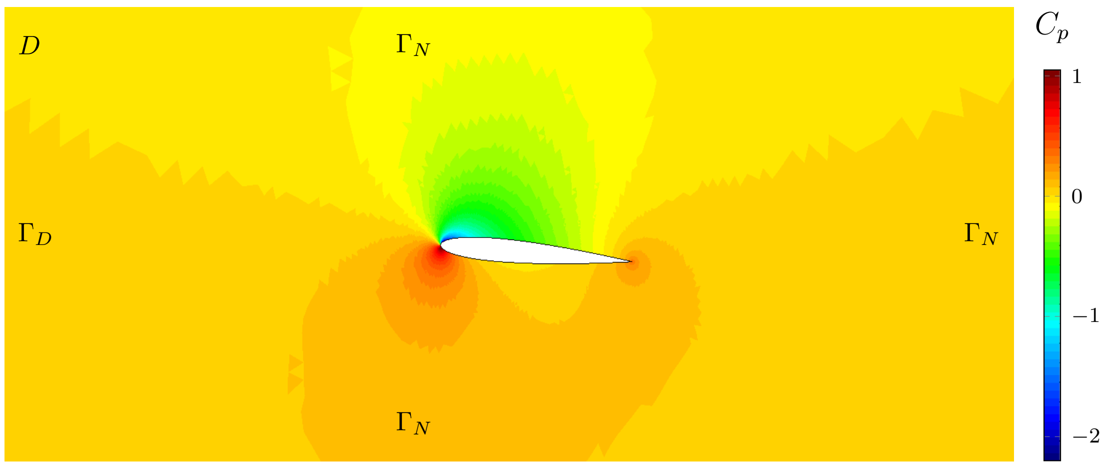


Figure 5.4: Domain and pressure coefficients of the compressible potential flow problem.

It is known that when the fluid proceeds at high Re and small angle of attack with respect to the airfoil, the flow can be considered steady-state, compressible and isentropic. Therefore, the analysis of such a two-dimensional flow around a NACA 0012, whose sketch can be observed in figure 5.4, is governed by the steady-state potential equation. The pressure coefficient shown in figure 5.4 is an adimensional number and is computed as $C_p = \frac{p - p_\infty}{\frac{1}{2}\rho_\infty u_\infty^2}$, where p_∞, ρ_∞ and $\boldsymbol{u}_\infty$ are the freestream pressure, density and velocity, respectively [47, chapter 8]. The partial differential equation describing the physical system is:

$$\nabla \cdot (\rho \nabla \Phi) = 0 \quad \text{in } D, \tag{5.1}$$

where ρ is the density and Φ the velocity potential. A stochastic inlet boundary condition on Γ_D is considered:

$$\Phi = \Phi_\infty \quad \text{on } \Gamma_D, \tag{5.2}$$

where $\Phi_\infty = f(M_\infty, \alpha)$ is function of the Mach number M_∞ and of the angle of attack of the airfoil α, and both these quantities are random. The PDF of the two stochastic variables are $M_\infty \sim \mathcal{N}(0.3, 0.003)$ and $\alpha \sim \mathcal{N}(5.0, 0.05)$, respectively. $\mathcal{N}(\mu, \sigma^2)$ denotes a normal distribution of mean μ and variance σ^2, and the angle of attack measures are expressed in degrees. Moreover, a Neumann condition is defined on Γ_N and a wake boundary condition is imposed in order to properly describe the wake generated by the airfoil. We refer to [46] for further details about the problem definition.

The output quantity of interest of the problem is the lift coefficient C_l of the airfoil, and the problem is studied with both MC and MLMC strategies. MC realizations are performed on the finest MLMC level, therefore the two strategies present the same discretization error.

Even though the physical analysis of the results is outside the scope of this work, we want to remark that in both scenarios the lift coefficient expected value estimation ($\mathrm{E}^{\mathrm{MC}}[C_l]$ or $\mathrm{E}^{\mathrm{MC}}[C_l]$) is consistent with literature results.

MC analysis

	B	it	N	$\mathrm{E}^{\mathrm{MC}}[\mathrm{Q}_H]$	SE	time to solution [s]	C [CPU h]
AMC	4	15	2700	0.6331	1.549e-6	177.7	37.9
SMC	1	15	2700	0.6340	1.538e-6	280.3	59.8
SAMC	1	3	2700	0.6340	1.525e-6	211.3	45.0

Table 5.2: Results of the MC algorithms running the compressible potential flow problem. Q_H is the lift coefficient C_l, time to solution is measured in seconds, and the computational cost C in CPU hours.

The problem is run considering a confidence $(1-\phi) = 0.99$ and a tolerance $\varepsilon = 0.004$. We exploit equation (4.14) for assessing convergence and results are presented in table 5.2.

The asynchronous algorithm is faster and cheaper compared to the other two synchronous strategies. The reason relies on the fact that asynchronous MC spawns many small batches at the beginning, therefore the machine is continuously fed and the idle time is reduced to the minimum. On the other hand, this is not happening in the other two scenarios, in which the synchronization points leave the machine underutilized for longer times, producing computational inefficiencies.

In addition, considering the synchronous algorithms, we can state that the computational efficiency, for a given amount of samples, grows as the number of iterations *it* decreases. This happens because we reduce the amount of synchronization points. Nevertheless, since the convergence check is done less frequently, we risk to do much more computations than the ones really needed to achieve the desired accuracy, if the hierarchy estimation is not accurate enough.

Even though we assess convergence for the lift coefficient expected value, other physical quantities are computed when solving the system, as the pressure coefficient. We report in figure 5.5 the risk measure expected value $\pm$ standard deviation of the pressure coefficient.

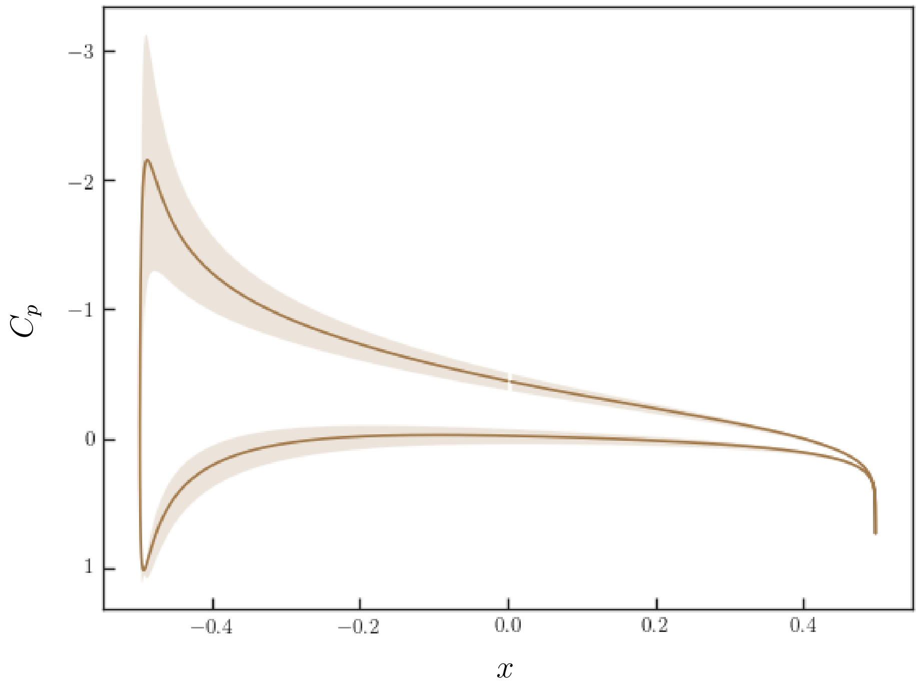


Figure 5.5: Risk measure $\mathbb{E}[C_p(x)] \pm 3\sigma[C_p(x)]$, where x denotes the horizontal coordinate.

MLMC analysis

For running MLMC, the tolerance is set to $\varepsilon = 0.03$ and the confidence is $(1-\phi) = 0.99$. Convergence is assessed with equation (4.13).

We report in table 5.3 the results of the analyses for the different strategies. The asynchronous non-adaptive MLMC outperforms the two synchronous algorithms, requiring less than half of their computational cost and time to solution. Therefore, spawning many small batches from the beginning is the best strategy.

Comparing MC and MLMC runs, the two strategies provide a comparable discretization error, since MC analyses are run on the finest MLMC level. On the other hand, the

	B	it	N	$\mathrm{E}^{\mathrm{MLMC}}[\mathrm{Q}_H]$	SE	time to solution [s]	C [CPU h]
AMLMC	4	7	4620,70,35	0.6319	8.983e-7	192.8	41.1
SMLMC	1	7	4620,70,35	0.6321	9.023e-7	466.3	99.4
SAMLMC	1	3	4620,70,35	0.6314	8.749e-7	448.1	95.6

Table 5.3: Results of the MLMC algorithms running the compressible potential flow problem. Q_H is the lift coefficient C_l, time to solution is measured in seconds, and the computational cost C in CPU hours.

best MLMC strategy provides a smaller statistical error than the best MC approach, for the same computational cost. Then, for the same computational cost, asynchronous non-adaptive MLMC gives a smaller total error than asynchronous non-adaptive MC.

Traces

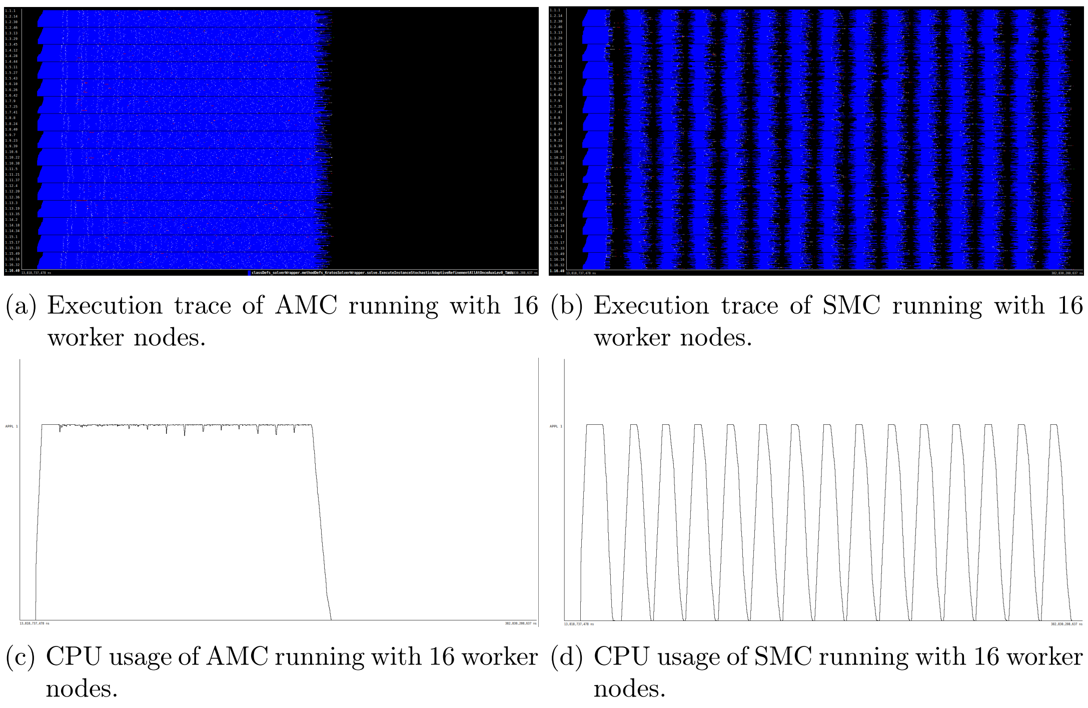

(a) Execution trace of AMC running with 16 worker nodes.
(b) Execution trace of SMC running with 16 worker nodes.
(c) CPU usage of AMC running with 16 worker nodes.
(d) CPU usage of SMC running with 16 worker nodes.

Figure 5.6: Execution traces of asynchronous and synchronous algorithms running the compressible potential flow problem.

Apart from looking at the results of tables 5.2 and 5.3, a computational efficiency comparison can be done also looking at how the algorithms feed the HPC machine.

In figure 5.6a we report the execution trace of the AMC algorithm, run with 16 worker nodes, i.e. 768 cores, and performing 15 iterations. The black spaces denote when the CPU has not received any task to execute. The absence of these spaces in the middle of the execution shows that the algorithm has no apparent synchronization points when analyzing its behavior.

Figure 5.6c shows the CPU usage of the AMC run. We observe the machine is constantly working at its maximum potential for the whole simulation, confirming we obtained the desired behavior. In fact, the main point here is that the asynchronous batch spawning and the dynamic scheduling of the tasks produce an excellent level of resources utilization. Indeed, the real behavior is the one expected, maximizing the computational efficiency and reducing to the minimum the idle time. On the other hand, SMC trace and CPU efficiency are reported in figures 5.6b and 5.6d. We can observe that at each convergence check, the machine is idle, and this is the cause of the larger time to solution required by synchronous approaches. Since in figure 5.6 all plots have the same time scale, a black space is present in figure 5.6a at the end of the execution, and means the simulation finishes before. Similar traces are obtained for synchronous adaptive MC and the three MLMC executions.

5.3.2 Simplified CFD problem

Governing equations

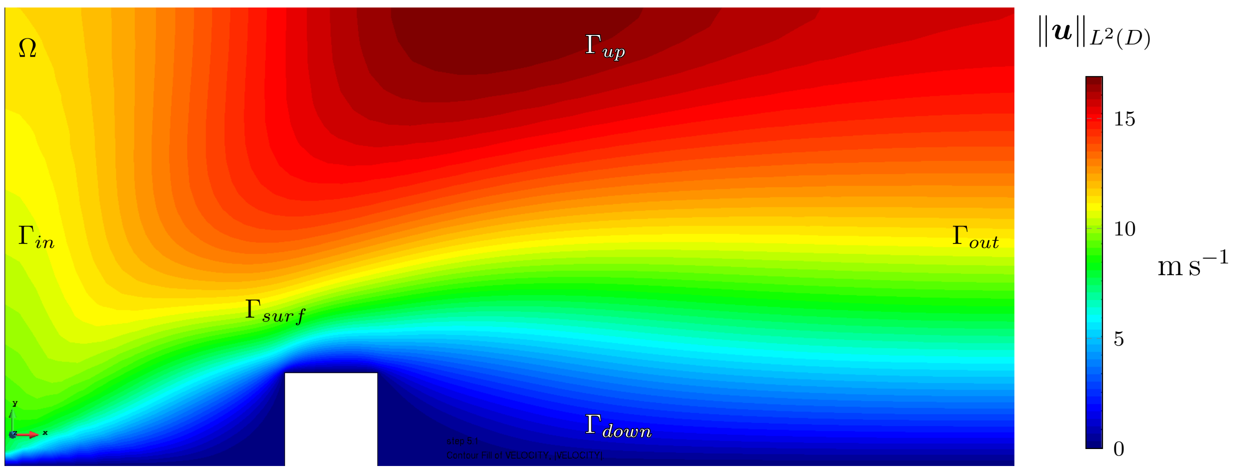


Figure 5.7: Domain and velocity field of the simplified CFD problem.

In the second numerical example, we aim at studying the two-dimensional flow past a square obstacle in a domain D and a time interval $[0, T]$, whose behavior is described by the incompressible Navier-Stokes equations, introduced in chapter 2 and reported here for the sake of simplicity:

$$\begin{aligned} \frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} - \nu \Delta \boldsymbol{u} + \nabla p = \boldsymbol{f} \qquad & \text{in } [0, T] \times D \\ \nabla \cdot \boldsymbol{u} = 0 \qquad & \text{in } [0, T] \times D, \end{aligned} \tag{5.3}$$

where $\boldsymbol{u}$ is the velocity field, p the pressure field, ν the kinematic viscosity and $\boldsymbol{f}$ the vector field of body forces. The Re of the problem is 1.

The boundary condition defined in the inlet Γ_{in} is stochastic and constant in time. The flow in the inlet follows the power law

$$\begin{aligned} \boldsymbol{u} \cdot \boldsymbol{n} = \overline{\boldsymbol{u}} \Big(\frac{z}{z_0} \Big)^{\alpha} \qquad & \text{on } \Gamma_{in} \\ \boldsymbol{u} \cdot \boldsymbol{n}^{\perp} = 0 \qquad & \text{on } \Gamma_{in}, \end{aligned} \tag{5.4}$$

where $\boldsymbol{n}$ is the unit normal on ∂D, $\overline{u} \sim \mathcal{N}(10, 1.0)$, $\alpha \sim \mathcal{N}(0.12, 0.012)$, z and z_0 the vertical coordinate and a reference height, respectively, and $\mathcal{N}$ denotes a normal distribution. Wall boundary conditions are applied to Γ_{surf}, free slip conditions to Γ_{up}, zero flux boundary conditions to Γ_{out} and no slip conditions to Γ_{down} [6].

The output QoI of the system is the drag force F_d computed on the surface Γ_{surf}. Our goal is to compare the synchronous and asynchronous approaches for both MC and MLMC. MC is run on the finest MLMC level, thus all algorithms will have a comparable discretization error.

The choice of a case like this is particularly relevant, since it is a realistic and challenging simplification of CFD engineering problems, and we aim at showing how adopting the asynchronous framework increases the overall computational efficiency.

MC analysis

The tolerance is set to $\varepsilon = 0.5$ and the confidence is $(1 - \phi) = 0.99$. We remark that ε is a dimensional quantity. Equation (4.14) is exploited for assessing convergence.

In table 5.4 the results of the MC algorithms are reported. We can see that the asynchronous algorithm is faster and cheaper than the two synchronous approaches. The reason is the same as before: spawning many small batches at the beginning of the

	B	it	N	$\mathrm{E}^{\mathrm{MC}}[\mathrm{Q}_H]$	SE	time to solution [s]	C [CPU h]
AMC	4	15	2700	5018.9	0.0335	1034.0	220.6
SMC	1	15	2700	5019.2	0.0306	1193.3	254.5
SAMC	1	3	2700	5019.1	0.0322	1116.5	238.2

Table 5.4: Results of the MC algorithms running the simplified CFD problem. Q_H is the drag force F_d, time to solution is measured in seconds, and the computational cost C in CPU hours.

simulation is more convenient than launching one per iteration, even in the case of a reduced number of synchronization points.

MLMC analysis

	B	it	N	$\mathrm{E}^{\mathrm{MLMC}}[\mathrm{Q}_H]$	SE	time to solution [s]	C [CPU h]
AMLMC	4	15	3600,1200,600	5019.0	0.0217	958.5	204.4
SMLMC	1	15	3600,1200,600	5019.0	0.0225	2534.3	540.6
SAMLMC	1	3	3600,1200,600	5019.3	0.0211	1042.4	222.3

Table 5.5: Results of the MLMC algorithms running the simplified CFD problem. Q_H is the drag force F_d, time to solution is measured in seconds, and the computational cost C in CPU hours.

For the MLMC analysis, the tolerance is set to $\varepsilon = 2.5$ and the confidence is $(1-\phi) = 0.99$. Convergence is assessed with equation (4.13).

In table 5.5 we can observe the results obtained with the three different MLMC strategies. Once more, the asynchronous method is the cheapest and fastest, compared to the synchronous algorithms. Even reducing a lot the number of iterations, the asynchronous strategy is the most efficient.

Similarly to the previous problem, we compare MC and MLMC. We can observe that, for the same computational cost, MLMC presents a smaller statistical error, thus asynchronous MLMC is the preferred strategy. The discretization error of MC and of MLMC is the same, since MC realizations are run on MLMC finest level.

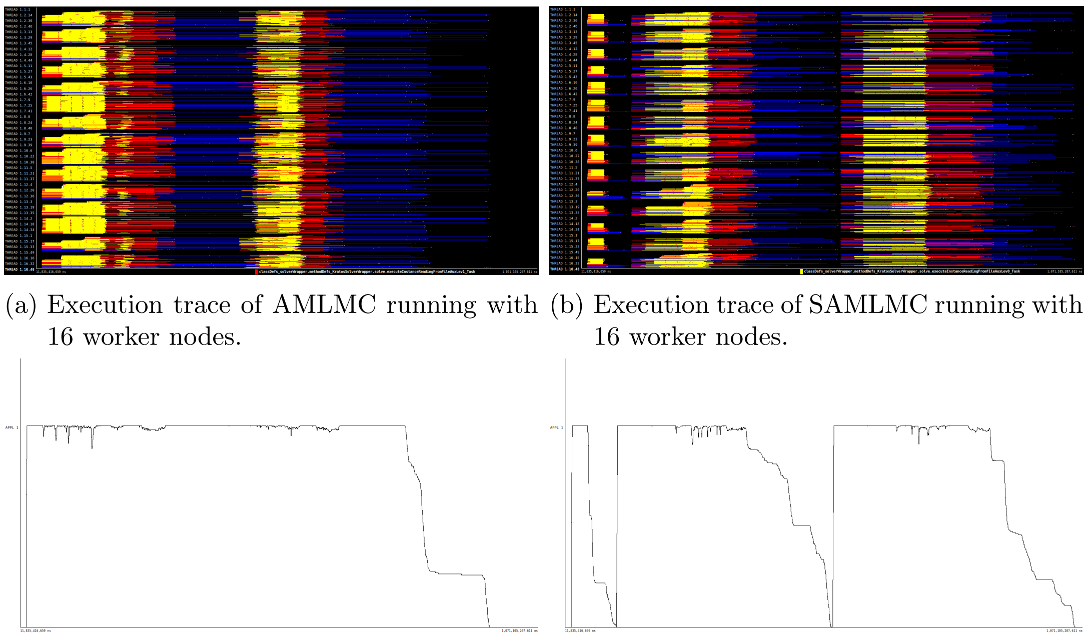

(a) Execution trace of AMLMC running with 16 worker nodes.
(b) Execution trace of SAMLMC running with 16 worker nodes.
(c) CPU usage of AMLMC running with 16 worker nodes.
(d) CPU usage of SAMLMC running with 16 worker nodes.

Figure 5.8: Execution traces of asynchronous and synchronous algorithms running the simplified CFD problem.

Traces

In figures 5.8a and 5.8c, the executing trace of the asynchronous MLMC algorithm and its CPU usage are reported, while in figures 5.8b and 5.8d we observe the trace and the CPU usage of the synchronous adaptive MLMC algorithm. The time scale is the same for all plots.

One important observation to be made is that simulations on higher levels are running on multiple threads. On the other hand, the trace only shows the main thread that actually receives the task execution order. The rest of the working threads are shown as idle. This means that for each line corresponding to a 2 CPU task, there is one apparently idle CPU that is working. For the tasks running on 4 CPUs, there are 3 CPUs that seem idle but are actually busy. In order to ease the comprehension of this fact, we have included CPU usage graphs, that take into account the real amount of CPUs that are working. Looking at both figures at the same time, it can be deduced that the yellow tasks occupy a single CPU, the red tasks 2 and the blue tasks 4. Keeping that in mind, it is possible to state that the resource usage of the MLMC algorithm is more efficient. Indeed, figure 5.8d shows that the hardware is getting idle at some points while new batches are still not launched. On the other hand, we see how the asynchronous framework continuously feeds the machine, improving the computational efficiency of the execution.

5.4 Scalability of the asynchronous framework

5.4.1 Scalability with solver using OpenMP parallelism

N	CPUs	T	Speedup	Ideal Speedup	ζ	ξ
2	96	187347.8	1.0	1	1.0	-
4	192	95515.2	1.96	2	0.98	0.02
8	384	48093.9	3.90	4	0.97	0.01
16	768	24016.9	7.80	8	0.98	0.00
32	1536	12236.2	15.31	16	0.96	0.02
64	3072	6397.7	29.28	32	0.92	0.04
128	6144	4106.1	45.63	64	0.71	0.22

Table 5.6: Scalability results of asynchronous MLMC algorithm running the simplified CFD problem. T is expressed in seconds and is the total time the algorithm required to be run, and is expressed in seconds. ζ represents the performance efficiency, and ξ the performance lost with respect to the previous level.

In addition to the previous results, a strong scalability test of the asynchronous MLMC algorithm is presented in figure 5.9 and table 5.6. The algorithm runs the simplified CFD problem described above. Different settings and batch hierarchies with respect to previous executions are considered, thus results are not comparable.

Figure 5.9 reports the speedup of the problem, i.e. the ratio between the execution time with N worker nodes, $N = \{2, 4, 8, 16, 32, 64, 128\}$, and the execution time with 2 worker nodes. In the table, N indicates the amount of worker nodes, T is the execution time of the simulation, measured in seconds, ζ represents the performance efficiency, and ξ stands for the performance lost with respect to the previous level. Both ζ and ξ are expressed in percentage, the former is given by $\zeta = \frac{\text{speedup}}{\text{ideal speedup}}$, while the latter is $\xi = 1 - \frac{\zeta_N}{\zeta_{N-1}}$. In addition, when computing the ideal speedup, we have considered that the full code is parallelizable. Indeed, we have estimated that the initial serial part takes less than 0.01% of the execution time. Moreover, the initial amount of batches and

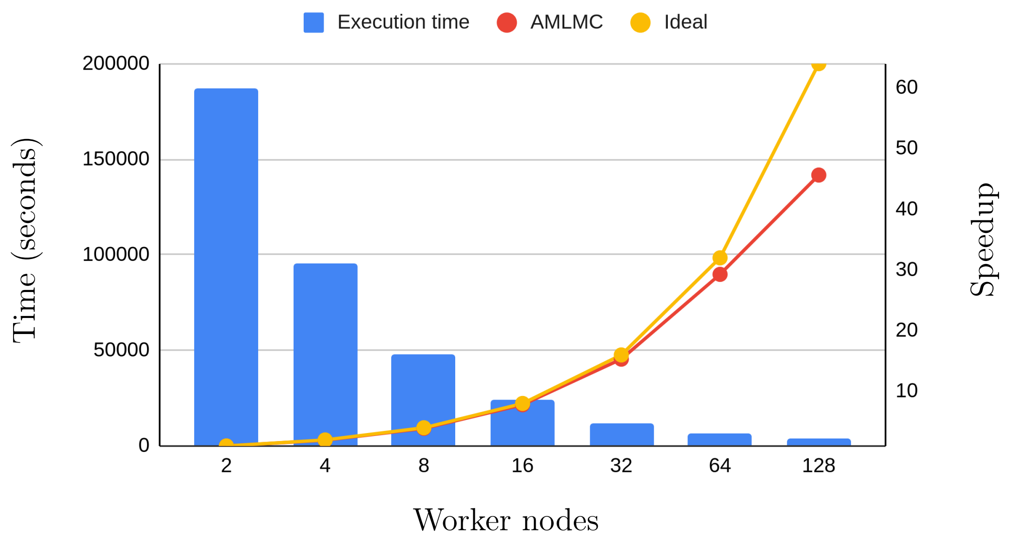


Figure 5.9: Strong scaling results of the asynchronous MLMC method integrated with the solver Kratos and the programming model for distributed computing PyCOMPSs. The baseline for calculating the scalability is the execution time obtained for 2 worker nodes. The test is run on the MareNostrum 4 supercomputer and each node has 48 CPUs.

their size are large enough to fill the whole machine at every moment. In this scenario, the Amdahl's law states that augmenting the available resources in a given ratio should decrease the execution time in the same exact linear proportion.

We can observe that the proposed algorithm implementation scales pretty well up to 128 worker nodes. Indeed, we scale almost linearly until 64 worker nodes, and only at 128 worker nodes we start to observe some parallel efficiency loss.

5.4.2 Scalability with solver using MPI parallelism

In [52] we assess the performance of the asynchronous MC method when running with Kratos as MPI parallel solver and PyCOMPSs as programming model for distributed computing. The problem we solve is the three-dimensional wind flow past a high-rise building and is presented in section 9.2.3.

We report in figures 5.10 and 5.11 the strong and weak scalability results, respectively. In both cases, the baseline for calculating the scalability is the execution time obtained for 16 worker nodes. The test is run on the MareNostrum 4 supercomputer and each node has 48 CPUs. We can observe that optimal efficiency up to 128 worker nodes (6144

CPUs) is achieved.

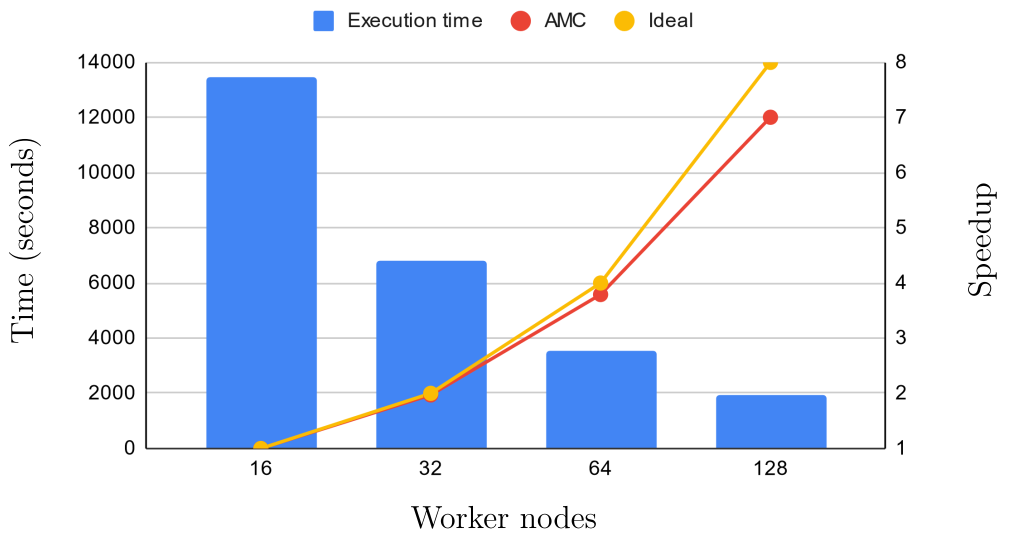


Figure 5.10: Strong scalability results of the asynchronous MC method integrated with the MPI parallel solver Kratos and the programming model for distributed computing PyCOMPSs. The image is taken from [52].

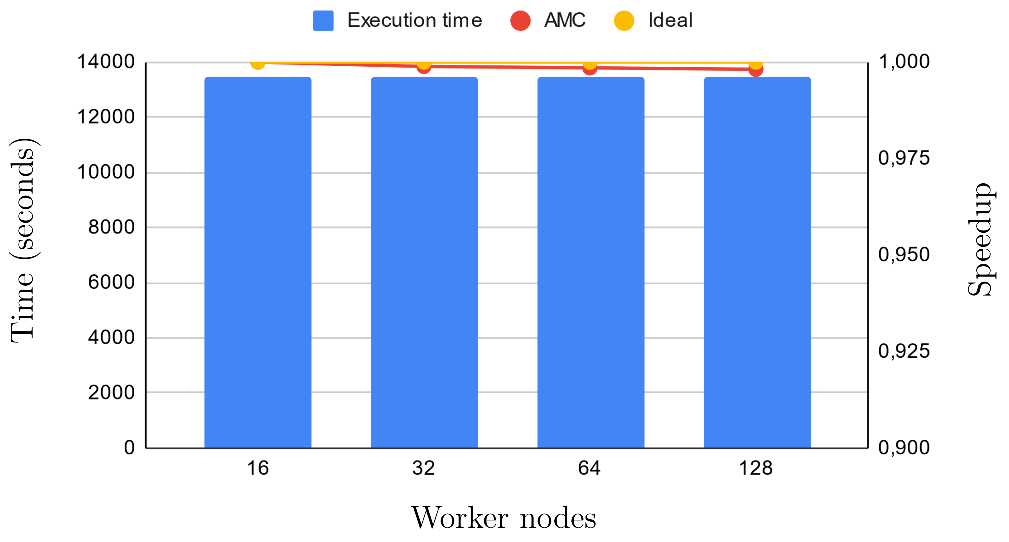


Figure 5.11: Weak scalability results of the asynchronous MC method integrated with the MPI parallel solver Kratos and the programming model for distributed computing PyCOMPSs. The image is taken from [52].

5.5 Summary

In this chapter, we have proposed a new asynchronous strategy for performing UQ analyses, built on top of hierarchical MC methods. The proposed method is particularly interesting when running on HPC environments, exploiting the concurrency capabilities of modern supercomputers.

The key feature of the asynchronous framework is the new level of parallelism, between batches, which is added to the existing parallelism between levels, samples per level, and on the solver of each realization of state of the art methods. The asynchronous methods possess both reliability and efficiency, as discussed and demonstrated.

We have analyzed the behavior of the asynchronous framework against state of the art hierarchical MC methods. In all cases, the asynchronous approach was the one with the best performance for satisfying the convergence criterion. The gain is huge against non-adaptive methods. On the other hand, if the number of iterations is minimized, the computational efficiency improves, since the idle time decreases, but never reaches the efficiency of the asynchronous approach. Moreover, we remark that the asynchronous method does not require any screening phase in order to run, since the batch hierarchy can be as small as desired.

Chapter 6

Adaptive mesh refinement for hierarchical Monte Carlo methods

In this thesis, we are interested in solving problems of engineering interest, which are computationally expensive. It may happen that some complex features take place only in some part of the domain. For this reason, using uniform meshes could be discouraged, since they provide high accuracy in regions of the domain where it may not be required. A potential solution could be using domain discretizations with multiple layers. However, such layers should be defined a priori, thus can easily be inaccurate if the system does not behave as expected. On the contrary, solution-based adaptively refined meshes do not present such issues, since they are adapted depending on features of the system we are interested in. AMR can be understood as a preprocess phase with respect to the solution of the problem.

The remeshing software we exploit is Mmg [45], which employs a metric-based adaptive refinement. In our case, we focus on Hessian-based metrics, that make use of some *error indicator* to refine the mesh. Details about the metric computation are provided in section 6.1, while the possibility of using different error indicators is described in section 6.2. Moreover, we propose an integration of AMR and the MLMC method in section 6.3, where AMR can optionally be performed on the fly.

As for other software implementation, we acknowledge that the AMR process described in section 6.1 is part of the Kratos software, developed by Kratos colleagues [94, chapter 6].

6.1 Hessian-based metric computation

In this thesis, we are interested in computing Hessian-based metric tensors. To perform AMR, we therefore need to construct a $d \times d$ nodal metric tensor $\boldsymbol{\mathcal{M}}$, where d is the domain dimension of our system. The metric tensor is symmetric positive, non-degenerated, diagonalizable and its purpose is setting sizes and directions of the refined mesh. The metric tensor can be written as [3, 2, 41]

$$\boldsymbol{\mathcal{M}} = \boldsymbol{\mathcal{R}} \widehat{\boldsymbol{\Lambda}}^t \boldsymbol{\mathcal{R}}, \tag{6.1}$$

where

$$\widehat{\boldsymbol{\Lambda}} = \operatorname{diag}(\widehat{\lambda}_i), \tag{6.2}$$

$$\widehat{\lambda}_i = \min\left(\max\left(\frac{c_d |\lambda_i|}{\epsilon}, h_{max}^{-2} \right), h_{min}^{-2} \right). \tag{6.3}$$

$\boldsymbol{\mathcal{R}}$ is the eigenvector matrix and λ_i are the eigenvalues of the Hessian $\boldsymbol{\mathcal{H}}_u$ of a scalar variable u (twice differentiable), that is representative of the system. We refer to such a variable u as error indicator. The metric tensor is function of a constant c_d (which depends on the system dimension), of the interpolation error ϵ and of h_{max} and h_{min}, respectively the maximum and minimal mesh sizes of the refined mesh. The interpolation error is defined as

$$\epsilon = \|u - u_h\|_K, \tag{6.4}$$

where $\| \cdot \|_K$ is some norm defined on each element, e.g. the $L^2(K)$-norm. In other words, it is the error we commit when estimating u with u_h, where u_h is computed on top of a discretization of the domain with characteristic size h.

In the case of tetrahedrons (denoted by K) with 6 edges, the interpolation error is bounded as [55, 2]

$$||u - u_h||_{L^\infty(K)} \leq c_d \max_{\boldsymbol{x} \in K} \max_{j=1 \dots 6} \langle e_j, |\boldsymbol{\mathcal{H}}_u(x)| e_j \rangle. \tag{6.5}$$

Citing [2], "a bound of the interpolation error in L^∞ norm is given by the square length of the largest edge of the tetrahedron computed with respect to the metric of the maximal absolute value of the Hessian". Such a relation permits to avoid computing equation (6.4), which is not viable if an analytical expression of u is not available. In fact, by using equation (6.5), the interpolation error can be set and the optimal edge lengths of the tetrahedrons can be estimated.

In case of multiple error indicators u, it is necessary to intersect the metric tensors into a final one. Each error indicator generates a metric, and when intersecting we keep the most restrictive size constraints for all directions [2].

6.2 Error indicators

The metric is computed on top of an error indicator u, which must be properly selected. As a general rule, it is important to use an error indicator that contains as much information about the system as possible. For this reason, we normally use some solution field of the partial differential equation describing the system as an error indicator.

Let us consider a CFD problem. In this context, one possibility is to use velocity and pressure fields as error indicators, which are the solution of the physical problem. Different combinations of the fields can be considered as error indicators to build the Hessian-based metric, and are presented next.

i) Time-averaged velocity field $\langle \boldsymbol{u} \rangle_{T_0,T}$. $\langle \cdot \rangle_{T_0,T}$ denotes to the time averaging operator with time window $[T_0, T]$.

ii) Time-averaged pressure field $\langle p \rangle_{T_0,T}$.

iii) Time-averaged velocity field $\langle \boldsymbol{u} \rangle_{T_0,T}$ and time-averaged pressure field $\langle p \rangle_{T_0,T}$.

iv) Velocity field $\boldsymbol{u}$. The metric is updated at each time step of the time window $[T_0, T]$ and the most restrictive metric is retained on each element.

v) Pressure field p. The metric is updated at each time step of the time window $[T_0, T]$ and the most restrictive metric is retained on each element.

vi) Velocity field $\boldsymbol{u}$ and pressure field p. The metric is updated at each time step of the time window $[T_0, T]$ and the most restrictive metric is retained on each element.

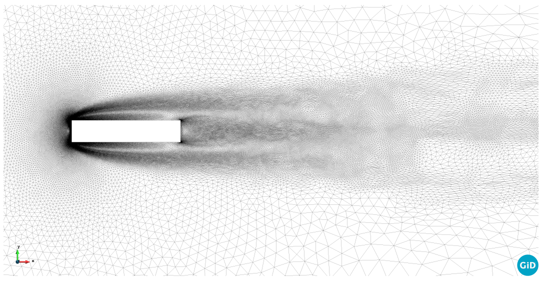
The cases i), ii), iv), v) use only one solution field, while iii) and vi) use the two solution fields. Another difference is that the error indicators i), ii) and iii) are cheaper, since the metric is computed only once at the end of the simulation, but less accurate, due to the time averaging process which dumps peaks and oscillations. On the other hand, strategies iv), v) and vi) take into account any peak that may appear in the time history to compute the metric, but present a larger computational cost, since the metric is updated at each time step.

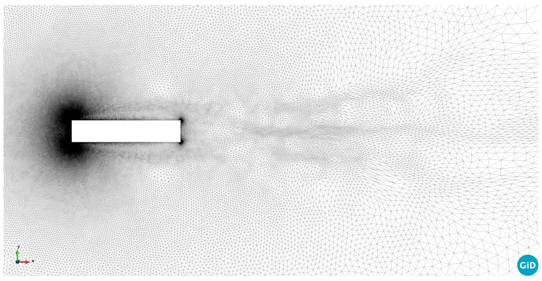
We comment that we propose a systematic way to choose the best error indicator for low Re flows around bodies in [14].

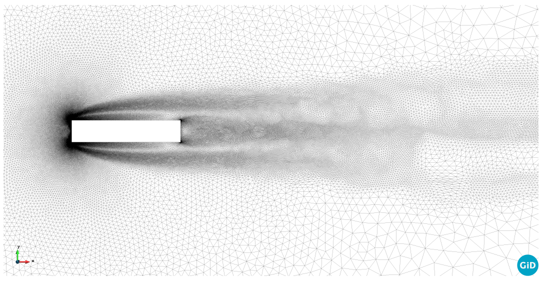
Rectangle obstacle problem To generate the mesh for solving the rectangle obstacle problem of sections 7.1, 8.2.1 and 9.2.1, the AMR strategy is performed starting from a background mesh generated by GiD [98]. The problem is solved once, for a time window of 600 s. The relative time required to compute the Hessian-based metric exploiting the velocity field, the pressure field or the velocity and pressure fields with respect to the overall simulation cost is approximately of 8.5%. On the other hand, the time to compute the Hessian-based metric using the time-averaged velocity field, the time-averaged pressure field or time-averaged velocity and time-averaged pressure fields is negligible with respect to the overall duration of the simulation.

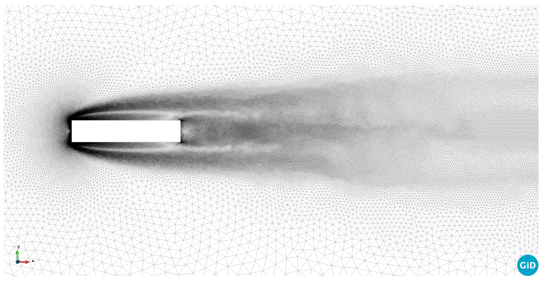
We can appreciate the difference between the error estimators discussed above in figure 6.1.

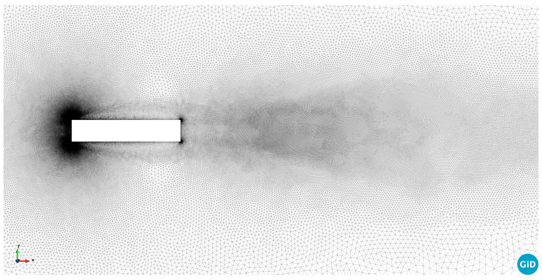
High-rise building problem To generate the mesh for solving the high-rise building problem of sections 7.2 and 9.2.3, we perform AMR. We start from a mesh generated from GiD [98], where multiple uniform layers are present. We run the problem once with such a background mesh, for a time window of 600 s, to compute the Hessian-based metric, and then we remesh. The relative cost of computing the Hessian-based metric with the velocity field, the pressure field or the velocity and pressure fields as error indicator is approximately 7% of the overall computational cost. On the other hand, the time to compute the Hessian-based metric using the time-averaged velocity field, the time-averaged pressure field or time-averaged velocity and time-averaged pressure fields is negligible with respect to the overall duration of the simulation. Using velocity and pressure fields is a convenient choice as an error indicator, since pressure and velocity fields represent the solution of the physical problems and updating at each time step the metric is not too expensive and allows us to consider peaks which may appear during the simulation. Further details about the mesh and its validation are provided in next chapters. Similar considerations hold true for the mesh used in sections 8.2.2 and 9.2.2.


(a) Time-averaged velocity field error estimator.


(b) Time-averaged pressure field error estimator.


(c) Time-averaged velocity field and time-averaged pressure field error estimator.


(d) Velocity field error estimator.


(e) Pressure field error estimator.

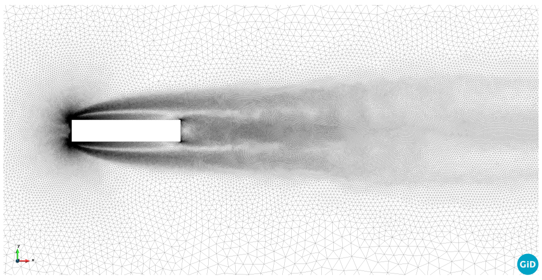

(f) Velocity field and pressure field error estimator.

Figure 6.1: Metric-based remeshing of the rectangle obstacle problem of sections 7.1, 8.2.1 and 9.2.1. The metric is computed using the six different error estimators of section 6.2 and the same interpolation error.

6.3 Adaptive refinement MLMC methods

We propose next two different algorithms, that aim at integrating AMR strategies with the MLMC method. The same approach can be easily applied to other multi-level methods, as CMLMC and to their asynchronous counterparts.

In the following, the discretization of the first level is fixed and presents an accuracy set by the user a priori. We consider two different methods for integrating AMR strategies with the MLMC method: *deterministic adaptive refinement* and *stochastic adaptive re-*

finement. The former generates deterministic adaptive meshes at the beginning of the algorithm, and the same discretizations are used for all realizations. Therefore, deterministic adaptive meshes are independent of the stochastic parameters of the problem. The second generates stochastic adaptive meshes for each realization, and therefore each discretization depends on the random parameters of the system. In both cases, meshes are controlled by a set of interpolation errors ϵ associated to each level. Moreover, we remark that, in the stochastic adaptive refinement case, the AMR procedure is called more frequently than in the deterministic adaptive refinement case.

The novelty of the two approaches we propose is that they reduce the mesh generation impact by exploiting *persistent storage* of the discretizations. This implies storing deterministic meshes once they are generated and making them available whenever needed. Alternatively, in the case of stochastic adaptive meshes, we generate them simultaneously to the running of other events, thus overlapping the two steps of the solver and of the remesher. We remark that in the case of single-level MC, persistent storage of the mesh is also applied, and the discretization is made available whenever needed.

Two different versions are currently available to perform persistent storage of domain discretizations. The former consists of storing a single serialized object when the simulation is local and all the data belongs to the same process of an HPC system. On the other hand, when the simulation is run using MPI parallelism, several processes are typically spawned across multiple computing nodes and the single-process serialization is no longer possible. Therefore, we perform a distributed serialization, which means that each MPI process is responsible for the serialization and then for the recovery of its piece of data.

We report in algorithms 5 and 6 the deterministic adaptive refinement MLMC method and the stochastic adaptive refinement MLMC method, respectively. To simplify comparisons with algorithm 2, h-statistics are used to estimate central moments, and both non-adaptive and adaptive strategies are considered.

We define $S_{l,p}$ as the power sum of level l and power p, where $p \in [1, P]$, P is the maximum order we need and the number of realizations N is omitted. Similarly, the h-statistic of level l and power p is defined as $h_{l,p}$. The number of iterations it is updated each time a convergence check is performed. The number of levels, of samples per level, the mesh parameters and the interpolation errors are represented by L, N, H and ϵ, respectively. The left horizontal arrow denotes the update or the computation of the left value as a function of the right values. $H_l \xleftarrow{\text{AMR}} \epsilon_l$ denotes the AMR process, and

L_{MAX} the maximum number of levels.

Algorithm 5 Deterministic adaptive refinement MLMC

```
$L, N, \epsilon$ initial hierarchy and interpolation errors
**for** $l = 0 : L_{\text{MAX}}$ **do**
  $H_l \xleftarrow{\text{AMR}} \epsilon_l$
**end for**
**while** convergence is not True **do**
  **if** non-adaptive **then**
    $L \longleftarrow L, it$
    $N \longleftarrow N, it$
  **else if** adaptive **then**
    $L \longleftarrow L, it, h_{l,p},\ l \in [0, L],\ p \in [1, P]$
    $N \longleftarrow N, it, h_{l,p},\ l \in [0, L],\ p \in [1, P]$
  **end if**
  **for** $l = 0 : L$ **do**
    **for** $n = 0 : N_l$ **do**
      $\mathrm{Q}_{H_l}^{(n)} \longleftarrow \text{solver}(w^{(n,l)})$
      $\mathrm{Q}_{H_{l-1}}^{(n)} \longleftarrow \text{solver}(w^{(n,l)})$
      $Y_{H_l}^{(n)} = \mathrm{Q}_{H_l}^{(n)} - \mathrm{Q}_{H_{l-1}}^{(n)}$
      $S_{l,p}^{n} \longleftarrow S_{l,p}^{n-1}, Y_{H_l}^{(n)}, n, p,\ p \in [1, P]$
    **end for**
  **end for**
  $h_{l,p} \longleftarrow S_{l,p}, N_l,\ l \in [0, L],\ p \in [1, P]$
  convergence $\longleftarrow$ equation (4.9)
  $it = it + 1$
**end while**
```

To compare the two strategies of algorithms 5 and 6, we solve first the compressible potential flow problem of section 5.3.1, and then the simplified CFD problem of section 5.3.2. To compare the two methods, we evaluate the time to solution of computing $\mathrm{Q}_{H_l}^{(n)} \longleftarrow \text{solver}(w^{(n,l)})$ of algorithm 5, and the time to solution of computing $H_l \xleftarrow{\text{AMR}} \epsilon_l$ and $\mathrm{Q}_{H_l}^{(n)} \longleftarrow \text{solver}(w^{(n,l)})$ of algorithm 6.

As we see in tables 6.1 and 6.2, the relative difference between the two times to solution decreases. This implies that, as the computational cost of the problem grows (thus $\mathrm{Q}_{H_l}^{(n)} \longleftarrow \text{solver}(w^{(n,l)})$ computational cost grows), the relative wall clock time difference between the two methods decreases. It seems therefore more promising to use the stochastic adaptive refinement MLMC method to solve complex problems, since for each

Algorithm 6 Stochastic adaptive refinement MLMC

```
$L, N, \epsilon$ initial hierarchy and interpolation errors
while convergence is not True do
    if non-adaptive then
        $L \longleftarrow L, it$
        $N \longleftarrow N, it$
    else if adaptive then
        $L \longleftarrow L, it, h_{l,p},\ l \in [0, L],\ p \in [1, P]$
        $N \longleftarrow N, it, h_{l,p},\ l \in [0, L],\ p \in [1, P]$
    end if
    for $l = 0 : L$ do
        for $n = 0 : N_l$ do
            $H_l \xleftarrow{\text{AMR}} \epsilon_l$
            $H_{l-1} \xleftarrow{\text{AMR}} \epsilon_{l-1}$
            $\mathrm{Q}_{H_l}^{(n)} \longleftarrow \text{solver}(w^{(n,l)})$
            $\mathrm{Q}_{H_{l-1}}^{(n)} \longleftarrow \text{solver}(w^{(n,l)})$
            $Y_{H_l}^{(n)} = \mathrm{Q}_{H_l}^{(n)} - \mathrm{Q}_{H_{l-1}}^{(n)}$
            $S_{l,p}^{n} \longleftarrow S_{l,p}^{n-1}, Y_{H_l}^{(n)}, n, p,\ p \in [1, P]$
        end for
    end for
    $h_{l,p} \longleftarrow S_{l,p}, N_l,\ l \in [0, L],\ p \in [1, P]$
    convergence $\longleftarrow$ equation (4.9)
    $it = it + 1$
end while
```

	time to solution [s]
$\mathrm{Q}_{H_l}^{(n)} \longleftarrow \text{solver}(w^{(n,l)})$	231
$H_l \xleftarrow{\text{AMR}} \epsilon_l$ and $\mathrm{Q}_{H_l}^{(n)} \longleftarrow \text{solver}(w^{(n,l)})$	353

Table 6.1: Time to solution required to solve $\mathrm{Q}_{H_l}^{(n)} \longleftarrow \text{solver}(w^{(n,l)})$ and the joint of $H_l \xleftarrow{AMR} \epsilon_l$ and $\mathrm{Q}_{H_l}^{(n)} \longleftarrow \text{solver}(w^{(n,l)})$. The former is representative of the deterministic adaptive refinement MLMC method, while the second of the stochastic adaptive refinement MLMC method. The problem solved is the compressible potential flow problem of section 5.3.1.

realization the discretization depends on the current random variables of the system and we can obtain higher accuracy for approximately the same computational cost.

	time to solution [s]
$\mathrm{Q}_{H_l}^{(n)} \longleftarrow \text{solver}(w^{(n,l)})$	4987
$H_l \xleftarrow{\text{AMR}} \epsilon_l$ and $\mathrm{Q}_{H_l}^{(n)} \longleftarrow \text{solver}(w^{(n,l)})$	6246

Table 6.2: Time to solution required to solve $\mathrm{Q}_{H_l}^{(n)} \longleftarrow \text{solver}(w^{(n,l)})$ and the joint of $H_l \xleftarrow{AMR} \epsilon_l$ and $\mathrm{Q}_{H_l}^{(n)} \longleftarrow \text{solver}(w^{(n,l)})$. The former is representative of the deterministic adaptive refinement MLMC method, while the second of the stochastic adaptive refinement MLMC method. The problem solved is the simplified CFD flow problem of section 5.3.2.

6.4 Summary

In this chapter, we have first introduced the Hessian-based metric that we use for performing AMR. Then, we have discussed the possibility of using different error estimators, such as time-dependent fields or their time-averaged counterparts, to compute the metric. Finally, we have proposed two possibilities for integrating MLMC and AMR. Both make use of persistent usage to optimize the runtime, and comparisons between the two methods are shown and discussed. It results that the stochastic adaptive refinement MLMC method is more promising for complex problems, since it provides higher discretization accuracy for approximately the same computational cost of the deterministic adaptive refinement MLMC method.

Chapter 7

Hierarchical Monte Carlo methods for chaotic systems

In this chapter, we solve two stochastic and chaotic fluid flow problems. The former is presented in section 7.1 and describes the two-dimensional flow of air around a rectangle obstacle. The second problem is presented in section 7.2 and solves the flow of air around the Commonwealth Advisory Aeronautical Council (CAARC) building, that is a standard benchmark building in wind engineering. Both problems are stochastic, in the sense that they present random boundary conditions. By solving the first problem, it is found in section 7.1.3 that the hypotheses for an optimal MLMC run (see section 4.3 or the references [58, complexity theorem 3.1] and [111, section 2.2]) are not satisfied. As commented in our works [14, 15], the reason is the chaotic nature of the flow. Thus, the asynchronous single-level MC method is used for solving the two stochastic problems.

For solving the CFD problems, we use Kratos [44, 43] as FE solver software, XMC [13] as hierarchical MC library and PyCOMPSs [16, 88, 126] as programming model for distributed computing. The integration of these software has been an important part of this thesis work, as documented in [127, 10, 11, 5, 4, 22].

The analyses were run on MareNostrum 4. This supercomputer has 11.15 Petaflops of peak performance, which consists of 3456 compute nodes equipped with two Intel R Xeon Platinum 8160 (24 cores at 2.1 GHz each) processors.

The content of this chapter is taken from the preprint of our conference proceeding [131] and from our ExaQUte report [22, section 3] and is adapted wherever needed.

7.1 Rectangle obstacle problem

We start describing the physical system and illustrating the problem uncertainties in section 7.1.1. Then, we validate the domain discretization and the solver in section 7.1.2 and we try to verify the MLMC hypotheses in section 7.1.3. Finally, we solve the problem using the asynchronous MC method in section 7.1.4.

We remark that the rectangle obstacle problem is analyzed and solved with different configurations. In section 7.1, constant in time stochastic boundary conditions are considered, and the problem is solved with hierarchical MC methods. In section 8.2.1, constant in time deterministic boundary conditions are considered, and the problem is solved with the standard time averaging method and the ensemble averaging method. In section 9.2.1, constant in time stochastic boundary conditions are considered, and the problem is solved with the ensemble-based MC method. Therefore, in this section we use constant in time stochastic boundary conditions.

7.1.1 Problem formulation

The problem consists of describing the flow of air around a $5\,\mathrm{m} \times 1\,\mathrm{m}$ rectangle obstacle [28]. The system is described by the incompressible Navier-Stokes equations, introduced in chapter 2 and reported here for the sake of simplicity:

$$\begin{aligned} \frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} - \nu \Delta \boldsymbol{u} + \nabla p = \boldsymbol{f} \qquad & \text{in } [0,T] \times D \\ \nabla \cdot \boldsymbol{u} = 0 \qquad & \text{in } [0,T] \times D, \end{aligned} \tag{7.1}$$

where $\boldsymbol{u}$ is the velocity field, p the pressure field, ν the kinematic viscosity and $\boldsymbol{f}$ the vector field of body forces. D refers to the problem domain, and $[0,T]$ is the time window. The problem domain is shown in figure 7.1. The inlet velocity is uniformly distributed on the y-axis, and has an average value of $2\,\mathrm{m\,s^{-1}}$. We assume the wind inlet velocity magnitude is represented by a normal distribution,

$$\boldsymbol{u}_{inlet} \sim \mathcal{N}(2.0, 0.02). \tag{7.2}$$

More consistent wind models are used in the next example in section 7.2 and in chapters 8 and 9. Slip boundary conditions are applied on the external boundaries, and no-slip boundary conditions are enforced on the rectangle body. The Reynolds number of the

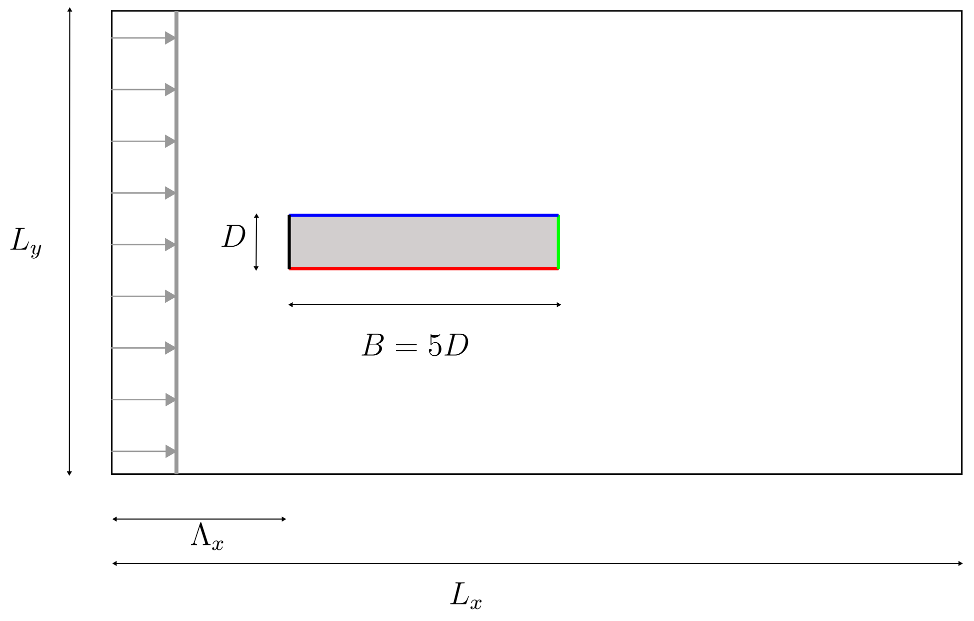


Figure 7.1: Scheme of the computational domain used for the rectangle problem, where $D = 1\,\mathrm{m}$, $B = 5D$, $L_x = 55B$, $L_y = 30B$ and $\Lambda_x = 15B$. Thus, the dimensions of the outer domain are $275 \times 150\,\mathrm{m}$, and the inner rectangle has size $5 \times 1\,\mathrm{m}$.

problem is 132719. We remark that, even though turbulence cannot be identified in two-dimensional systems, the rectangle obstacle problem we define is particularly important, since it provides a cheap yet accurate system for studying features of chaotic flows.

The *washout time*, which is the time needed for one particle to go from the inlet to the outlet of the domain, is 137.5 s, considering an average speed of $2\,\mathrm{m\,s^{-1}}$. The washout time is computed as

$$T_w = \frac{275\,\mathrm{m}}{2\,\mathrm{m\,s^{-1}}}. \tag{7.3}$$

Therefore, we consider a *burn-in time* T_0 of 140 s. T_0 is the simulation time information we discard to remove the influence of initial conditions on the problem solutions. More insights about optimal burn-in time estimations are given in chapter 8.

The QoIs we are interested in are:

1. time-averaged drag force $\langle F_d \rangle_{T_0,T}$,
2. time-averaged pitching moment $\langle M_p \rangle_{T_0,T}$,
3. time-averaged pressure field $\langle p(\boldsymbol{x}) \rangle_{T_0,T}$ on the body surface,
4. drag force F_d,

5. pitching moment M_p,

6. pressure field $p(\boldsymbol{x})$ on the body surface.

By *time-averaged*, we refer to quantities averaged over the interval $[T_0, T]$, thus some information is lost due to the averaging process, which damps peaks and oscillations. In other words, out of a single realization, the quantity of interest is a single scalar value. On the other hand, other quantities keep all the historical information. International units are used to measure quantities and we omit the subscript T_0, T if there is no risk of misunderstanding.

7.1.2 Validation

The mesh considered to solve the problem is a solution-oriented AMR discretization. The mesh is refined with respect to the time-averaged velocity field $\langle \boldsymbol{u}(t, \boldsymbol{x}) \rangle_{T_0,T}$, averaged on $[T_0, T]$, where $[0, T_0]$ is the burn-in time discarded to avoid dependencies on initial conditions. The metric computed to perform the refinement is built within Kratos [94], exploiting the averaged velocity field, and the original mesh is refined using the Mmg software [45].

The final mesh has around 25000 nodes, and a minimal size, close to the rectangle body, of 0.002 m. The chosen time step is 0.02 s, which gives a Courant-Friedrichs-Lewy (CFL) number of

$$\mathrm{CFL} = \frac{\Delta t\, \boldsymbol{u}}{h} \approx 20. \tag{7.4}$$

From the drag force of table 7.2, one can estimate the drag coefficient as

$$C_d = \frac{F_d}{\frac{1}{2}\rho \boldsymbol{u}^2 A}, \tag{7.5}$$

where ρ is the fluid density, $\boldsymbol{u}$ the speed of the rectangle body relative to the fluid and A the cross sectional area. The drag coefficient we obtain is $C_d = 1.320$, which is consistent with literature results [28].

7.1.3 MLMC validation

Our first attempt to solve the stochastic problem is using MLMC, which provides a better convergence rate with respect to standard MC. To use MLMC, the three hypotheses defined in section 4.3 must be satisfied. The set of meshes we consider are presented in table 7.1. These meshes are generated performing AMR using a reference inlet wind velocity of $2\,\mathrm{m\,s^{-1}}$. The error indicator used for performing AMR is the joint of velocity and pressure fields.

ϵ	h_{min}	N [$\cdot 1000$]
$1.5 \cdot 10^{2}$	0.054	0.9
$1.5 \cdot 10^{1}$	0.017	1.9
$1.5 \cdot 10^{0}$	0.0054	5.4
$1.5 \cdot 10^{-1}$	0.0019	25.6
$1.5 \cdot 10^{-2}$	0.00068	229.4

Table 7.1: List of meshes computed with AMR strategy. ϵ is the interpolation error, that is an indicator of the error that is committed by discretizing the domain with respect to an error indicator. N is the number of thousands of nodes of the mesh and h_{min} the minimal size.

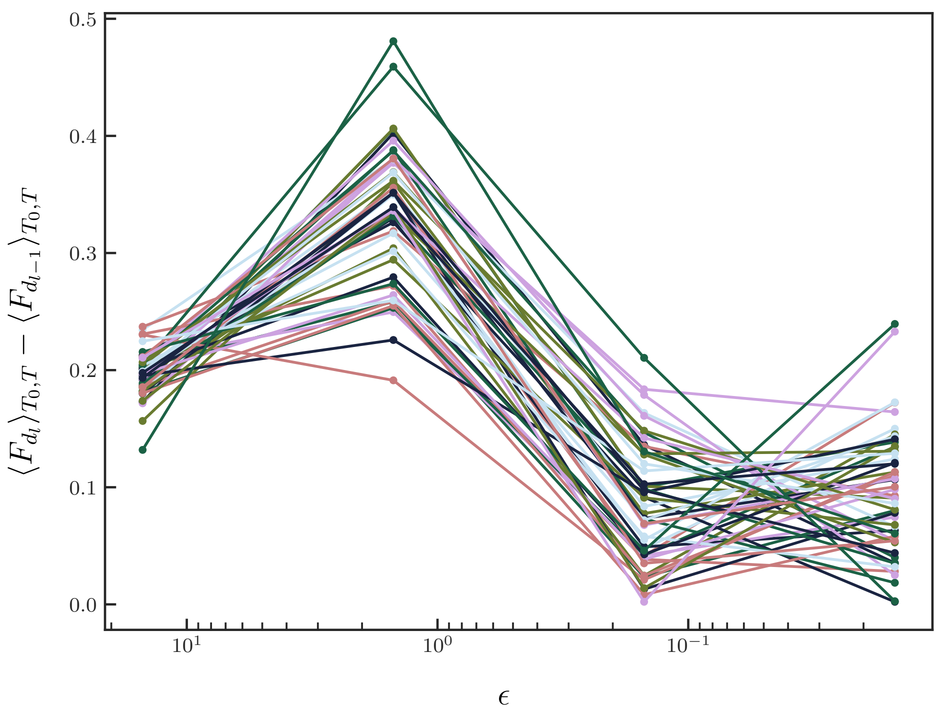


Figure 7.2: Difference of the drag force between consecutive levels, plotted as function of the interpolation error. 50 realizations are reported, each with wind inlet randomly generated.

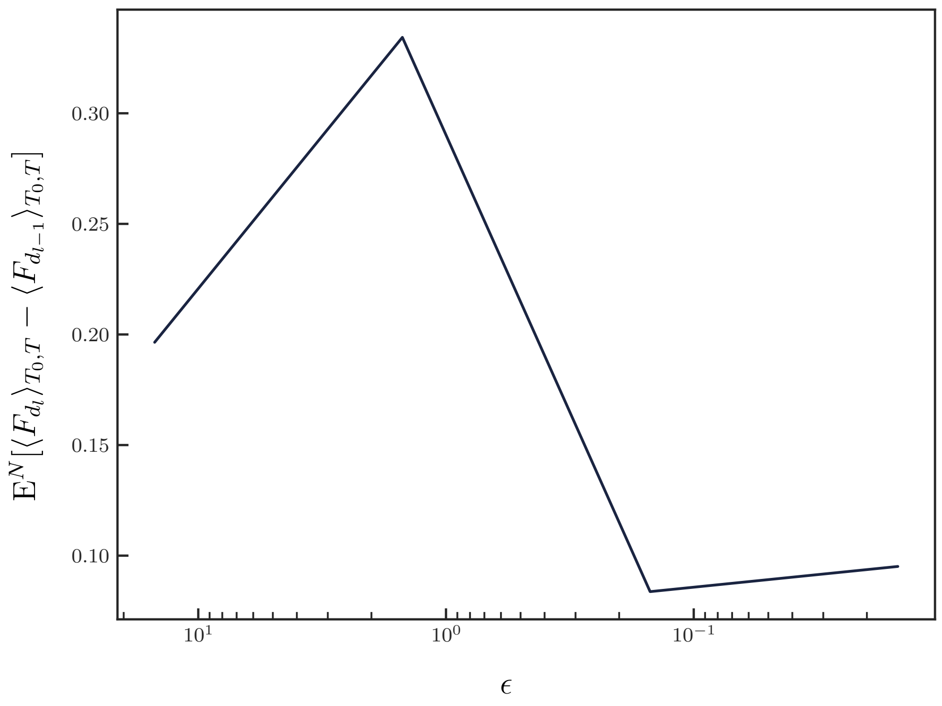


Figure 7.3: Expected value of the difference of the drag force, computed on consecutive levels. The expected value is computed from the 50 realizations of figure 7.2, and is plotted as function of the interpolation error.

First, we try to check if the expected value of the difference of F_d across consecutive levels decreases exponentially as the mesh accuracy grows. We generate 50 different random inlet conditions w, and we observe the behavior of $\mathrm{E}^{\mathrm{MC}}[\langle \mathrm{Q}_l\rangle_{T_0,T} - \langle \mathrm{Q}_{l-1}\rangle_{T_0,T}]$, where Q_l is the drag force F_d computed on level l. We report in figure 7.2 the difference between consecutive levels of the drag force, for each wind scenario. We can observe that there is no correlation. In figure 7.3, we plot the expected value of the difference of the drag force, from the 50 scenarios of figure 7.2. We can observe that there is no exponential decay.

Figure 7.4 plots the variance of the difference of the drag force across consecutive levels, as function of the interpolation error. We observe that there is no exponential decay.

Therefore, MLMC hypotheses are not satisfied by the current problem and we cannot apply standard MLMC to solve it. This conclusion is in agreement with [14, 15], which comment on the difficulties of finding time signal correlations between different discretization meshes for chaotic flows. Specifically, in [15, section 2] the authors show that time signal correlations can be achieved by reducing the time horizon as

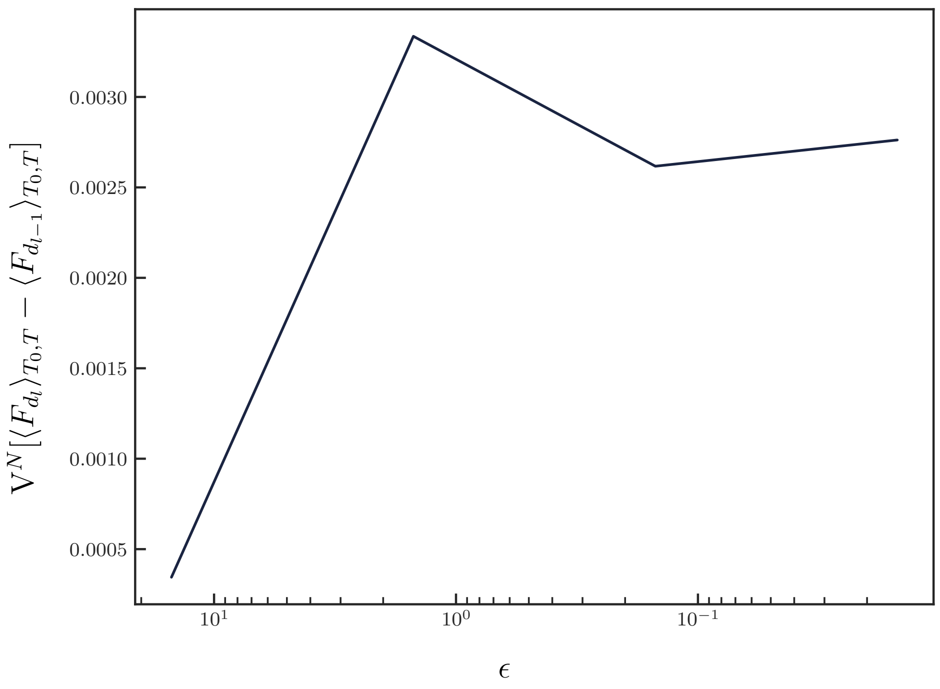


Figure 7.4: Variance of the difference of the drag force computed on consecutive levels. The variance is plotted against the interpolation error. The 50 scenarios of figure 7.2 are considered.

$T < \mathcal{O}(\mathcal{L}^{-1})$. $\mathcal{L}$ is a constant related to how quickly two very close paths diverge, and it is large and positive for chaotic systems, thus implying T should be very small to satisfy MLMC hypotheses for chaotic systems. Since using a very small T is not possible for the class of problems we aim at solving, the only possibility is making $\mathcal{L}$ small enough. In [122, 54], the authors achieve reducing such a constant in the case of stochastic differential equations. However, to the best of the authors' knowledge, no methods have been proposed to reduce such a constant $\mathcal{L}$ for chaotic NS flows yet.

Therefore, due to the impossibility of using the MLMC method, we apply the asynchronous MC method for solving the stochastic problem.

7.1.4 Results

For solving the stochastic problem, we want to find the time window $[0, T]$ and the number of realizations N which satisfy equation (4.9), with confidence $1 - \phi = 0.99$ and tolerance $\varepsilon = 0.0085$. Even though multiple quantities of interest are computed, convergence is checked only for the time-averaged drag force. We remark also that the tolerance is absolute, and it has a relative value of around 0.2%, with respect to the time-averaged drag force expected value estimation.

The chosen algorithm is asynchronous MC, therefore we exploit equation (4.14) for assessing convergence. We remark that the number of computing nodes of the supercomputer is chosen accordingly to the initial MC hierarchy, to make sure a single batch of the asynchronous MC method can properly fill the HPC machine. Given a time window $[0, 300]$ seconds, it is found that we satisfy equation (4.9) after running 960 realizations. The overall computational cost for solving the problem is approximately 35000 CPU hours, and is computed as the product between the total number of simulation hours multiplied by the number of cores exploited.

We report in figure 7.5 and figure 7.6 the instantaneous velocity and pressure fields at $t = 200\,\text{s}$ for one realization.

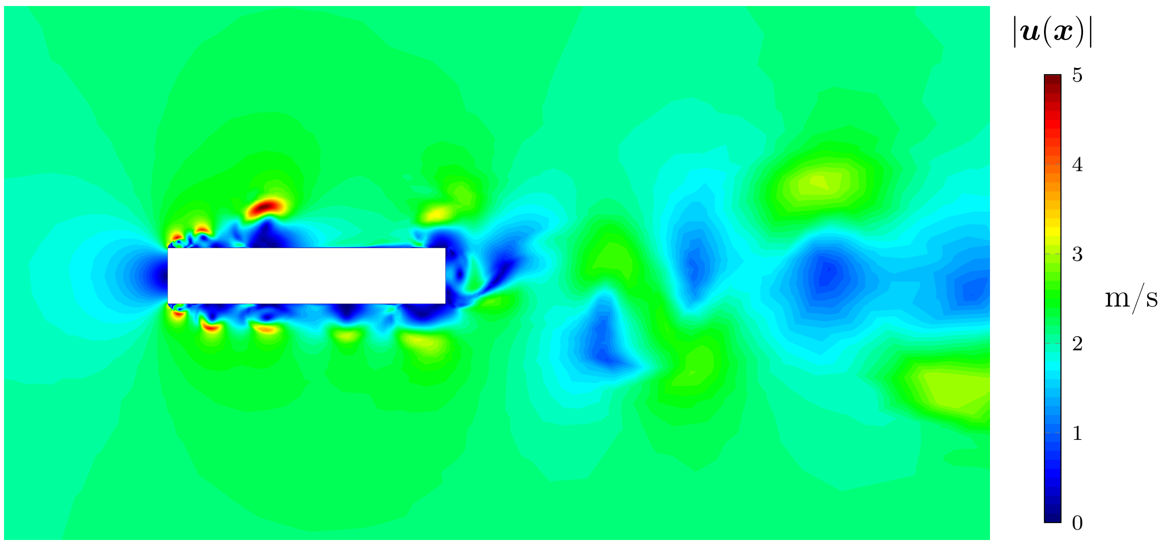


Figure 7.5: Velocity field snapshot at $t = 200\,\text{s}$.

We report in table 7.2 the expected value and standard deviation estimations of the quantities of interest. As we can readily observe, the mean values are the same, which is the expected behavior. On the other hand, we see that standard deviations values are different, much smaller for time-averaged quantities of interest. This happens because oscillations are damped due to the intermediate averaging process.

The different behavior between scalar quantities and their time-averaged counterparts can be observed also looking at the pressure field distribution around the rectangle body,

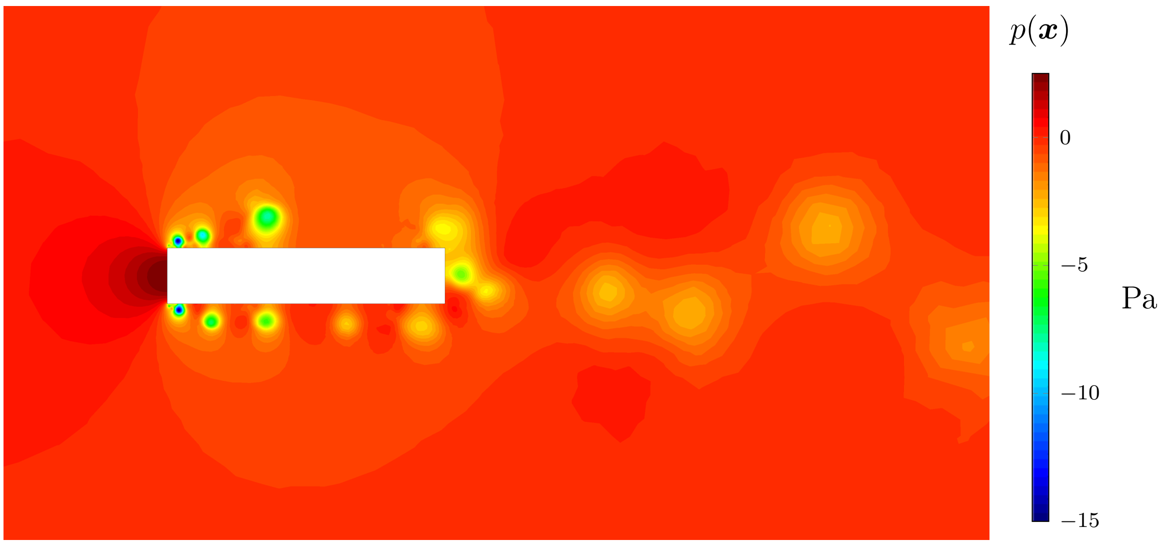


Figure 7.6: Pressure field snapshot at $t = 200\,\mathrm{s}$.

Q	$\mathrm{E}^N[\mathrm{Q}]$	$\sigma^N[\mathrm{Q}]$
$\langle F_d \rangle_{T_0,T}$	3.23506	0.01242
$\langle M_p \rangle_{T_0,T}$	-0.01055	0.01449
F_d	3.23506	0.33241
M_p	-0.01055	4.55940

Table 7.2: Statistical analysis of time-averaged drag force $\langle F_d \rangle_{T_0,T}$, time-averaged pitching moment $\langle M_p \rangle_{T_0,T}$, drag force F_d and pitching moment M_p. Results for $N = 960$ and $T = 300\,\mathrm{s}$ are provided.

which is plotted in figure 7.7. We refer to figure 7.1 as a reference for understanding the plots of figure 7.7.

Q	CVaR	α
$\langle F_d \rangle_{T_0,T}$	3.26569	0.9
F_d	4.33234	0.9

Table 7.3: CVaR analysis of the time-averaged drag force $\langle F_d \rangle_{T_0,T}$ and the drag force F_d. Results for $N = 960$, $T = 300\,\mathrm{s}$ and $\alpha = 0.9$ are provided.

Finally, another statistical quantity one may be interested in computing is the CVaR, defined in chapter 3. Looking at table 7.3, we can observe the CVaR values for the drag force and its time-average counterpart, with $\alpha = 0.9$. Similar to what we com-

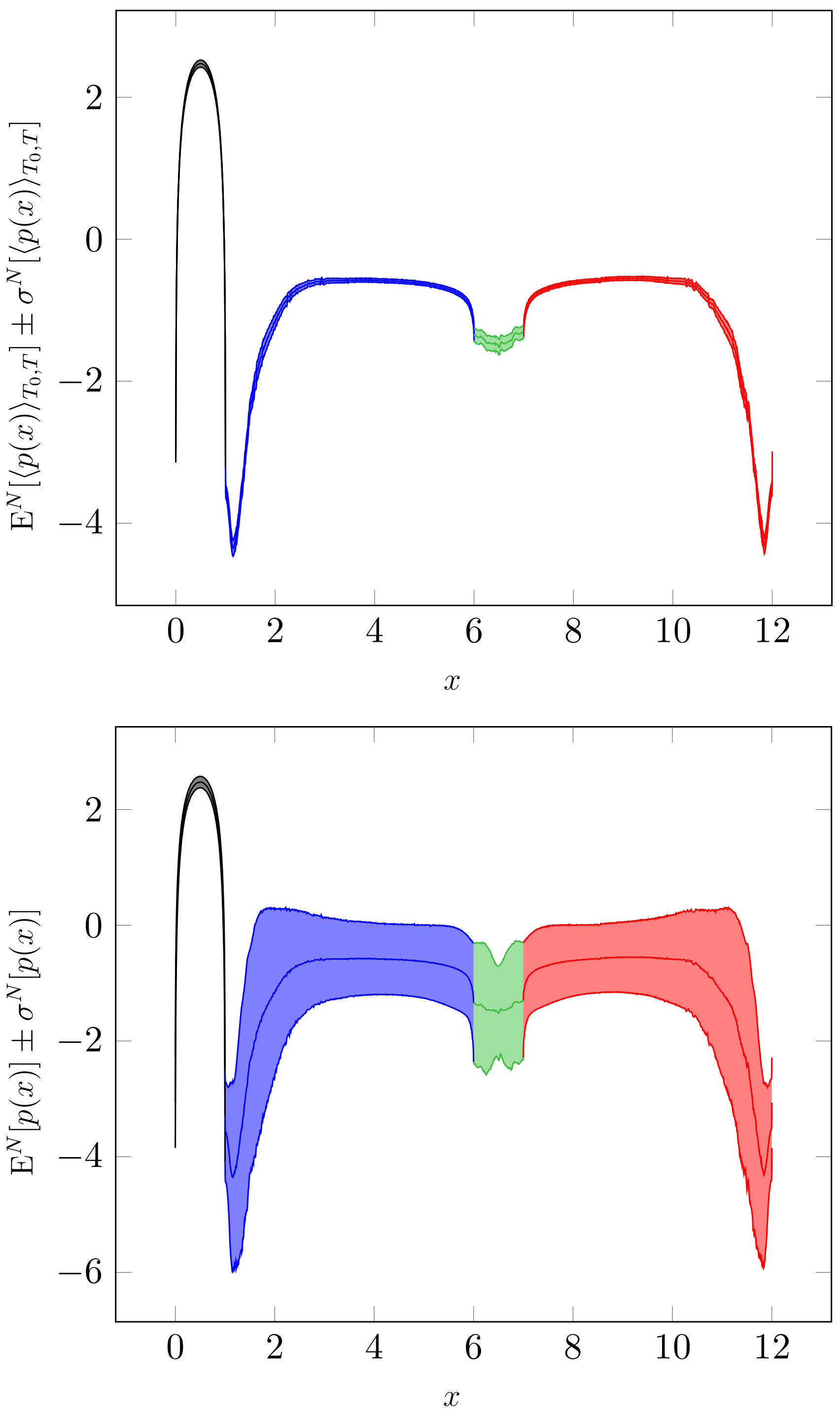


Figure 7.7: The upper figure shows the risk measure expected value $\pm$ standard deviation of the time-averaged pressure field ($\mathrm{Q} = \langle p(\boldsymbol{x})\rangle_{T_0,T}$), and the lower figure the same risk measure of the pressure field ($\mathrm{Q} = p(\boldsymbol{x})$).

mented before, we can observe that time-averaged values are smaller since oscillations are damped by the intermediate averaging process.

7.2 High-rise building problem

The second problem considers the flow of wind around the CAARC building [24, 67, 66, 107]. We start presenting the system in section 7.2.1, we validate the mesh discretiztiona and the solver in section 7.2.2 and we present results in section 7.2.3.

We remark that the CAARC building problem is analyzed and solved with different configurations. In section 7.2, turbulent fluctuations around a fixed mean wind field boundary conditions are considered, and the problem is solved with hierarchical MC methods. In section 8.2.2, constant in time fixed mean wind field boundary conditions are considered, and the problem is solved with the standard time averaging method and the ensemble averaging method. In section 9.2.2, constant in time stochastic mean wind field boundary conditions are considered, and the problem is solved with the ensemble-based MC method. In section 9.2.3, turbulent fluctuations around a stochastic mean wind field boundary conditions are considered, and the problem is solved with the ensemble-based MC method. Therefore, in this section we use turbulent fluctuations around a fixed mean wind field boundary conditions.

7.2.1 Problem formulation

The flow is modeled with the incompressible NS equations (see chapter 2 or section 7.1). The problem domain is presented in figure 7.8. The CAARC is a parallelepiped building with width 45 m, length 30 m and height 180 m. The domain is 1800 m long, 864 m large and 576 m high. Slip boundary conditions are applied on the walls and the ceiling, no-slip boundary conditions on the building and on the floor.

$\bar{\boldsymbol{u}}_H$	H	z_0	ρ	μ	Re
$40\,\mathrm{m\,s^{-1}}$	$180\,\mathrm{m}$	$2\,\mathrm{m}$	$1.225\,\mathrm{kg/m^3}$	$1.846 \cdot 10^{-5}\,\mathrm{kg\,m^{-1}\,s^{-1}}$	119447453

Table 7.4: Physical parameters problem.

We consider a scenario typical of centers of very large cities, and we report in table 7.4 the physical properties of the problem. Therefore, the roughness height z_0 is set to

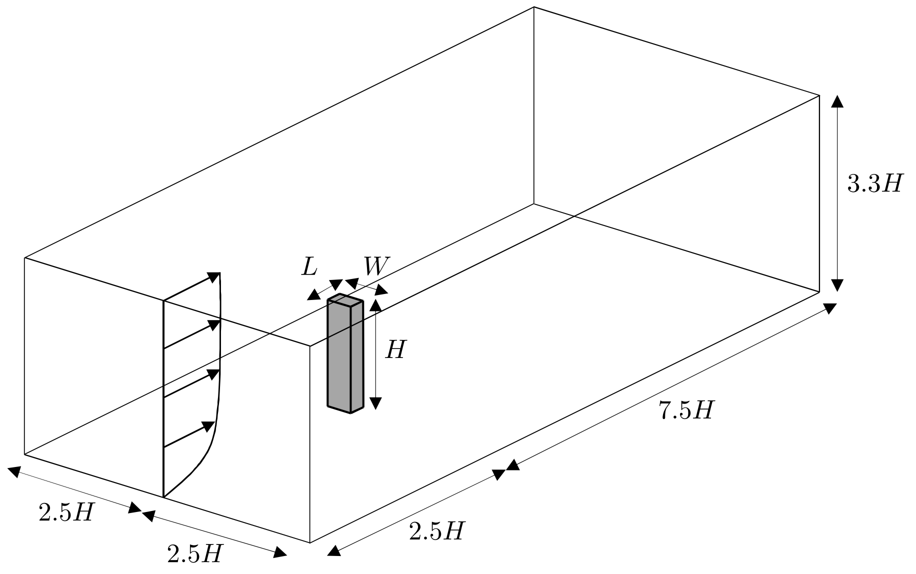


Figure 7.8: CAARC problem domain. $H = 180\,\mathrm{m}$, $W = 45\,\mathrm{m}$ and $L = 30\,\mathrm{m}$. Wind gusts are omitted for the sake of simplicity, and only the mean wind profile is represented.

2.0 m [73], which implies the wind velocity mean profile $\overline{\boldsymbol{u}}$ to be fixed. The wind velocity is modeled as described in section 2.4. The mean profile $\overline{\boldsymbol{u}}$ follows the logarithmic profile

$$\bar{\boldsymbol{u}}(z) = \frac{\boldsymbol{u}^*}{k} \ln(z/z_0), \tag{7.6}$$

where $k \approx 0.4$ is von Karman's constant and $\boldsymbol{u}^*$ the friction velocity. We fix a reference velocity of $40\,\mathrm{m\,s^{-1}}$ at the reference height of 180 m of the building. At the same time, wind fluctuations $\boldsymbol{u}'$ are considered and are the uncertain parameter of the system. Wind gusts are modeled with the Mann model, as commented in section 2.4. The Reynolds number is of the order of 10^8, computed with a characteristic length of 45 m and density and viscosity of air.

Since in section 7.1.3 and in [14, 15] it is shown that satisfying MLMC hypotheses is challenging for chaotic flows, we decide to solve such a stochastic problem applying the asynchronous MC method. Convergence of the asynchronous MC algorithm is checked using the failure probability criterion, defined in equation (4.9). Since we use a single-level method for solving the stochastic problem, convergence is assessed by computing equation (4.14).

The QoI for which we assess convergence is the time-averaged drag force $\langle F_d \rangle_{T_0,T}$, averaged over the effective time window $[T_0, T]$, where we remark that the burn-in time T_0 is the time history we discard to neglect the effect of initial conditions on the QoI, and we set it to 32 s. The confidence $1-\phi$ is 99% and the tolerance ε is 65000, whose corresponding relative tolerance with respect to the time-averaged drag force expected value is approximately 0.7%. Other observables are computed: the drag force F_d, the base moment M_b and the time-averaged base moment $\langle M_b \rangle_{T_0,T}$, the pressure field on the building $p(\boldsymbol{x})$ and the time-averaged pressure field on the building $\langle p(\boldsymbol{x}) \rangle_{T_0,T}$. International units are used to measure quantities and we omit the subscript T_0, T if there is no risk of misunderstanding.

7.2.2 Validation

The mesh we use is adaptively refined with respect to pressure and velocity fields. It presents around 283 thousand nodes and a minimal size of 0.2 m close to the building. The chosen time step is 0.2375 s, which gives a CFL of 100. We shall remark that the mesh remains relatively coarse with respect to the resolution that would be needed to resolve the flow at the Reynolds number of interest. This situation is often encountered in the field of wind engineering [124]. The VMS stabilization, which is used as basis of our solver, represents an alternative to classical LES approaches and provides a simple yet effective turbulence model for the applications of interest [39, 113, 42].

To ensure correctness of the solver and of the mesh, we compute the following normalized coefficients [24]

$$\begin{aligned} C_{F_X} &= \frac{F_X}{1/2 \rho W \int_0^H \bar{\boldsymbol{u}}^2 \, \mathrm{d}Z} \quad C_{F_Y} = \frac{F_Y}{1/2 \rho W \int_0^H \bar{\boldsymbol{u}}^2 \, \mathrm{d}Z} \\ C_{M_X} &= \frac{M_X}{1/2 \rho \bar{\boldsymbol{u}}_H^2 W H^2} \quad C_{M_Y} = \frac{M_Y}{1/2 \rho \bar{\boldsymbol{u}}_H^2 W H^2}, \end{aligned} \tag{7.7}$$

where ρ is the density of the fluid, $\bar{\boldsymbol{u}}$ the velocity of the mean profile, $\bar{\boldsymbol{u}}_H$ the velocity at height $H = 180$ m, W the building width and F_X, F_Y, M_X and M_Y the forces and moments computed on directions X and Y, respectively. X is the direction parallel to the ground and orthogonal to the inlet, while Y is parallel to both the ground and the inlet. We compare the time-averaged normalized coefficients with [107, 67, 24] in table 7.5.

References	$\langle C_{F_X} \rangle$	$\langle C_{F_Y} \rangle$	$\langle C_{M_X} \rangle$	$\langle C_{M_Y} \rangle$
Current work	1.818	0.001	-0.001	0.694
[107]	1.490	-0.039	0.000	0.640
[67]	1.830	0.006	-	-
[24]	1.660	0.008	0.004	0.570

Table 7.5: Normalized force and moment coefficients mean values of current work, compared with literature. Time averages of our work are estimated for an effective time window $T - T_0 = 600\,\text{s}$.

7.2.3 Results

The convergence criterion is satisfied by a number of realizations $N = 126$ and an effective time window $T - T_0 = 300\,\text{s}$. The final error $\mathcal{C}_\phi\text{SE}$ we obtain is 52755, whose relative value $\frac{\mathcal{C}_\phi\text{SE}}{\text{E}^{\text{MC}}[\langle F_d \rangle_{T_0,T}]}$ is smaller than 0.6%.

We report in figure 7.9 and figure 7.10 the instantaneous velocity and pressure fields at $t = 50\,\text{s}$ for one realization.

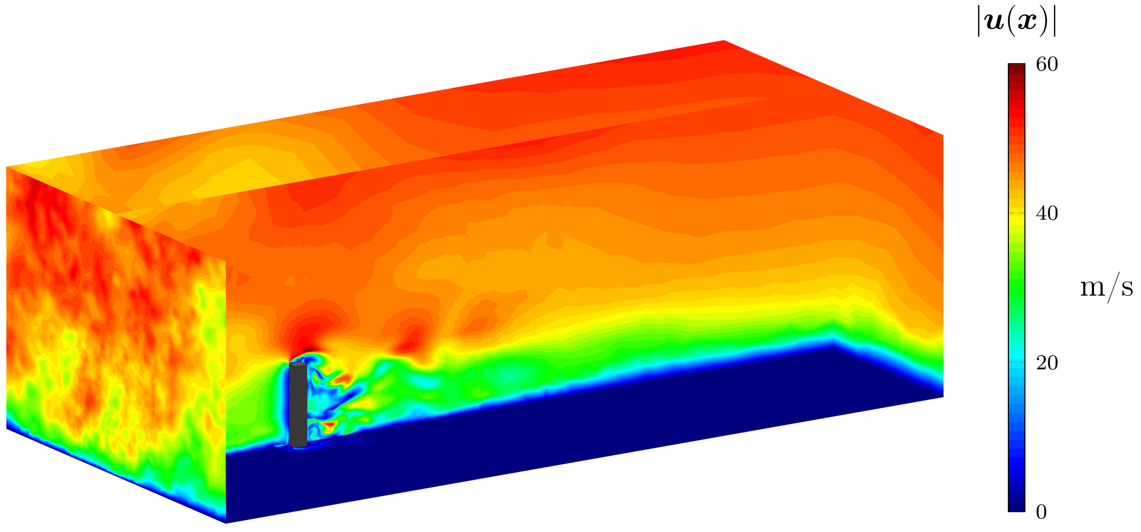


Figure 7.9: Velocity field snapshot at $t = 50\,\text{s}$.

In table 7.6 we report the sample expected value and the sample standard deviation of $\langle F_d \rangle_{T_0,T}$, $\langle M_b \rangle_{T_0,T}$, F_d and M_b. We can observe that, as expected, time-averaged quantities present the same expected value as their corresponding standard values. On the other hand, standard deviation values are smaller for time-averaged quantities since

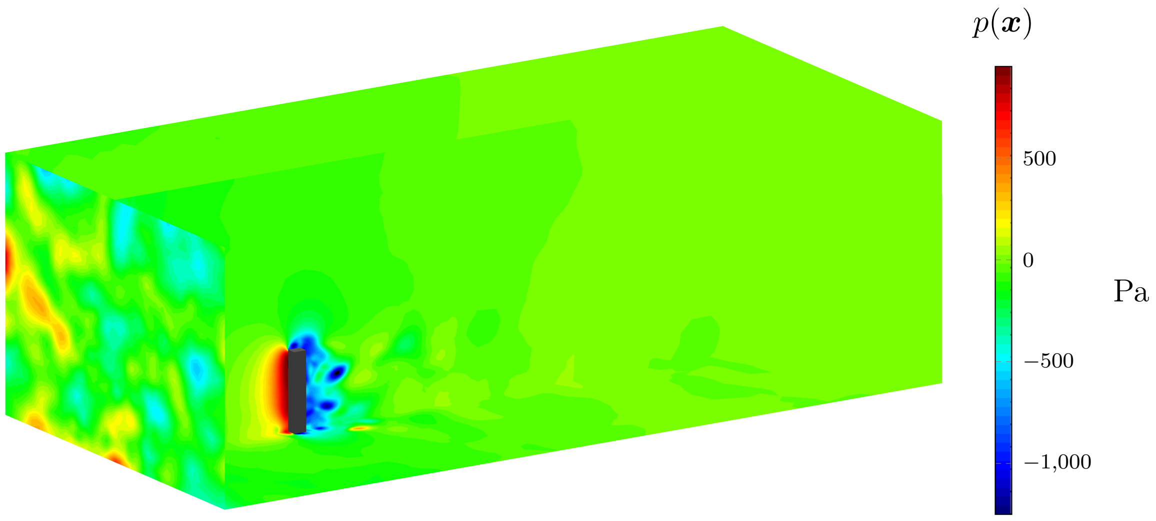


Figure 7.10: Pressure field snapshot at $t = 50\,\mathrm{s}$.

the intermediate time-averaging process damps oscillations and peaks.

Q	$\mathrm{E}^N[\mathrm{Q}]$	$\sigma^N[\mathrm{Q}]$
$\langle F_d \rangle_{T_0,T}$	9417766	254555
$\langle M_b \rangle_{T_0,T}$	-36001	720874
F_d	9417766	2365324
M_b	-36001	9434439

Table 7.6: Expected value and standard deviation of time-averaged drag force $\langle F_d \rangle_{T_0,T}$, time-averaged base moment $\langle M_b \rangle_{T_0,T}$, drag force F_d and base moment M_b. Results for $N = 126$ and $T - T_0 = 300\,\mathrm{s}$ are provided.

Similar conclusions can be drawn for the time-averaged pressure field $\langle p(\boldsymbol{x}) \rangle_{T_0,T}$ and the pressure field $p(\boldsymbol{x})$. Observing figures 7.11 and 7.12, we can readily observe that the mean values are the same, while standard deviations are not.

It is known that a risk measure as $\mathbb{E}[\mathrm{Q}] \pm \sigma[\mathrm{Q}]$ is not optimal for many physical variables, especially when Q presents a non-symmetric PDF [115]. For this reason, it is interesting to estimate other statistical estimators as the VaR or the CVaR. We report in table 7.7 the CVaR results for the time-averaged drag force and the drag force. Once more, we observe the difference between standard physical quantities and their time-averaged counterparts.

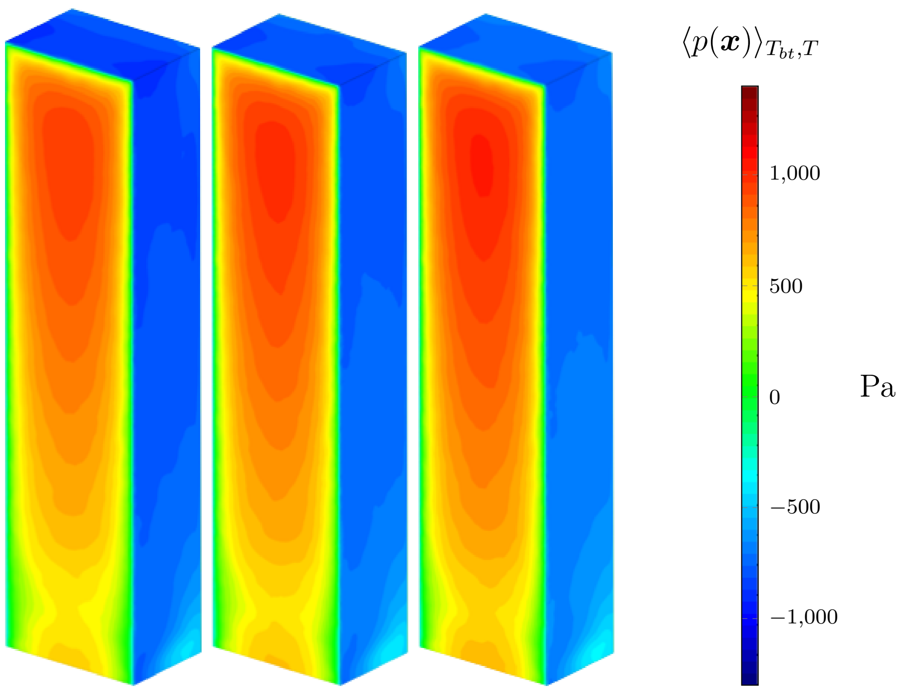


Figure 7.11: Statistical result of the time-averaged pressure field $\langle p(\boldsymbol{x})\rangle_{T_0,T}$. From left to right, $\mathbb{E}[\langle p(\boldsymbol{x})\rangle_{T_0,T}] - \sigma[\langle p(\boldsymbol{x})\rangle_{T_0,T}]$, $\mathbb{E}[\langle p(\boldsymbol{x})\rangle_{T_0,T}]$ and $\mathbb{E}[\langle p(\boldsymbol{x})\rangle_{T_0,T}] + \sigma[\langle p(\boldsymbol{x})\rangle_{T_0,T}]$.

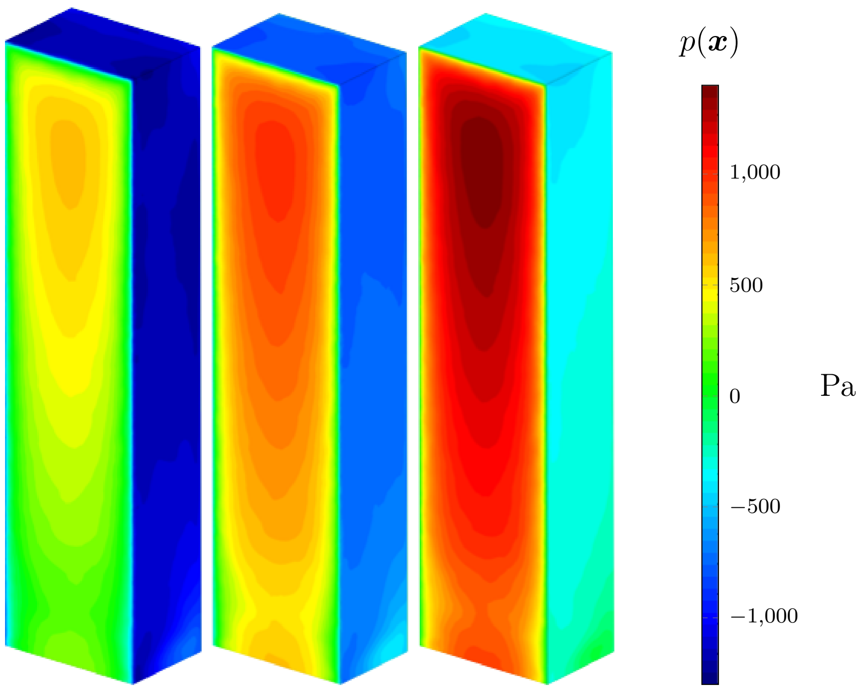


Figure 7.12: Statistical result of the pressure field $p(\boldsymbol{x})$. From left to right, $\mathbb{E}[p(\boldsymbol{x})] - \sigma[p(\boldsymbol{x})]$, $\mathbb{E}[p(\boldsymbol{x})]$ and $\mathbb{E}[p(\boldsymbol{x})] + \sigma[p(\boldsymbol{x})]$.

Q	CVaR	α
$\langle F_d \rangle_{T_0,T}$	9845804	0.9
F_d	13871810	0.9

Table 7.7: CVaR analysis of time-averaged drag force $\langle F_d \rangle_{T_0,T}$ and drag force F_d. Results for $N = 126$, $T - T_0 = 300\,\mathrm{s}$ and $\alpha = 0.9$.

7.3 Summary

We have applied hierarchical MC methods, introduced in chapters 4 and 5, to solve two high Re chaotic systems, which are typical benchmarks in the wind engineering community.

First, we have verified that MLMC hypotheses are not satisfied by the chaotic flows we are interested in, due to the lack of pathwise correlation of the QoI time signals on different mesh discretizations. Therefore, we have applied the asynchronous single-level MC method to solve the stochastic problems.

Solving the problems, multiple statistics of different physical variables have been computed. Specifically, we have estimated the expected value, the standard deviation, and the CVaR. The expected value is not sensitive to high values and the standard deviation takes into account oscillations, meanwhile, the CVaR measures the weight of the tail of the probability density function. Moreover, we have observed that time-averaged quantities damp oscillations and peaks, so they can lose information about wind critical scenarios.

Since the problems we are interested in are ergodic, time-dependent and they normally require large human time to be solved, strategies to reduce the time to solution are explored in chapters 8 and 9.

Chapter 8

Ensemble averaging for chaotic and ergodic systems

In this chapter, we aim at estimating statistics of QoIs by averaging over numerous independent simulations, i.e. statistical ensembles. The upshot of this strategy is that each of the simulations within the ensemble can be launched independently and run in parallel, thus providing an obvious opportunity for acceleration when abundant computational resources are available.

Even though ensemble averaging has been investigated previously, the application we target (wind engineering), as well as the numerical method we employ (implicit LES), are significantly different from previous investigations, thus leaving the applicability of ensemble averaging unclear. The goal of this chapter is therefore to develop a technique to assess the efficacy of ensemble averaging when applied to any given turbulent flow problem. To this end, we present a statistical analysis of the approach in section 8.1.

Another question we seek to address is how long each realization should be. Increasing the number of realizations improves concurrency but also increases the aggregated burn-in time, so there is a trade-off that needs to be considered. The present study evaluates this trade-off in the case of flows around bodies in section 8.2. We demonstrate that very short simulations are sufficient, thus making the approach highly efficient for the class of problems we have targeted.

The content of this chapter is taken from a manuscript in preparation and is adapted wherever needed.

8.1 Statistical analysis

In this section, we introduce our statistical framework. In section 8.1.1 we define the statistical operators and the probability convergence criterion. Different sources of error are identified and analyzed in section 8.1.2. We discuss the generation of initial velocity fields, which provides independent ensemble realizations in section 8.1.3. Finally, considerations on how to determine the length of the burn-in time are provided in section 8.1.4.

8.1.1 Problem outline

Let $u(t, \boldsymbol{x}, w)$ denote the solution to equation (2.1), where the initial condition $u_0(\boldsymbol{x}) = u_0(\boldsymbol{x}; w)$, $w \in \Omega$, is a random field over D. In this setting, $u(t, \boldsymbol{x}, w)$ is a random field over $(0, T) \times D$. From u, we can compute the quantity of interest $\mathrm{Q}(t, w) := \mathrm{Q}(u(t, \boldsymbol{x}, w))$, which is a stochastic process. For ease of notation, we omit the dependency on w in what follows, except when it is needed to clarify the presentation. In agreement with what introduced in chapter 3, the expected value of a process X is denoted $\mathbb{E}[X]$, its variance is denoted $\mathbb{V}[X]$ and its covariance with a process Y is denoted $\mathrm{co}\mathbb{V}[X, Y]$. These quantities can be estimated by sample averaging. For instance, for any N independent samples, $X_1, \ldots, X_N$, we define the sample mean $\mathrm{E}^N[X] = \frac{1}{N}\sum_{n=1}^{N} X_n$ and the sample variance $\mathrm{V}^N[X] = \frac{1}{N-1}\sum_{n=1}^{N}(X_n - \mathrm{E}^N[X])^2$.

Our goal is to compute the long-term expected value of $\mathbb{E}[\mathrm{Q}]$, that is,

$$\overline{\mathrm{Q}} = \lim_{t\to\infty} \mathbb{E}[\mathrm{Q}(t)]. \tag{8.1}$$

Assuming ergodicity, $\overline{\mathrm{Q}}$ can also be computed as the time average, defined as

$$\langle \mathrm{Q} \rangle = \lim_{T\to\infty} \frac{1}{T} \int_0^T Q(t)\,\mathrm{d}t. \tag{8.2}$$

Time averages performed over the effective time $[T_0, T]$ are defined as

$$\langle \mathrm{Q} \rangle_{T_0,T} = \frac{1}{T - T_0} \int_{T_0}^T Q(t)\,\mathrm{d}t, \tag{8.3}$$

and the notation in equation (8.3) simplifies to $\langle \mathrm{Q} \rangle_T$ when $T_0 = 0$, that is $\langle \mathrm{Q} \rangle_T := \langle \mathrm{Q} \rangle_{0,T}$.

Given N realizations of the process Q, $\mathrm{Q}_n := \mathrm{Q}(t, w^{(n)})$, for $n = 1, ..., N$, we aim to approximate $\overline{\mathrm{Q}}$ by

$$\overline{\mathrm{Q}} \approx \mathrm{E}^N[\langle \mathrm{Q} \rangle_{T_0,T}] = \frac{1}{N}\sum_{n=1}^{N} \langle \mathrm{Q}_n \rangle_{T_0,T}. \tag{8.4}$$

Using this notation, the statistical problem we face is to find the optimal values of N, T_0, T (those that minimize the computational cost and/or the time to solution) while satisfying the probability convergence criterion of equation (4.9). We report it here for the sake of simplicity:

$$\mathbb{P}\left[\left|\mathrm{E}^N[\langle \mathrm{Q} \rangle_{T_0,T}] - \overline{\mathrm{Q}}\right| > \varepsilon\right] \leq \phi,\ \varepsilon > 0,\ \phi \ll 1, \tag{8.5}$$

where ε is the desired tolerance and $1 - \phi$ the confidence on the sampled statistical estimator. Such a condition requires that the probability of the error exceeding ε should not be greater than ϕ.

8.1.2 Error analysis

There are two sources of error in equation (8.4). First, the choice of the random initial condition may not be compatible with the long-term statistically stationary, thus triggering a transient perturbation with a nonvanishing contribution to the mean, sometimes called initialization bias [18]. The use of a finite number of samples of finite duration is a second source of error. Increasing the number of samples, as well as the length of their effective time intervals, will also reduce the influence of the initialization bias.

Initialization bias

To analyze the error in approximation (8.4) we decompose each realization Q_n into two components,

$$\mathrm{Q}_n = A_n + S_n, \tag{8.6}$$

where S_n is the realization of an ergodic and stationary process S, and A_n is a transient perturbation. This means to assume $\lim_{t\to\infty} \mathbb{E}[A_n(t)] = 0$ or, equivalently,

$$\lim_{t\to\infty} \mathbb{E}[\mathrm{Q}_n(t)] = \overline{\mathrm{Q}} = \mathbb{E}[S]. \tag{8.7}$$

In this setting,

$$\mathbb{E}[\mathrm{E}^N[\langle \mathrm{Q}\rangle_{T_0,T}]] = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}[\langle \mathrm{Q}_n\rangle_{T_0,T}] = \frac{1}{N}\sum_{n=1}^{N}\left(\mathbb{E}[\langle S_n\rangle_{T_0,T}] + \mathbb{E}[\langle A_n\rangle_{T_0,T}]\right). \tag{8.8}$$

Since S is statistically stationary,

$$\mathbb{E}[\langle S_n\rangle_{T_0,T}] = \mathbb{E}\left[\frac{\int_{T_0}^{T} S_n(t)\,\mathrm{d}t}{T-T_0}\right] = \frac{1}{T-T_0}\int_{T_0}^{T}\mathbb{E}[S_n]\,\mathrm{d}t = \mathbb{E}[S_n] = \mathbb{E}[S] = \overline{\mathrm{Q}}. \tag{8.9}$$

Therefore,

$$\mathbb{E}[\mathrm{E}^N[\langle \mathrm{Q}\rangle_{T_0,T}]] = \overline{\mathrm{Q}} + \mathrm{B}, \tag{8.10}$$

where

$$\mathrm{B} = \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}[\langle A_n\rangle_{T_0,T}] \tag{8.11}$$

is the bias of the initial condition, which can be mitigated if $\mathbb{E}[\langle A_n\rangle_{T_0,T}]$ decays sufficiently fast. If

$$\int_0^{\infty} |\mathbb{E}[A_n](t)|\ \mathrm{d}t < \infty, \tag{8.12}$$

then we have that

$$\begin{aligned} |\mathbb{E}[\langle A_n\rangle_{T_0,T}]| &= \frac{1}{T-T_0}\left|\int_{T_0}^{T}\mathbb{E}[A_n](t)\,\mathrm{d}t\right| \le \frac{1}{T-T_0}\int_{T_0}^{T}|\mathbb{E}[A_n](t)|\ \mathrm{d}t \\ &\le \frac{1}{T-T_0}\int_0^{\infty}|\mathbb{E}[A_n](t)|\ \mathrm{d}t \xrightarrow[T-T_0\to\infty]{} 0. \end{aligned} \tag{8.13}$$

In other words, if equation (8.12) holds, then increasing $T-T_0$ eventually decreases $|\mathrm{B}|$. An estimation of the decay rate can be made under stronger assumptions on the transient perturbation. For illustration purposes we consider a fast decay of the form

$$A_n^f(t) = A_{0,n}^f \mathrm{e}^{-\frac{t}{\tau}}, \tag{8.14}$$

which is an example of exponentially ergodic processes, where $\tau > 0$. For a wide class of stochastic processes satisfying a dissipativity condition, it can be proved that the transient perturbation decays exponentially [100, theorem 6.1], i.e. it satisfies

$$|\,\mathbb{E}[A^f]| = |\,\mathbb{E}[A_0^f]|\mathrm{e}^{-\frac{t}{\tau}}, \tag{8.15}$$

as in [54, equation (3)].
If we now consider its time average we get

$$\mathbb{E}[\langle A_n^f \rangle_{T_0,T}] = \mathbb{E}[A_{0,n}^f]\langle \mathrm{e}^{-\frac{t}{\tau}} \rangle_{T_0,T} = \tau\, \mathbb{E}[A_{0,n}^f] \left(\frac{\mathrm{e}^{-T_0/\tau} - \mathrm{e}^{-T/\tau}}{T - T_0} \right), \tag{8.16}$$

from where we see a decay of the form

$$\mathbb{E}[\langle A_n^f \rangle_{T_0,T}] = \mathcal{O}((T - T_0)^{-1}). \tag{8.17}$$

Therefore, $\mathbb{E}[\langle A_n^f \rangle_{T_0,T}]$ is a decreasing function of $T - T_0$. However, as it can be seen in equation (8.16), it is also decreasing when $T - T_0$ is kept constant while T and T_0 separately increase. In practice, T is fixed so increasing T_0 decreases $T - T_0$; this is the trade-off we analyze in the examples of section 8.2.

Statistical error

The previous analysis makes it clear that bias can be reduced by increasing $T - T_0$ and, for specific transient perturbations, increasing T_0. However, to assess statistical accuracy, equation (4.9) needs to be evaluated. Given a bound $\left|\mathrm{E}^N[\langle \mathrm{Q} \rangle_{T_0,T}] - \mathbb{E}[\mathrm{E}^N[\langle \mathrm{Q} \rangle_{T_0,T}]]\right| \leq \varepsilon$, the asymptotic normality of the estimator $\mathrm{E}^N[\langle \mathrm{Q} \rangle_{T_0,T}]$, in the limit $N \to \infty$, implies that [49, chapter 3]

$$\left|\mathrm{E}^N[\langle \mathrm{Q} \rangle_{T_0,T}] - \mathbb{E}[\mathrm{E}^N[\langle \mathrm{Q} \rangle_{T_0,T}]]\right| \leq \mathcal{C}_\phi \sqrt{\mathbb{V}[\mathrm{E}^N[\langle \mathrm{Q} \rangle_{T_0,T}]]} \leq \varepsilon, \tag{8.18}$$

with probability $1 - \phi$ as the tolerance $\varepsilon \to 0$. $\mathcal{C}_\phi$ is the confidence coefficient defined as $\mathcal{C}_\phi = \Phi^{-1}(1 - \frac{\phi}{2})$, there Φ is the CDF of a standard normal distribution. The total error in equation (4.9) can then be bounded with confidence $1 - \phi$, as follows,

$$\begin{aligned} \left|\mathrm{E}^N[\langle \mathrm{Q} \rangle_{T_0,T}] - \overline{\mathrm{Q}}\right| &\leq \left|\overline{\mathrm{Q}} - \mathbb{E}[\mathrm{E}^N[\langle \mathrm{Q} \rangle_{T_0,T}]]\right| + \left|\mathrm{E}^N[\langle \mathrm{Q} \rangle_{T_0,T}] - \mathbb{E}[\mathrm{E}^N[\langle \mathrm{Q} \rangle_{T_0,T}]]\right| \\ &\leq \left|\overline{\mathrm{Q}} - \mathbb{E}[\mathrm{E}^N[\langle \mathrm{Q} \rangle_{T_0,T}]]\right| + \mathcal{C}_\phi \sqrt{\mathbb{V}[\mathrm{E}^N[\langle \mathrm{Q} \rangle_{T_0,T}]]}. \end{aligned} \tag{8.19}$$

We define the SE to be $\mathrm{SE} = \sqrt{\mathbb{V}[\mathrm{E}^N[\langle \mathrm{Q} \rangle_{T_0,T}]]}$, consistently with chapter 4. Thus, using equation (8.10), we get

$$\left|\mathrm{E}^N[\langle \mathrm{Q} \rangle_{T_0,T}] - \overline{\mathrm{Q}}\right| \leq |\mathrm{B}| + \mathcal{C}_\phi \mathrm{SE}, \tag{8.20}$$

where B is the initialization bias defined in equation (8.11). For a given confidence $1-\phi$, the probability convergence criteria then reads

$$|\mathrm{B}| + \mathcal{C}_\phi \mathrm{SE} \leq \varepsilon. \tag{8.21}$$

The bias error $|\mathrm{B}|$ was analyzed above, let us now focus on the SE term. Assuming each A_n and S_n are independent, we have

$$\begin{aligned}\mathbb{V}[\mathrm{E}^N[\langle \mathrm{Q}\rangle_{T_0,T}]] &= \frac{1}{N^2}\sum_{n=1}^{N}\mathbb{V}[\langle S_n\rangle_{T_0,T}] + \frac{1}{N^2}\sum_{n=1}^{N}\mathbb{V}[\langle A_n\rangle_{T_0,T}] \\ &+ \frac{1}{N^2}\sum_{\substack{n,m=1\\ n\neq m}}^{N}\mathrm{co}\mathbb{V}[\langle \mathrm{Q}_n\rangle_{T_0,T}, \langle \mathrm{Q}_m\rangle_{T_0,T}].\end{aligned} \tag{8.22}$$

The first term on the right-hand side of equation (8.22) can be written as

$$\mathbb{V}[\langle S_n\rangle_{T_0,T}] = 2\frac{\mathbb{V}[S]}{T-T_0}\int_{T_0}^{T}\left(1-\frac{t}{T-T_0}\right)\rho(t)\,\mathrm{d}t, \tag{8.23}$$

where ρ is the autocorrelation function, see [112, problem 3.37]. The long-time limit of the integral in equation (8.23) is the integral time scale of the process [112, section 3.6], which is a correlation constant associated to the quantity of interest. Therefore, $\mathbb{V}[\langle S_n\rangle_{T_0,T}]$ decays like $(T-T_0)^{-1}$.

The second term on the right-hand side of equation (8.22) cannot be estimated without making assumptions on the behavior of the transient perturbation A. If we consider the same fast decay of the previous subsection, equation (8.14), a straightforward computation shows that $\mathbb{V}[\langle A_n^f\rangle_{T_0,T}] = \mathcal{O}((T-T_0)^{-2})$.

The last term in equation (8.22) depends on the correlation between realizations. An example of the effects of the correlation between realizations of turbulent flow in a channel is presented in [90]. If these realizations are independent, the final term in equation (8.22) is negligible and the dominant term in equation (8.21) depends on the decay rate of the transient perturbations A_n. To this end, we discuss two initial condition strategies which help to provide independent realizations in section 8.1.3.

If the decay of the transient perturbation is slower than equation (8.14), the decay of the statistical error will be dominated by the second term in equation (8.22). Therefore, both the bias and statistical error will decay at the same rate, and the left-hand side of equation (8.21) will decay like $(T - T_0)^{-q}$ for some $q < 1$. If the decay of the transient perturbation is fast (equation (8.14) holds), the overall error will be dominated by the first term in equation (8.22), and the left-hand side of equation (8.21) will decay like $N^{-0.5}$ and $(T - T_0)^{-0.5}$. In the numerical experiments presented in section 8.2, we verify the decay rates to assess if the initialization bias can be neglected in computing the error.

The left-hand side of equation (8.20) is estimated by approximating the variance by the sample variance. Because we aim at computing the variance of an average, we perform K repetitions of each experiment, totaling KN independent simulations. In this way, the right-hand side of equation (8.20) can be approximated as follows,

$$\mathcal{C}_\phi \sqrt{\mathbb{V}[\mathrm{E}^N[\langle \mathrm{Q} \rangle_{T_0,T}]]} \approx \mathcal{C}_\phi \sqrt{\mathrm{V}^K[\mathrm{E}^N[\langle \mathrm{Q} \rangle_{T_0,T}]]}, \tag{8.24}$$

with K sufficiently large.

8.1.3 On the generation of initial conditions

Ensemble averaging benefits from independent initial conditions to generate uncorrelated flow evolutions. It is known that different turbulent flows will diverge with a rate determined by the Lyapunov exponent [106, 104], and that this is the case of our target problems. We decide then to generate perturbed initial conditions, and to let the system evolve for a defined burn-in time T_0 to arrive at uncorrelated solutions. In this section, we discuss what to assign as a distribution for the initial conditions. To this end, two different types of Gaussian random vector fields are considered; white noise perturbations and spatially-correlated solenoidal fields.

The first approach simply consists of adding Gaussian white noise to a precomputed average velocity field $\langle \boldsymbol{u} \rangle$. In this work, this strategy of generating *spatially-uncorrelated* fluctuations is referred to as the spatially uncorrelated (SU) approach. We note that it is similar to the approach used in [90].

The second approach consists of adding nonlocal *spatially-correlated* and *divergence-free* solenoidal noise to the averaged velocity field $\langle \boldsymbol{u} \rangle$; we refer to this as the spatially correlated (SC) approach. Exploiting solenoidal fluctuations in the initial conditions is not new; we refer for example to [84], where the author used uncorrelated divergence-free initial conditions to ensure independence of different realizations. Our novelty is that we propose to generate spatially-correlated fluctuations $\boldsymbol{w}(\boldsymbol{x})$, which arise from a well-established synthetic turbulence model.

Our approach is inspired by the work of Hunt in [71] (see also [72, 105, 79]). The underlying assumption is that the inhomogeneous contributions to fully developed turbulence fluctuations in the inviscid source layer above a solid body have negligible vorticity. From this assumption, one arrives at the following inhomogeneous turbulent fluctuation model: $\boldsymbol{w}(\boldsymbol{x}) = \boldsymbol{w}^{(\mathrm{H})}(\boldsymbol{x}) - \nabla\phi(\boldsymbol{x})$, where $\boldsymbol{w}^{(\mathrm{H})}(\boldsymbol{x})$ is a homogeneous turbulent velocity field and $\phi(\boldsymbol{x})$ satisfies

$$\Delta\phi = \nabla \cdot \boldsymbol{w}^{(\mathrm{H})} \quad \text{in } D, \quad (\nabla\phi - \boldsymbol{w}^{(\mathrm{H})}) \cdot \boldsymbol{n} = 0 \quad \text{on } \partial D. \tag{8.25}$$

In this work, we adopt the classical von Kárman model [135] for the homogeneous random field $\boldsymbol{w}^{(\mathrm{H})}(\boldsymbol{x})$. Realizations of this type of nonlocal spatially-correlated field can be generated using a Fourier transform on a Cartesian grid containing D; see, e.g., [92]. Once a realization $\boldsymbol{w}^{(\mathrm{H})}(\boldsymbol{x})$ is generated, we may interpolate the boundary conditions so that the solution to equation (8.25) can be solved with the same finite element spaces used in equation (2.10). After interpolating the sum $\boldsymbol{w}^{(\mathrm{H})}(\boldsymbol{x}) - \nabla\phi$, we arrive at the nonlocal spatially-correlated perturbation $\boldsymbol{w}(\boldsymbol{x})$ and, in turn, the SC initial condition $\boldsymbol{u}_0(\boldsymbol{x}) = \langle\boldsymbol{u}\rangle(\boldsymbol{x}) + \boldsymbol{w}(\boldsymbol{x})$.

8.1.4 On the optimal choice of the burn-in time

When a single long simulation is performed, the burn-in time is small compared to the remaining simulation time, which contains the effective dynamics. Unfortunately, this is not the case when the same amount of simulation time is distributed across an ensemble. Indeed, the same burn-in time will be paid by all realizations in the ensemble and, as a consequence, the total effective time will be reduced. The reduction of the burn-in time is therefore key to making ensemble averaging feasible. Our statistical model provides the tools to analyze the bias associated with the initial conditions, thus allowing us to faithfully select a practical burn-in time.

Given the full time interval $[0, T]$, we split it into a burn-in time interval $[0, T_0]$ and an effective time interval $[T_0, T]$. In this subsection, we focus on how to optimally choose T_0.

First, a single simulation is executed for a time long enough to reach a statistically stationary turbulent state, which is saved. Thereafter, N realizations are run with SU or SC initial conditions to ensure independent flow evolution. Once the required transient time T_0 is passed, statistical data are collected and updated on the fly, until the end of the effective time window.

We propose a systematic manner to minimize T_0, which makes use of the SE defined above. Given N realizations and a quantity of interest Q, our idea is to analyze how the statistical estimates of the QoI change for different burn-in times. We can observe this plotting the mean $\mathrm{E}^N[\langle \mathrm{Q} \rangle_{T_0,T}]$ as function of T_0, together with its confidence intervals. The time interval $T - T_0$ is kept constant, while varying T_0. The confidence intervals are computed as $\mathcal{C}_\phi$SE, with confidence $1 - \phi$. By looking at the plot, we can detect a starting point after which the statistical result is effectively insensitive to T_0 variations. In addition to the statistical checks, we decide to apply a physical constraint, which in our case is the time the flow needs to go from the inlet to the obstacle. Therefore, T_0 will be the *maximum* of these two time values. In order to further reduce the computational cost of the transient phase, we also explore the possibility of using larger time steps in $[0, T_0]$.

Another way to estimate T_0 is analyzed in [21], where the authors choose a burn-in time which minimizes the estimated variance of the sample average estimator of the time average for a given signal. To do so, we average at each time step over all realizations, for different numbers of realizations N, and apply the procedure to the resultant time signal. As we will see in section 8.2, both procedures give similar results.

8.2 Numerical experiments

The practical question we address in this section is: How efficient is the ensemble approach in the context of under-resolved LES methods, in particular, in wind engineering applications? Although we have targeted a specific class of engineering problems, our strategies are general and can be applied to assess the ensemble average approach for other, unrelated, problems.

The first problem we consider (section 8.2.1) is the incompressible flow around a two-dimensional rectangle, already introduced in section 7.1. We first check that statistical results are independent of the initial condition strategies, and we compare ensemble average against standard time averaging. Then, we check if it is possible to exploit a larger CFL during the burn-in time phase, and how much the burn-in time window can be reduced. Finally, a comparison study between different strategies is made.

The second problem is presented in section 8.2.2 and describes wind flowing around a three-dimensional building; this problem has been introduced in section 7.2. A comparison between ensemble averaging and standard time averaging is presented, together with the burn-in time study.

For solving the CFD problems, we use Kratos [44, 43] as FE solver software, XMC [13] as hierarchical MC library and PyCOMPSs [16, 88, 126] as programming model for distributed computing. The integration of these software has been an important part of this thesis work, as documented in [127, 10, 11, 5, 4, 22]. We remark that international units are used to measure physical quantities.

The analyses were run on MareNostrum 4. This supercomputer has 11.15 Petaflops of peak performance, which consists of 3456 compute nodes equipped with two Intel R Xeon Platinum 8160 (24 cores at 2.1 GHz each) processors.

8.2.1 Rectangle obstacle problem

We remark that the rectangle obstacle problem is analyzed and solved with different configurations. In section 7.1, constant in time stochastic boundary conditions are considered, and the problem is solved with hierarchical MC methods. In section 8.2.1, constant in time deterministic boundary conditions are considered, and the problem is solved with the standard time averaging method and the ensemble averaging method. In section 9.2.1, constant in time stochastic boundary conditions are considered, and the problem is solved with the ensemble-based MC method. Therefore, in this section we use constant in time deterministic boundary conditions.

Problem formulation

We consider the two-dimensional flow around a rectangle body [28], introduced in section 7.1. The domain is reported in figure 7.1 and the system is described in section 7.1. The flow is modeled by the incompressible NS equations (see equation (2.1)) and we recall that slip boundary conditions are applied on the external boundaries and no-slip boundary conditions on the rectangle body. Standard air density and viscosity are considered and the Re is 132719.

The mesh considered to solve the problem has around 25000 nodes, and a minimal size, close to the rectangle body, of 0.002 m. The chosen time step is 0.02 s, which gives a CFL of 20. Such mesh is adaptive with respect to a solution-oriented metric, namely the time-averaged velocity field $\langle \boldsymbol{u}(t, x) \rangle_{T_0,T}$. The metric is computed exploiting Kratos [94], and the original mesh is refined using the Mmg software [45]. Further details are presented in chapter 6. The problem is validated in section 7.1.2.

The quantities of interest are the drag force F_d on the body, the pitching moment M_p on the body and the pressure field $p(x)$ on all nodes of the body surface. However, even though we compute all these quantities of interest, we assess statistical convergence only for the drag force. Therefore we set $\mathrm{Q} \equiv F_d$. For the sake of simplicity, we may use $\langle \mathrm{Q} \rangle$ instead of $\langle \mathrm{Q} \rangle_{T_0,T}$.

Perturbation of initial conditions

We can observe in figure 8.1 the time evolution of the time-averaged drag force $\langle F_d \rangle_{40\,\mathrm{s},t}$ for 128 contributions, when using the correlated and divergence-free initial condition strategy. The burn-in time we select is 40 s, which is the optimal T_0 we find below. Similar plots are obtained for SU initial conditions and for others T_0.

Each of these samples runs for 600 s, which, as we can observe by looking at the oscillations we have, is a time horizon not long enough to reach convergence for the time-averaged drag value. In the case of infinitely large time windows, one would expect each realization to converge to the same value. Since this is not feasible, figure 8.1 gives us an estimate of the error that is being committed by considering truncated time windows. The estimations of expected value, standard deviation and statistical error for both perturbations are reported in table 8.1.

	$\mathrm{E}^N[\mathrm{Q}]$	$\sigma^N[\mathrm{Q}]$	$\mathcal{C}_\phi\mathrm{SE}$
SU	3.238648	0.052823	0.010819
SC	3.246658	0.055877	0.011444

Table 8.1: Estimations of expected value, standard deviation and statistical error with 99% confidence for SU and SC initial conditions. The quantity of interest Q is the time-averaged drag force $\langle F_d \rangle_{40\,\mathrm{s},600\,\mathrm{s}}$. 128 realizations are considered.

Comparison ensemble average and time average

In order to compare ensemble averaging and standard time averaging approaches, we compute and compare the total error, given by the left-hand side of equation (8.21), for different computational costs.

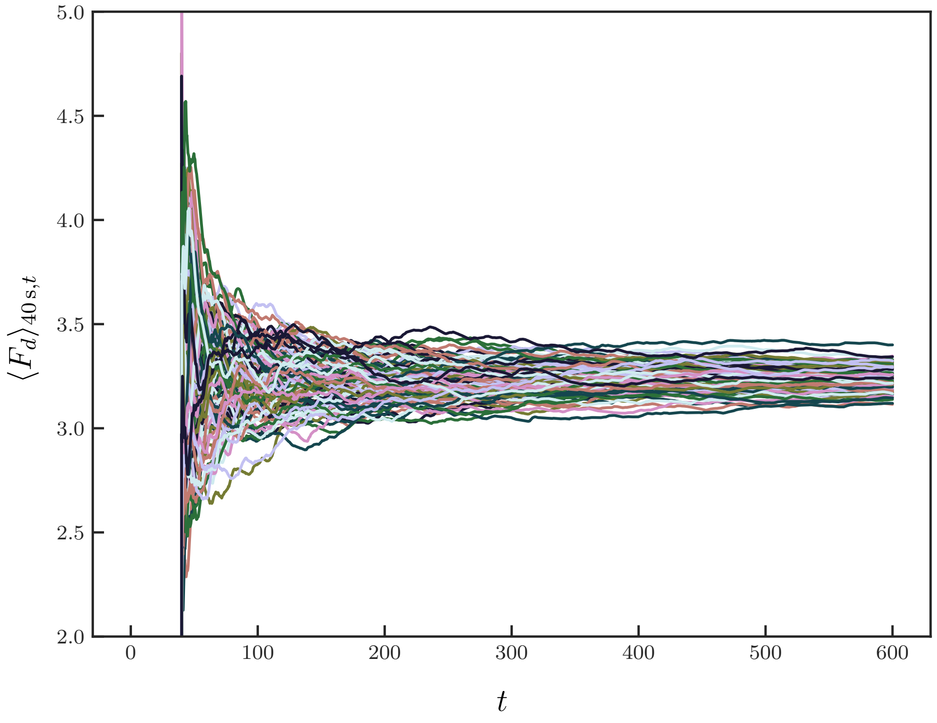
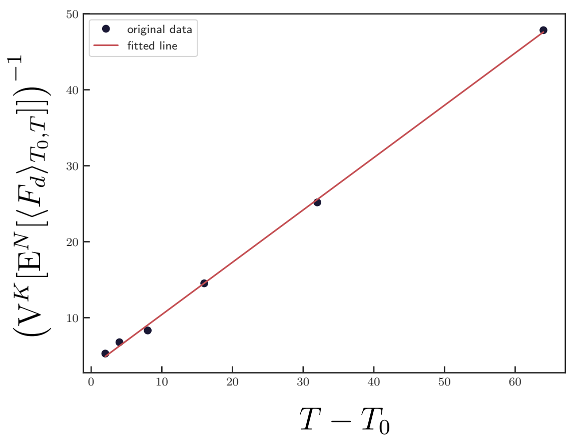


Figure 8.1: Time-averaged drag force $\langle F_d \rangle_{40\,\mathrm{s},t}$ evolution as function of time. Initial conditions are perturbed following the SC initial conditions.

First, we plot $\left(\mathrm{V}^K[\mathrm{E}^N[\langle \mathrm{Q} \rangle_{T_0,T}]]\right)^{-1}$ in figure 8.2 to analyze which are the dominant terms of equation (8.21), where $\mathrm{Q} = F_d$ and different K and N are considered. In the plot, black dots are estimations of the reciprocal of the variance, while the red line is the linear interpolation of such estimations. We observe that the variance estimation decays linearly as N and $(T - T_0)$ grow. Therefore, the fast decay of section 8.1.2 is happening and the dominant term of the total error is $\mathcal{C}_\phi \sqrt{\mathrm{V}^K[\mathrm{E}^N[\langle \mathrm{Q} \rangle_{T_0,T}]]}$. We can then simplify equation (8.21) to $\mathcal{C}_\phi \mathrm{SE} \approx \mathcal{C}_\phi \sqrt{\mathrm{V}^K[\mathrm{E}^N[\langle \mathrm{Q} \rangle_{T_0,T}]]} \leq \varepsilon$.

Tables 8.2 and 8.3 show the SE for both ensemble average and standard time average approaches. First, we observe SE decreases as expected (proportional to $N^{-0.5}$ and $(T - T_0)^{-0.5}$) as more realizations or larger time windows are considered. For example, let's focus on the second line of table 8.2 and the sixth of table 8.3. SE values are approximately similar, but ensemble average employs 10 samples that can be run concurrently. Consequently, the total computational cost corresponds to running 10 simulations, each with an effective time window of 160 s. Moreover, we point out that the expected value estimations of the two average strategies are in agreement with each other. As expected,



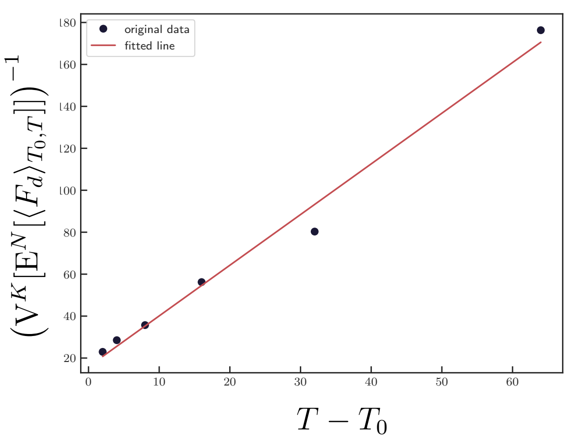


Figure 8.2: Computation of $\mathrm{V}^K[\mathrm{E}^N[\langle \mathrm{Q}\rangle_{T_0,T}]]$ as function of $T - T_0$ for $\mathrm{Q} = F_d$. The left plot presents $(K, N, T_0) = (128, 1, 40\,\mathrm{s})$ and the right plot $(K, N, T_0) = (32, 4, 40\,\mathrm{s})$.

$\mathrm{E}^N[\langle F_d\rangle]$	SE	N	$T - T_0$	C	time to solution
3.2779	0.0403	5	160	136	3.40
3.2684	0.0285	10	160	272	3.40
3.2583	0.0201	20	160	544	3.40
3.2527	0.0142	40	160	1088	3.40
3.2584	0.0100	80	160	2176	3.40
3.2456	0.0071	160	160	4352	3.40

Table 8.2: The table reports the mean estimation and its associated SE computed for ensemble averaging for the drag force mean estimation. N and $T - T_0$ refer to the number of realizations and the effective time window of the simulation, respectively. C is the computational cost and is expressed in CPU hours. Time to solution is the human time we need to wait for getting results, and is expressed in hours.

the ensemble average approach drastically reduces the time to solution, *for the same statistical error*, provided of course that more computing resources are used to enable the concurrent solution of the ensemble. This means that more working nodes are used as more realizations are run. If enough resources are allocated, the runtime is shorter, and this is our case.

The results suggest that the ensemble average approach is more appropriate than standard time averaging for running on supercomputers since it allows to fully exploit supercomputer capabilities in order to reduce the time to solution. This comes at the price of a larger combined computational cost, due to the need of going multiple times through

$\mathrm{E}^N[\langle F_d \rangle]$	SE	N	$T - T_0$	C	time to solution
3.3056	0.1595	1	50	17.2	2.15
3.2171	0.1127	1	100	21.76	2.72
3.2752	0.0797	1	200	30.8	3.85
3.2605	0.0563	1	400	48.96	6.12
3.2472	0.0398	1	800	85.28	10.66
3.2552	0.0281	1	1600	158.02	19.75

Table 8.3: The table reports the mean estimation and its associated SE computed for standard time averaging for the drag force mean estimation. N and $T-T_0$ refer to the number of realizations and the effective time window of the simulation, respectively. C is the computational cost and is expressed in CPU hours. Time to solution is the human time we need to wait for getting results, and is expressed in hours.

the initial burn-in time.

On the reduction of burn-in time computational cost

We analyze how the statistical results of the time-averaged drag force change when varying the burn-in time. For each case we consider 128 realizations, and we keep constant $T - T_0$. We plot in figure 8.3 the expected value estimation as function of T_0, together with its 99% confidence intervals. We observe that the statistical result is relatively insensitive to T_0 for $T_0 > 20\,\mathrm{s}$.

As mentioned in section 8.1, another way to estimate T_0 is following the approach presented in [21], in which the authors choose a burn-in time which minimizes the estimated variance of the sample average estimator of the time average for a given signal. Figure 8.4 reports the estimated variance of the sample average estimator of the time average as function of the burn-in time, for a different number of realizations and fixed $T - T_0$. As we can see, we reach the minimum after a few seconds.

Even though both ways of estimating the burn-in time suggest that only a very short time span is needed, we consider as "physical constraint" the time required by the information to travel from the inlet to past the object. This means that we wait at least the physical constraint time before we can start trusting the solver results. The time needed for this to happen, for an average speed of $2\,\mathrm{m/s}$, is $40\,\mathrm{s}$. Our conclusion is that we can safely assume $T_0 = 40\,\mathrm{s}$ without changing statistical results. We remark as well that the same conclusion follows for other effective time windows $T - T_0$.

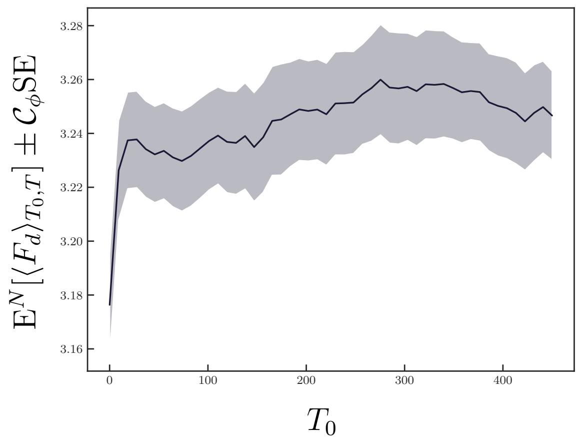


Figure 8.3: Expected value estimation and associated SE for a confidence of 99% as a function of the burn-in time. $N = 128$ and $T - T_0 = 150\,\mathrm{s}$.

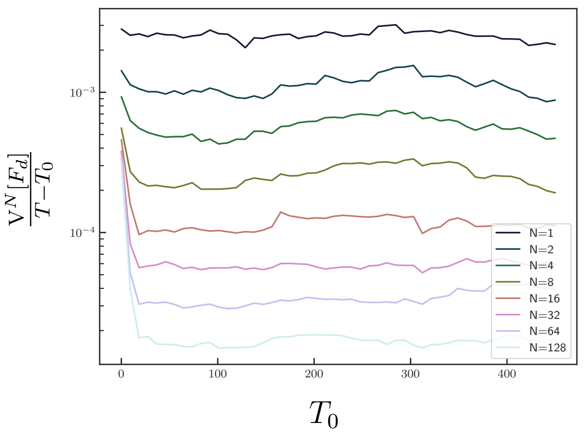


Figure 8.4: Ratio between drag force variance and effective time window, for different realizations and $T - T_0 = 150\,\mathrm{s}$.

Another approach we consider to reduce the time to solution is to *exploit larger time steps* during the burn-in time. To do so, we must first verify that a larger time step does not change the statistical results. Moreover, to improve the consistency of such an approach, we should ensure that the chosen larger time step, if used during the whole time window, would give different results.

We report in table 8.4 the expected value estimation and the associated SE for a confidence of 99%. We can see that in the case of time step $\Delta t = 0.02$ used in the averaging window $[T_0, T]$, we obtain consistent statistical results, independently of Δt_0 values. On the other hand, for a different Δt we obtain a different statistical result (see third row in the table compared to the first two). Therefore, a larger time step can safely be employed to reduce the time to solution of the burn-in phase.

$\mathrm{E}^N[\langle F_d \rangle_{T_0,T}]$	$\mathcal{C}_\phi\mathrm{SE}$	$T - T_0$	Δt	Δt_0
3.237284	0.009183	760	0.02	0.02
3.232869	0.008286	760	0.02	0.05
3.312648	0.008199	760	0.05	0.05

Table 8.4: Expected value estimation and associated SE with 99% confidence for different time steps during both burn-in and effective phases.

Results

Combining all of the ideas presented above we obtain the results reported in table 8.5, which shows the statistical analysis of the time-averaged drag force. The analyses presented in table 8.5 are driven by constant product between $N = 128$ and $T = 300\,\mathrm{s}$. Convergence is checked via equation (8.21), which is simplified to $\mathcal{C}_\phi \mathrm{SE} \leq \varepsilon$. The absolute tolerance is $\varepsilon = 0.02$ (the relative value is $\approx 0.6\%$) and the confidence is $1-\phi = 0.99$. Different T_0, Δt_0 and perturbation of initial conditions are considered. On one hand, we consider the optimal $T_0 = 40\,\mathrm{s}$, on the other $T_0 = 140\,\mathrm{s}$, which is directly related to the time one particle needs to travel from the inlet to the outlet for an average speed of $2\,\mathrm{m\,s^{-1}}$. We observe all strategies give the same statistical result since all expected value estimations fall within the range of confidence $1 - \phi = 0.99$. Moreover, both the time to solution and the computational cost are smaller if larger time steps are exploited in the burn-in time phase. Therefore, we conclude the most promising strategy consists in exploiting $\frac{\Delta t_0}{\Delta t} > 1$ and T_0 small enough but still ensuring the error decays as $N^{-0.5}$ and $(T - T_0)^{-0.5}$.

$\mathrm{E}^N[\langle F_d \rangle_{T_0,T}]$	$\mathcal{C}_\phi \mathrm{SE}$	initial conditions	N	$T - T_0$	T_0	$\frac{\Delta t_0}{\Delta t}$	C	time to solution
3.245649	0.018503	SU	128	160	140	1.0	3655	3.57
3.244356	0.013949	SU	128	260	40	1.0	3655	3.57
3.237846	0.014395	SU	128	260	40	2.5	3389	3.31
3.235604	0.019199	SC	128	160	140	1.0	3727	3.64
3.235677	0.014501	SC	128	260	40	1.0	3727	3.64
3.236612	0.014390	SC	128	260	40	2.5	3420	3.34

Table 8.5: Statistical analyses of time-averaged drag force $\langle F_d \rangle_{T_0,T}$. The expected value estimation and the associated statistical error, with a confidence $1 - \phi = 0.99$, are reported. Both uncorrelated and correlated initial condition perturbations are presented. N refers to the number of ensembles realizations. Effective time window $T - T_0$ and burn-in time T_0 are expressed in seconds, and $\frac{\Delta t_0}{\Delta t}$ shows if a larger CFL is used in the transient phase. The computational cost C and time to solution unit measures are CPU hours and hours, respectively. The product between number of realizations and time window is constant among different analyses.

Finally, we recall that the instantaneous velocity and pressure fields at $t = 200\,\mathrm{s}$ for one realization can be observed in figure 7.5 and figure 7.6.

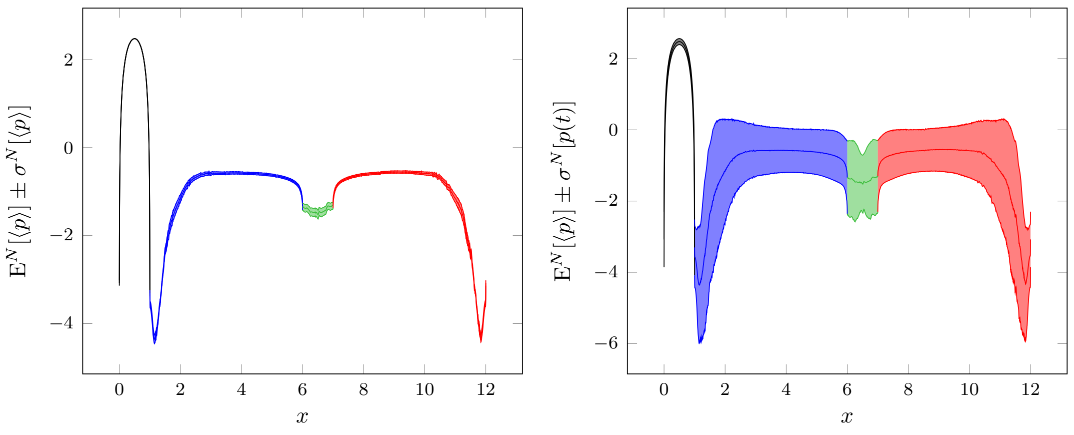
Other observables

In addition to the statistical analysis reported above for the drag force, we present here results for the expected value and standard deviation estimations of the drag force, pitching moment and pressure field on the rectangle body. Specifically, we compute the standard deviation of an observable Q and of its time average, which read $\sigma[\mathrm{Q}]$ and $\sigma[\langle \mathrm{Q} \rangle_{T_0,T}]$, respectively. $\sigma[\mathrm{Q}]$ can be understood as an indicator of the distribution around the mean value, while $\sigma[\langle \mathrm{Q} \rangle_{T_0,T}]$ as an error indicator of the expected value estimation.

Table 8.6 reports results for the drag force and the base moment, while figures 8.5a and 8.5b for the pressure field.

Q	$\mathrm{E}^N[\langle \mathrm{Q} \rangle_{T_0,T}]$	$\sigma^N[\langle \mathrm{Q} \rangle_{T_0,T}]$	$\sigma^N[\mathrm{Q}]$
F_d	3.238950	0.086650	0.575146
M_p	-0.014169	0.123063	2.141448

Table 8.6: Statistical analysis of the drag force F_d and of the pitching moment M_p.



(a) Estimation of the expected value and of the time-averaged standard deviation of the pressure field $p(\boldsymbol{x})$ around the rectangle body. Each color represent one side of the rectangle, and the color of each side is presented in figure 7.1.

(b) Estimation of the expected value and of the standard deviation of the pressure field $p(\boldsymbol{x})$ around the rectangle body. Each color represent one side of the rectangle, and the color of each side is presented in figure 7.1.

Figure 8.5: Statistical result of the time-averaged pressure field $\langle p(\boldsymbol{x}) \rangle_{T_0,T}$ and the pressure field $p(\boldsymbol{x})$.

Finally, we report in table 8.7 the CVaR results for the time-averaged drag force and the drag force of the last case of table 8.5. We remark to observe the difference between standard physical quantities and their time-averaged counterparts.

Q	CVaR	α
$\langle F_d \rangle_{T_0,T}$	3.34554	0.9
F_d	4.38821	0.9

Table 8.7: CVaR analysis of time-averaged drag force $\langle F_d \rangle_{T_0,T}$ and drag force F_d. Results for $N = 128$, $T - T_0 = 260\,\mathrm{s}$ and $\alpha = 0.9$.

8.2.2 High-rise building problem

We remark that the CAARC building problem is analyzed and solved with different configurations. In section 7.2, turbulent fluctuations around a fixed mean wind field boundary conditions are considered, and the problem is solved with hierarchical MC methods. In section 8.2.2, constant in time fixed mean wind field boundary conditions are considered, and the problem is solved with the standard time averaging method and the ensemble averaging method. In section 9.2.2, constant in time stochastic mean wind field boundary conditions are considered, and the problem is solved with the ensemble-based MC method. In section 9.2.3, turbulent fluctuations around a stochastic mean wind field boundary conditions are considered, and the problem is solved with the ensemble-based MC method. Therefore, in this section we use constant in time fixed mean wind field boundary conditions.

Problem formulation

The second problem is the wind flow around the CAARC building [107, 67, 24, 66], that is introduced in section 7.2. A steady state logarithmic wind profile is considered. The wind mean profile is described by [73]

$$\bar{\boldsymbol{u}}(z) = \frac{\boldsymbol{u}^*}{k} \ln(z/z_0), \tag{8.26}$$

where $k \approx 0.4$ is von Karman's constant, $\boldsymbol{u}^*$ the friction velocity and z_0 the roughness length. The system is described by the NS equations (see equation (2.1)), slip boundary conditions are applied on the walls and the ceiling, no-slip boundary conditions on the building and on the floor. The reference mean wind velocity $\bar{\boldsymbol{u}}(z)$ is defined at reference height H. We refer to section 2.4 for further details.

In table 7.4 we present the physical properties of the problem, and we recall that $z_0 = 2\,\mathrm{m}$ is typical of centers of large cities [73]. The Re is 119 millions, where a characteristic length of 45 m is considered. Under such conditions, it is clear that the problem is badly under-resolved.

The quantities of interest are the drag force F_d on the body, the base moment M_b on the body and the pressure field $p(x)$ on all nodes of the body surface. The quantity of interest we choose to analyze is the drag force. Therefore we set $\mathrm{Q} \equiv F_d$. As above, we omit the subscript T_0, T if there is no risk of misunderstanding.

The mesh considered to solve the problem has approximately 312000 nodes, and a minimal size, close to the body, of 0.2 m. The considered mesh is adaptive with respect to a metric built on top of velocity and pressure fields.

Validation

We compute and compare the normalized formulas for forces and moments (see equation (7.7)) against [24]. We recall that C_{F_X} and C_{F_Y} represent the force coefficients in the direction X and Y, respectively. M_{F_X} and M_{F_Y} denote the moment coefficients in the same directions, where the moment is computed around the centroid of the plan geometry of the building at ground location. We remark that, differently to the validation of section 7.2.2, the problem considered here has null wind gusts and we use a different domain discretization to solve it. Figure 8.6 shows a good agreement of our solution with respect to literature.

Comparison ensemble average and time average

We have observed in section 8.2.1 that perturbing initial conditions with SU or SC noise is equivalent, from both computational and statistical points of view. For this reason, we prefer to use the latter, since more consistent from a physical point of view.

First, we analyze in figure 8.7 which are the dominant terms of equation (8.21) by plotting $\left(\mathrm{V}^K[\mathrm{E}^N[\langle \mathrm{Q} \rangle_{T_0,T}]]\right)^{-1}$ for different K and N and $\mathrm{Q} = F_d$. The linear decay of the variance estimation with respect to N and $(T - T_0)$ suggests that the dominant term of the total error is $\mathcal{C}_\phi \sqrt{\mathrm{V}^K[\mathrm{E}^N[\langle \mathrm{Q} \rangle_{T_0,T}]]}$, which implies simplifying equation (8.21) to $\mathcal{C}_\phi \mathrm{SE} \approx \mathcal{C}_\phi \sqrt{\mathrm{V}^K[\mathrm{E}^N[\langle \mathrm{Q} \rangle_{T_0,T}]]} \leq \varepsilon$.

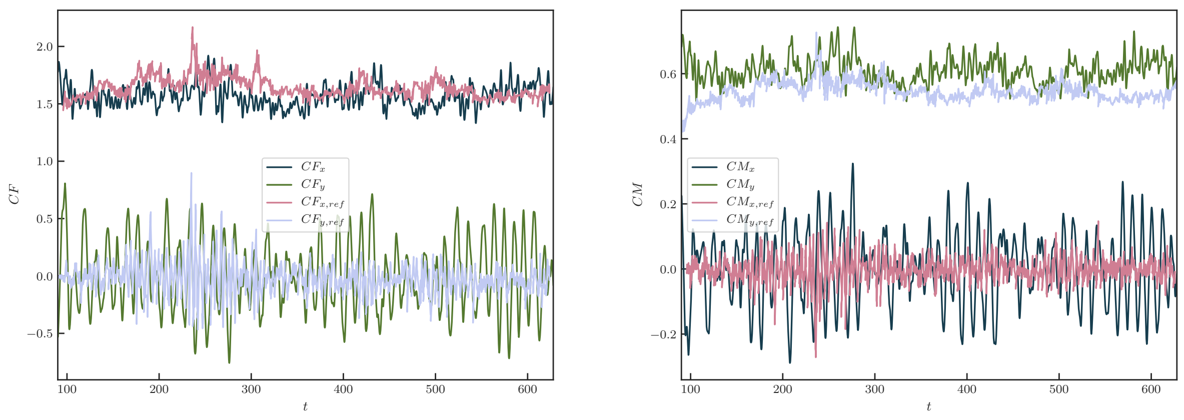


Figure 8.6: Drag and moment coefficients comparison between our work and [24]. The plot is done using the "web based tool WebPlotDigitizer to extract numerical data from plot images" [116].

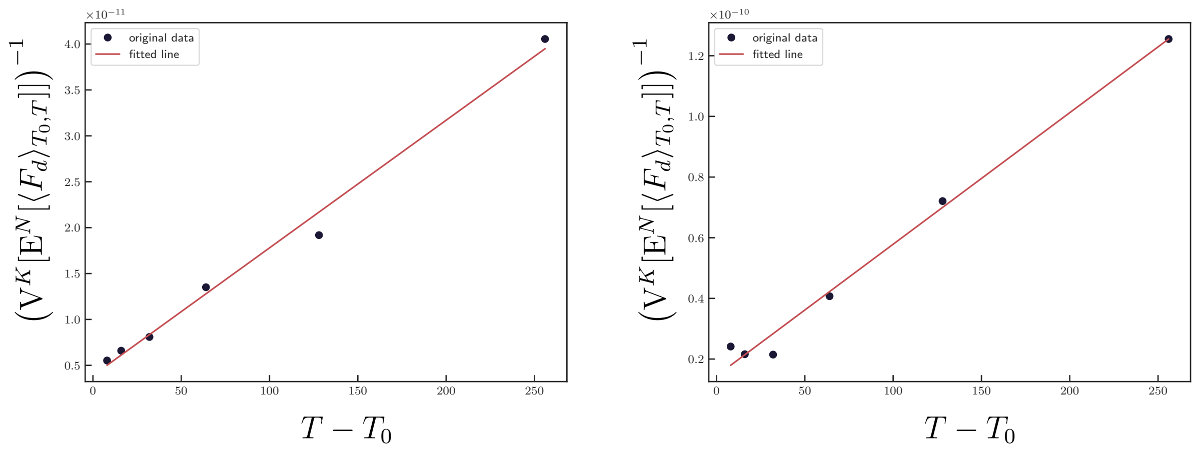


Figure 8.7: Computation of $\mathrm{V}^K[\mathrm{E}^N[\langle \mathrm{Q}\rangle_{T_0,T}]]$ as function of $T - T_0$ for $\mathrm{Q} = F_d$. The left plot presents $(K, N, T_0) = (128, 1, 30\,\mathrm{s})$ and the right plot $(K, N, T_0) = (32, 4, 30\,\mathrm{s})$.

We compute the SE for ensemble averaging and standard time averaging in tables 8.8 and 8.9. The SE decreases as expected as long as more realizations or larger time windows are considered. Moreover, the ensemble average approach drastically reduces the time to solution, for the same statistical error. For example, the case $N = 4$, $T - T_0 = 210\,\mathrm{s}$ of the ensemble average approach, compared to $N = 1$, $T - T_0 = 840\,\mathrm{s}$ of standard time average, reduces the time to solution by almost a factor 4, to obtain a similar SE. We remark as well that the expected value estimations for both ensemble averaging and standard time averaging are consistent within each other.

$\mathrm{E}^N[\langle F_d \rangle]$	SE	N	$T - T_0$	C	time to solution
8982493	84767	4	210	1666	17.36
8932223	59939	8	210	3333	17.36
8973444	42383	16	210	6666	17.36
8986913	29969	32	210	13332	17.36
8927955	21191	64	210	26664	17.36
8930547	14984	128	210	53329	17.36

Table 8.8: The table reports the mean estimation and its associated SE of ensemble averaging for the estimation of the drag force mean. N and $T - T_0$ refer to the number of realizations and the effective time window of the simulation, respectively. C is the computational cost and is expressed in CPU hours, while time to solution is expressed in hours.

$\mathrm{E}^N[\langle F_d \rangle]$	SE	N	$T - T_0$	C	time to solution
8658884	324256	1	52.5	215.58	8.98
8879404	229284	1	105	300.19	12.50
9109719	162128	1	210	468.56	19.52
9003445	114642	1	420	853.28	35.55
8950303	93604	1	630	1189.42	49.55
8956216	81064	1	840	1524.88	63.53

Table 8.9: The table reports the mean estimation and its associated SE of standard time averaging for the estimation of the drag force mean. N and $T - T_0$ refer to the number of realizations and the effective time window of the simulation, respectively. C is the computational cost and is expressed in CPU hours, while time to solution is expressed in hours.

On the reduction of burn-in time computational cost

We analyze now if it is statistically consistent to reduce the burn-in time. As before, we consider 128 realizations and we keep constant $T - T_0 = 110\,\mathrm{s}$. By looking at figures 8.8 and 8.9, we conclude that the burn-in time can be reduced to any value larger than 30 s. The same conclusion holds also for different effective time windows. To ensure robustness of our strategy, we apply the physical constraint that T_0 should be larger than the time to travel from the inlet to the body. Since for an average velocity of $40\,\mathrm{m\,s^{-1}}$ such time is 11.625 s, it is statistically consistent to use $T_0 = 30\,\mathrm{s}$.

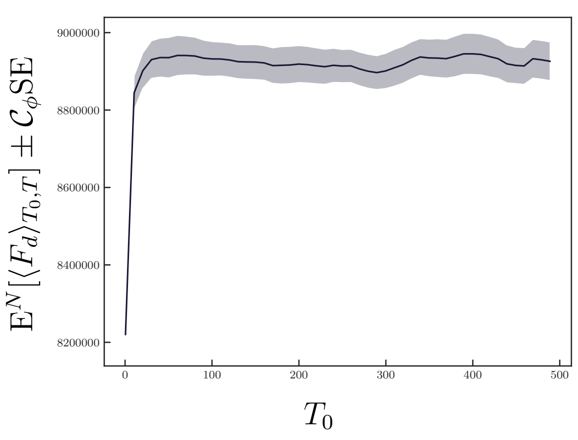


Figure 8.8: Expected value estimation and associated SE for a confidence of 99% as a function of the burn-in time. $N = 128$ and $T - T_0 = 110\,\mathrm{s}$.

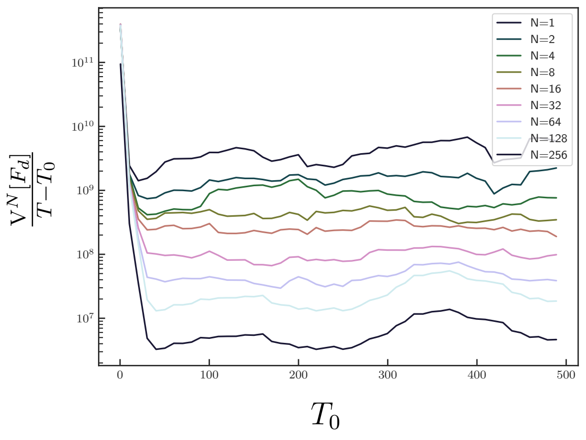


Figure 8.9: Ratio between drag force variance and effective time window, for different realizations and $T - T_0 = 110\,\mathrm{s}$.

We also check if it is possible to exploit larger time steps during the burn-in time, in order to reduce its computational cost. Table 8.10 shows that running with a larger time step during T_0 is statistically equivalent to exploit a constant time step, where $\mathcal{C}_\phi$ is computed for a 99% confidence.

$\mathrm{E}^N[\langle F_d \rangle_{T_0,T}]$	$\mathcal{C}_\phi\mathrm{SE}$	$T - T_0$	$\frac{\Delta t_0}{\Delta t}$
8922399	39013	170	1.0
8907406	39534	170	2.5

Table 8.10: Expected value and associated SE with 99% confidence for different time steps during burn-in time.

Results

Finally, we run the problem exploiting ensemble average, larger time step Δt_0 and smaller burn-in time $T_0 = 30\,\mathrm{s}$. Convergence is checked with equation (8.21), which is simplified to $\mathcal{C}_\phi\mathrm{SE} \leq \varepsilon$. The chosen confidence is 99%, and the relative tolerance with respect to the time-averaged drag force mean estimator is around 0.5%. We run the problem for different configurations, keeping the overall cost given by the product between time window and number of realizations TN approximately constant. Results are shown in table 8.11.

$\mathrm{E}^N[\langle F_d\rangle_{T_0,T}]$	$\mathcal{C}_\phi$SE	N	$T - T_0$	T_0	$\frac{\Delta t_0}{\Delta t}$	C	time to solution
8972727	48924	142	110	30	2.5	35749	10.34
8946768	45797	100	170	30	2.5	38041	15.54
8943515	40755	76	230	30	2.5	38566	20.60

Table 8.11: The table reports the expected value and the statistical error values of the time-averaged drag force $\langle F_d\rangle_{T_0,T}$, with a 99% confidence. N, T and T_0 refer to the number of ensemble realizations, the time window $[0,T]$ upper bound of the simulation and the burn-in time, respectively. These last two are measured in seconds. $\frac{\Delta t_0}{\Delta t}$ is the ratio between the time steps of T_0 and of the effective time window $T - T_0$. C is the computational cost, expressed in CPU hours, and time to solution is the real time we need to wait for the solution and is expressed in hours.

We report in figure 8.10 and figure 8.11 the instantaneous velocity and pressure fields at $t = 200\,\mathrm{s}$ for one realization.

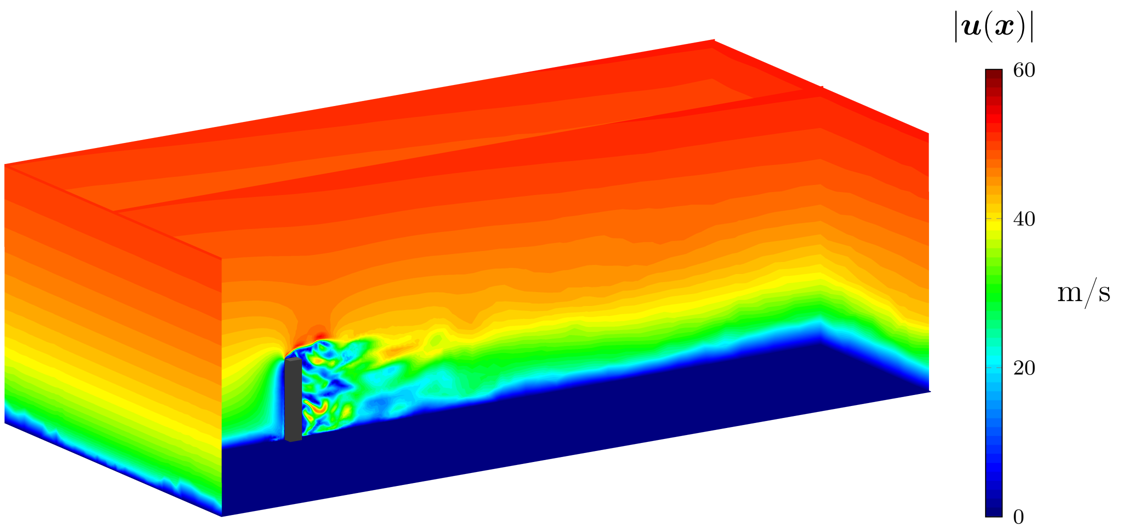


Figure 8.10: Velocity field snapshot at $t = 200\,\mathrm{s}$.

Other observables

We select the case with minimal statistical error of table 8.11 to show the statistical results for other quantities of interest. Table 8.12 shows the expected value and the standard deviation estimators for the drag force and the base moment. Figures 8.12

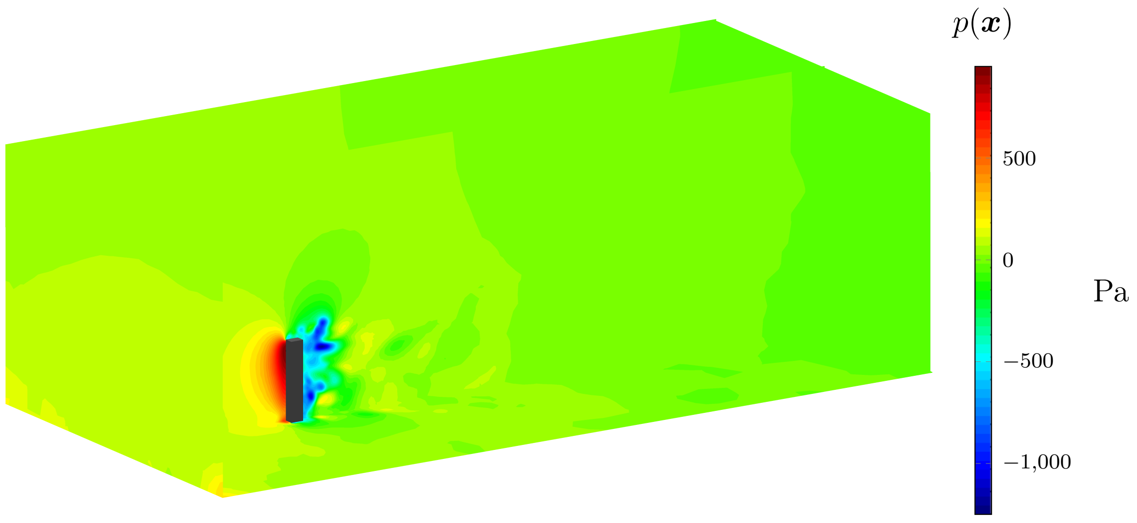


Figure 8.11: Pressure field snapshot at $t = 200\,\mathrm{s}$.

and 8.13 show the estimations of expected value and standard deviation for the pressure field.

Q	$\mathrm{E}^N[\langle \mathrm{Q}\rangle_{T_0,T}]$	$\sigma^N[\langle \mathrm{Q}\rangle_{T_0T}]$	$\sigma^N[\mathrm{Q}]$
F_d	8943515	152726	662129
M_b	-14943	436540	7332992

Table 8.12: Statistical analysis of drag force and base moment.

Finally, we report in table 8.13 the CVaR results for the time-averaged drag force and the drag force of the case with minimal statistical error of table 8.11. We remark to observe the difference between standard physical quantities and their time-averaged counterparts.

Q	CVaR	α
$\langle F_d\rangle_{T_0,T}$	9208076	0.9
F_d	10291350	0.9

Table 8.13: CVaR analysis of time-averaged drag force $\langle F_d\rangle_{T_0,T}$ and drag force F_d. Results for $N = 76$, $T - T_0 = 230\,\mathrm{s}$ and $\alpha = 0.9$.

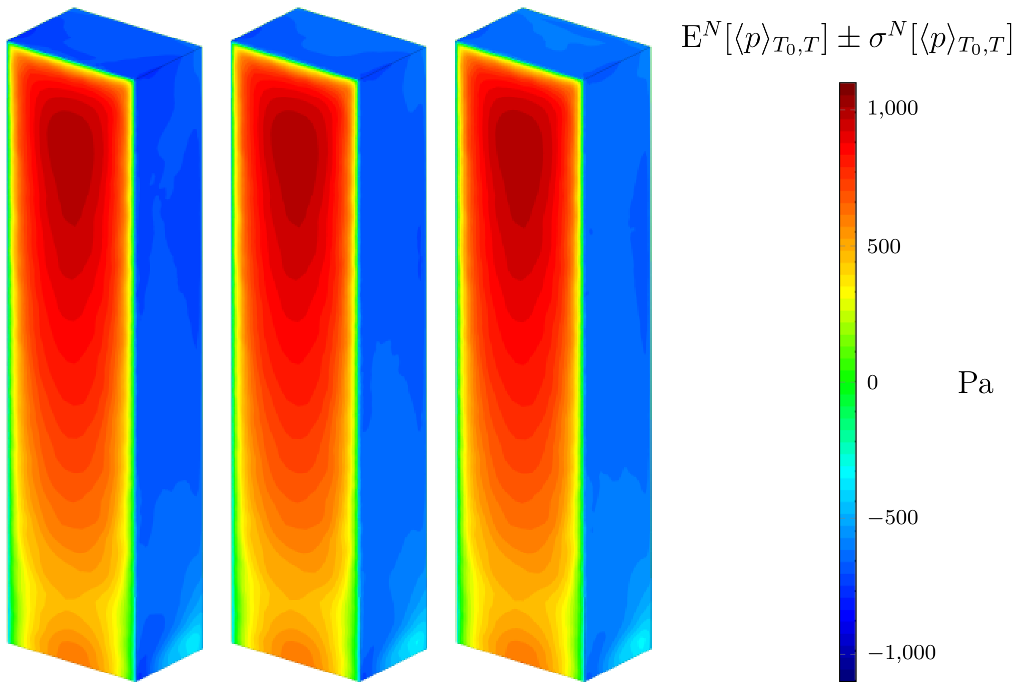

$\mathrm{E}^N[\langle p\rangle_{T_0,T}] \pm \sigma^N[\langle p\rangle_{T_0,T}]$
1,000
500
0
−500
−1,000
Pa

Figure 8.12: Statistical result of the pressure field $\langle p(\boldsymbol{x})\rangle_{T_0,T}$. From left to right, $\mathrm{E}^N[\langle p(\boldsymbol{x})\rangle_{T_0,T}] - \sigma^N[\langle p(\boldsymbol{x})\rangle_{T_0,T}]$, $\mathrm{E}^N[\langle p(\boldsymbol{x})\rangle_{T_0,T}]$ and $\mathrm{E}^N[\langle p(\boldsymbol{x})\rangle_{T_0,T}] + \sigma^N[\langle p(\boldsymbol{x})\rangle_{T_0,T}]$.

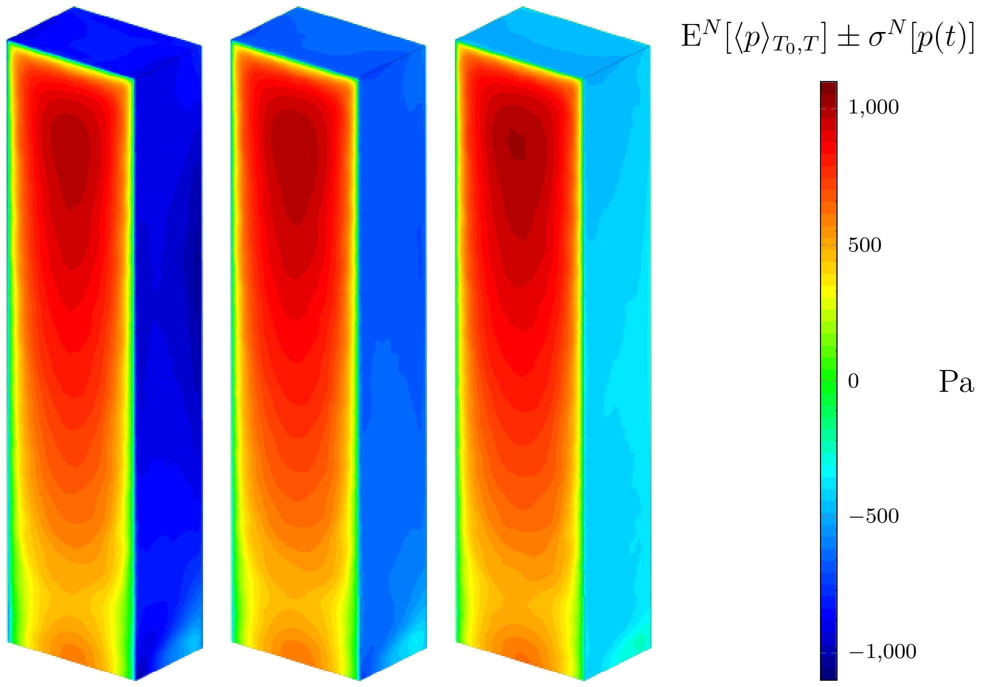


Figure 8.13: Statistical result of the pressure field $p(\boldsymbol{x})$. From left to right, $\mathrm{E}^N[\langle p(\boldsymbol{x})\rangle_{T_0,T}] - \sigma^N[p(\boldsymbol{x})]$, $\mathrm{E}^N[\langle p(\boldsymbol{x})\rangle_{T_0,T}]$ and $\mathrm{E}^N[\langle p(\boldsymbol{x})\rangle_{T_0,T}] + \sigma^N[p(\boldsymbol{x})]$.

8.3 Summary

In this chapter, we show that ensemble averaging can be successfully applied to highly chaotic incompressible flows and we propose strategies to minimize the total error and the computational cost of the simulation. Two numerical examples are considered to demonstrate the advantage of using ensemble averaging over standard time averaging when running on HPC systems and to validate our proposals.

The statistical analysis of ensemble averaging expected value estimator leads to the identification of two error components: an initialization bias, related to the transient perturbation of the flow, and a statistical error, related to finite sampling. Convergence rates of both error contributions are analyzed by considering two scenarios: one with fast decay of the transient perturbation and one with slow decay. This allows understanding how the error contributions should decay in order to assume null initialization bias. For both numerical examples, decay rates are estimated to assess if the initialization bias is negligible. For both problems, the burn-in phase computational cost is minimized by following a statistical-based approach and a less accurate and less expensive time integration procedure during the burn-in phase.

Multiple observables (drag force, base and pitching moment and pressure field) are computed. By applying the proposed statistical ensemble averaging framework, statistical estimators are efficiently and accurately estimated, and decisions based on top of such statistics can therefore be taken faster.

Chapter 9

Ensemble-based hierarchical Monte Carlo methods for chaotic systems

In this chapter, we propose a non-intrusive approach that integrates ensemble averaging and hierarchical MC methods for reducing the time to solution required to solve stochastic turbulent flow problems. Our idea consists of running hierarchical MC methods to quantify uncertainties and applying ensemble averaging to each MC realization to reduce the overall runtime of the single instance. The advantage of this strategy is that all simulations can be launched independently and run in parallel, thus providing an obvious opportunity for acceleration when abundant computational resources are available.

We are interested in solving turbulent flow problems which present stochastic boundary conditions. We recall that in section 8.1 we propose a statistical framework for ensemble averaging for solving problems with known boundary conditions, and random initial conditions. In this chapter, we propose an extension of such a statistical framework in section 9.1. Our analysis permits to assess the efficacy of ensemble averaging for solving stochastic CFD problems of engineering interest, and is validated in section 9.2 by solving two challenging wind engineering problems.

The content of this chapter is taken from a manuscript in preparation and is adapted wherever needed.

9.1 Statistical analysis

In this section, we extend our statistical framework of section 8.1 to the case of stochastic turbulent flows. In section 9.1.1 we recall some preliminary notation and we define the problem we aim at solving. The sources of error of section 8.1.2 are generalized in section 9.1.2.

In agreement with what introduced in chapter 3, the expected value of a process X is denoted $\mathbb{E}[X]$, its variance is denoted $\mathbb{V}[X]$ and its covariance with a process Y is denoted $\mathrm{co}\mathbb{V}[X, Y]$. These quantities can be estimated by sample averaging. For instance, for any N independent samples, $X_1, \dots, X_N$, we define the sample mean $\mathrm{E}^N[X] = \frac{1}{N}\sum_{n=1}^{N} X_n$ and the sample variance $\mathrm{V}^N[X] = \frac{1}{N-1}\sum_{n=1}^{N}(X_n - \mathrm{E}^N[X])^2$.

Let us consider the NS problem of equation (2.1), integrated with initial conditions and wind inlet boundary conditions. The problem then reads

$$
\begin{aligned}
\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u} - \nu\Delta\boldsymbol{u} + \nabla p = \boldsymbol{f} \qquad & \text{in } [0,T]\times D \\
\nabla\cdot\boldsymbol{u} = 0 \qquad & \text{in } [0,T]\times D \\
\boldsymbol{u} = \boldsymbol{u}_0(\boldsymbol{x}) \qquad & \text{in } t=0\times D \\
\boldsymbol{u} = f(z, z_0) \qquad & \text{on } [0,T]\times\Gamma_{\text{in}},
\end{aligned}
\tag{9.1}
$$

where D is the domain, Γ_{in} refers to the inlet boundary of the domain and $[0, T]$ is the time window. We recall that the incoming velocity field $\boldsymbol{u} = \overline{\boldsymbol{u}} + \boldsymbol{u}'$ is decomposed into its stationary mean profile $\overline{\boldsymbol{u}}$ and its unsteady turbulent fluctuations $\boldsymbol{u}'$. Moreover, z_0 is the roughness height and we refer to section 2.4 for more details about wind modeling. z is the vertical coordinate and $f(\cdot)$ is a generic function useful to express dependencies.

9.1.1 Problem outline

From the definition of equation (9.1), we have $\boldsymbol{u}_0(\boldsymbol{x}, w_1)$ that is a random field, but also $z_0(w_2)$ is a random variable.

Remark 1. *If $\boldsymbol{u}\prime \neq 0$, wind gusts dominate over initial conditions $\boldsymbol{u}_0(\boldsymbol{x}, w_1)$ and we can therefore consider $\boldsymbol{u}\prime(w_1)$ as random variable.*

We describe this situation with $\Omega = \Omega_1 \times \Omega_2$, where $w_1 \in \Omega_1$ and $w_2 \in \Omega_2$ are independent uncertainties and Ω_1 and Ω_2 are their sample spaces [117, chapter 6]. Then,

$\mathbb{P} = \mathbb{P}_1 \times \mathbb{P}_2$ and the expected value of a generic joint random variable $X(w_1, w_2)$ is

$$\mathbb{E}[X] = \int_{\Omega_1\times\Omega_2} X(w_1, w_2)\,\mathrm{d}\Omega_1 \times \Omega_2(w_1, w_2) = \int_{\Omega_1}\int_{\Omega_2} X(w_1, w_2)\,\mathrm{d}\Omega_2(w_2)\,\mathrm{d}\Omega_1(w_1), \tag{9.2}$$

where $\mathrm{d}\Omega_i(w_i) := \varrho_i\,\mathrm{d}w_i$ and ϱ_i is the probability density function of w_i, with $i = \{1, 2\}$. We can introduce the operators $\mathbb{E}_1$ and $\mathbb{E}_2$ as

$$\mathbb{E}_i[X] = \int_{\Omega_i} X(w_1, w_2)\,\mathrm{d}\Omega_i(w_i), \tag{9.3}$$

where $i = \{1, 2\}$. We remark that $\mathbb{E}_i$ is a random variable in Ω_j, where $i \neq j$. Moreover,

$$\mathbb{E}[X] = \mathbb{E}_1[\mathbb{E}_2[X]] = \mathbb{E}_2[\mathbb{E}_1[X]] = \mathbb{E}_1 \circ \mathbb{E}_2[X]. \tag{9.4}$$

Let $u(t, \boldsymbol{x}, w_1, w_2) := (\boldsymbol{u}(t, \boldsymbol{x}, w_1, w_2), p(t, \boldsymbol{x}, w_1, w_2))$ denote the solution of the NS problem. From the solution field we compute the QoI, that is a process, and reads $\mathrm{Q}(t, w_1, w_2) := \mathrm{Q}(u(t, \boldsymbol{x}, w_1, w_2))$. The estimator we aim at computing is the long-term expected value of $\mathrm{Q}(t, w_1, w_2)$, which reads

$$\overline{\mathrm{Q}} = \lim_{t\to\infty} \mathbb{E}[\mathrm{Q}](t). \tag{9.5}$$

Let us now consider to have *ergodicity* in Ω_1, and not in Ω_2. If we think, for example, about $z_0(w_2)$, it is easy to see that different realizations will not converge to the same value, no matter how long the simulation is.

Now, $\mathbb{E}_1$ can be approximated in two ways, since

$$\mathbb{E}_1[\mathrm{Q}] = \langle \mathrm{Q}(t, w_1, w_2)\rangle = \lim_{T\to\infty} \frac{1}{T}\int_0^T \mathrm{Q}(t, w_1, w_2)\,\mathrm{d}t. \tag{9.6}$$

We can take a finite time interval

$$\mathbb{E}_1[\mathrm{Q}] \approx \langle \mathrm{Q}(t, w_1, w_2)\rangle_{T_0,T} = \frac{1}{T - T_0}\int_{T_0}^T \mathrm{Q}(t, w_1, w_2)\,\mathrm{d}t \tag{9.7}$$

or, as in chapter 8, an ensemble of them

$$\mathbb{E}_1[\mathrm{Q}] \approx \sum_{m=1}^{M} \langle \mathrm{Q}_m\rangle_{T_0,T} = \frac{1}{M}\sum_{m=1}^{M} \frac{1}{T - T_0}\int_{T_0}^T \mathrm{Q}_m\,\mathrm{d}t, \tag{9.8}$$

where $Q_m := Q(t, w_1^{(m)}, w_2)$ are different realizations in w_1 and T_0 is the burn-in time.

The expected value operator *without ergodicity* can be approximated using the MC expected value estimator, and reads

$$\mathbb{E}_2[Q] \approx \frac{1}{N}\sum_{n=1}^{N} Q_n, \tag{9.9}$$

where Q_n are different realizations in w_2, i.e. $Q_n := Q\left(t, w_1, w_2^{(n)}\right)$. Even though we restrict ourselves to using the MC method for approximating the expected value $\mathbb{E}_2$, other hierarchical MC methods can be used. However, we do not go deep into this topic in this thesis.

Finally, the long term expected value can be estimated as

$$\overline{Q} \approx E^{N,M}[\langle Q_{n,m}\rangle_{T_0,T}] = \frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}\langle Q_{n,m}\rangle_{T_0,T}, \tag{9.10}$$

where $Q_{n,m} := Q\left(t, w_1^{(m)}, w_2^{(n)}\right)$.

Remark 2. *The estimation provided by equation* (9.10) *comes from considering* $\boldsymbol{u}_0(\boldsymbol{x}, w_1)$ *(or* $\boldsymbol{u}\prime(w_1)$*) and* $z_0(w_2)$*, but it is not limited to such random variables. It can be generalized to any random variable satisfying ergodicity in* Ω_1 *and any random variable not satisfying ergodicity in* Ω_2*.*

Using this notation, the statistical problem we face is to find the optimal values of N, M, T_0, T (those that minimize the computational cost and/or the time to solution) while satisfying the failure probability convergence criterion (see equation (4.9))

$$\mathbb{P}\left[\left|E^{N,M}[\langle Q_{n,m}\rangle_{T_0,T}] - \overline{Q}\right| > \varepsilon\right] \leq \phi,\ \varepsilon > 0,\ \phi \ll 1, \tag{9.11}$$

where ε and $1-\phi$ are the absolute tolerance and the confidence on the final estimation, respectively.

9.1.2 Error analysis

We analyze now how to estimate equation (9.11), similarly to what we do in section 8.1.2. We assume each realization $Q_{n,m}$ can be described as the sum of a transient perturbation

$A_{n,m}$ and of a statistical steady-state process $S_{n,m}$. Then,

$$\mathrm{Q}_{n,m} = A_{n,m} + S_{n,m}. \tag{9.12}$$

Assuming equation (9.12) implies that $\mathbb{E}_1[A_{n,m}](t) \xrightarrow[t\to\infty]{} 0$, from which it follows that

$$\mathbb{E}_1[\mathrm{Q}_{n,m}](t) \xrightarrow[t\to\infty]{} \mathbb{E}_1[S_{n,m}]. \tag{9.13}$$

By applying the $\mathbb{E}_2$ operator, we obtain

$$\mathbb{E}[\mathrm{Q}_{n,m}](t) \xrightarrow[t\to\infty]{} \mathbb{E}[S_{n,m}] = \overline{\mathrm{Q}}. \tag{9.14}$$

We remark that $\mathbb{E}[A_{n,m}](t)$ and $\mathbb{E}[\mathrm{Q}_{n,m}](t)$ are the expected values of the transient perturbation $A_{n,m}$ and of the QoI $\mathrm{Q}_{n,m}$ at time t.

Initialization bias

First, we evaluate the initialization bias. We can write

$$\begin{aligned}\mathbb{E}\left[\mathrm{E}^{N,M}[\langle \mathrm{Q}_{n,m}\rangle_{T_0,T}]\right] &= \frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}\mathbb{E}\left[\langle \mathrm{Q}_{n,m}\rangle_{T_0,T}\right] \\ &= \frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}\left(\mathbb{E}\left[\langle A_{n,m}\rangle_{T_0,T}\right] + \mathbb{E}\left[\langle S_{n,m}\rangle_{T_0,T}\right]\right).\end{aligned} \tag{9.15}$$

Since $S_{n,m}$ is statistically steady-state,

$$\mathbb{E}[\langle S_{n,m}\rangle_{T_0,T}] = \mathbb{E}\left[\frac{\int_{T_0}^{T} S_{n,m}(t)\,\mathrm{d}t}{T-T_0}\right] = \frac{1}{T-T_0}\int_{T_0}^{T}\mathbb{E}[S_{n,m}]\,\mathrm{d}t = \mathbb{E}[S_{n,m}] = \overline{\mathrm{Q}}. \tag{9.16}$$

Equation (9.15) then becomes

$$\mathbb{E}\left[\mathrm{E}^{N,M}[\langle \mathrm{Q}_{n,m}\rangle_{T_0,T}]\right] = \mathrm{B} + \overline{\mathrm{Q}}, \tag{9.17}$$

where the initialization is

$$\mathrm{B} = \left|\frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}\mathbb{E}\left[\langle A_{n,m}\rangle_{T_0,T}\right]\right|. \tag{9.18}$$

The initialization bias can be neglected if $\langle A_{n,m}\rangle_{T_0,T}$ decays sufficiently fast. In fact, if

$$\int_0^\infty |\mathbb{E}[A_{n,m}](t)| \, \mathrm{d}t < \infty, \tag{9.19}$$

then we have that

$$\begin{aligned} |\mathbb{E}[\langle A_{n,m}\rangle_{T_0,T}]| &= \frac{1}{T-T_0}\left|\int_{T_0}^{T} \mathbb{E}[A_{n,m}](t)\,\mathrm{d}t\right| \le \frac{1}{T-T_0}\int_{T_0}^{T} |\mathbb{E}[A_{n,m}](t)| \, \mathrm{d}t \\ &\le \frac{1}{T-T_0}\int_0^\infty |\mathbb{E}[A_{n,m}](t)| \, \mathrm{d}t \xrightarrow[T-T_0\to\infty]{} 0. \end{aligned} \tag{9.20}$$

In other words, if equation (9.19) holds, then increasing $T-T_0$ eventually decreases |B|.
As in chapter 8, we provide an estimation of the decay rate under stronger assumptions on the transient perturbation. Let us consider the same fast decay of equation (8.14), that we report here for the sake of simplicity

$$A^f_{n,m}(t) = A^f_{0,n,m}\mathrm{e}^{-\frac{t}{\tau}}. \tag{9.21}$$

We recall that its time average is

$$\mathbb{E}[\langle A^f_{n,m}\rangle_{T_0,T}] = \mathbb{E}[A^f_{0,n,m}]\langle \mathrm{e}^{-\frac{t}{\tau}}\rangle_{T_0,T} = \tau\,\mathbb{E}[A^f_{0,n,m}]\left(\frac{\mathrm{e}^{-T_0/\tau}-\mathrm{e}^{-T/\tau}}{T-T_0}\right), \tag{9.22}$$

and therefore the decay behaves as

$$\mathbb{E}[\langle A^f_{n,m}\rangle_{T_0,T}] = \mathcal{O}((T-T_0)^{-1}). \tag{9.23}$$

Therefore, $\mathbb{E}[\langle A^f_{n,m}\rangle_{T_0,T}]$ is a decreasing function of $T-T_0$. We also recall that $\mathbb{E}[\langle A^f_{n,m}\rangle_{T_0,T}]$ decreases when $T-T_0$ is kept constant while T and T_0 separately increase.

Statistical error

Following the same approach of chapter 8, we bound the total error of equation (9.11) with confidence $1-\phi$ as

$$\begin{aligned} \left|\mathrm{E}^{N,M}[\langle \mathrm{Q}_{n,m}\rangle_{T_0,T}] - \overline{\mathrm{Q}}\right| &\le \left|\mathbb{E}\left[\mathrm{E}^{N,M}[\langle \mathrm{Q}_{n,m}\rangle_{T_0,T}]\right] - \overline{\mathrm{Q}}\right| \\ &+ \left|\mathbb{E}\left[\mathrm{E}^{N,M}[\langle \mathrm{Q}_{n,m}\rangle_{T_0,T}]\right] - \mathrm{E}^{N,M}[\langle \mathrm{Q}_{n,m}\rangle_{T_0,T}]\right|, \end{aligned} \tag{9.24}$$

which implies

$$\left|\mathrm{E}^{N,M}[\langle \mathrm{Q}_{n,m}\rangle_{T_0,T}] - \overline{\mathrm{Q}}\right| \leq |\mathrm{B}| + \mathcal{C}_\phi \sqrt{\mathbb{V}\left[\mathrm{E}^{N,M}[\langle \mathrm{Q}_{n,m}\rangle_{T_0,T}]\right]}. \tag{9.25}$$

In equation (9.25), $\mathcal{C}_\phi = \Phi^{-1}(1-\frac{\phi}{2})$ and Φ is the cumulative distribution function of a standard normal random variable. The total error presents two components: a bias term and a SE term, where

$$\mathrm{SE} := \sqrt{\mathbb{V}\left[\frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}\langle \mathrm{Q}_{n,m}\rangle_{T_0,T}\right]}. \tag{9.26}$$

Therefore, for a given confidence $1-\phi$ and tolerance ε, the failure probability criterion of equation (9.11) simplifies to

$$|\mathrm{B}| + \mathcal{C}_\phi \mathrm{SE} \leq \varepsilon. \tag{9.27}$$

We focus now on the SE term. Assuming $A_{n,m}$ and $S_{n,m}$ to be independent, we can write

$$\begin{aligned}
&\mathbb{V}\left[\frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}\langle \mathrm{Q}_{n,m}\rangle_{T_0,T}\right] \\
&= \frac{1}{N^2}\sum_{n=1}^{N}\mathbb{V}\left[\frac{\sum_{m=1}^{M}\langle A_{n,m}\rangle_{T_0,T}}{M}\right] + \frac{1}{N^2}\sum_{n=1}^{N}\mathbb{V}\left[\frac{\sum_{m=1}^{M}\langle S_{n,m}\rangle_{T_0,T}}{M}\right] \\
&+ \frac{1}{N^2}\sum_{\substack{n,k=1\\ n\neq k}}^{N}\mathrm{co}\mathbb{V}\left[\frac{\sum_{m=1}^{M}\langle \mathrm{Q}_{n,m}\rangle_{T_0,T}}{M}, \frac{\sum_{m=1}^{M}\langle \mathrm{Q}_{k,m}\rangle_{T_0,T}}{M}\right].
\end{aligned} \tag{9.28}$$

For the second term of the right hand side of equation (9.28), we have

$$\begin{aligned}
\mathbb{V}\left[\frac{\sum_{m=1}^{M}\langle S_{n,m}\rangle_{T_0,T}}{M}\right] &= \frac{1}{M^2}\sum_{m=1}^{M}\mathbb{V}\left[\langle S_{n,m}\rangle_{T_0,T}\right] + \frac{1}{M^2}\sum_{\substack{m,k=1\\ m\neq k}}^{M}\mathrm{co}\mathbb{V}\left[\langle S_{n,m}\rangle_{T_0,T}, \langle S_{n,k}\rangle_{T_0,T}\right] \\
&= \frac{\mathbb{V}\left[S_{n,m}\right]}{M(T-T_0)^2}\int_{T_0}^{T}\int_{T_0}^{T}\rho(s-t)\,\mathrm{d}s\,\mathrm{d}t + \frac{1}{M^2}\sum_{\substack{m,k=1\\ m\neq k}}^{M}\mathrm{co}\mathbb{V}\left[\langle S_{n,m}\rangle_{T_0,T}, \langle S_{n,k}\rangle_{T_0,T}\right],
\end{aligned} \tag{9.29}$$

where ρ is the autocorrelation function, see [112, section 3.6]. As demonstrated in [112, problem 3.37], the long-time limit of the first term of the right hand side of equation (9.29) is $2\frac{\mathbb{V}[S_{n,m}]\overline{\tau}}{M(T-T_0)}$, where $\overline{\tau}$ is the integral timescale of the process [112, section 3.6], which is a correlation constant associated to the QoI.

The last term in equation (9.29) depends on the correlation between realizations with the same $w_2^{(n)}$. If these realizations are independent, the final term in equation (9.29) is negligible. To this end, we exploit the two initial condition strategies introduced in section 8.1.3.

The first term on the right-hand side of equation (9.28) cannot be estimated without making assumptions on the behavior of the transient perturbation A. If we consider the fast decay of equation (9.21), a straightforward computation shows that $\mathbb{V}[\langle A^f_{n,m}\rangle_{T_0,T}] = \mathcal{O}((T-T_0)^{-2})$.

If the decay of the transient perturbation is slower than equation (9.21), the decay of the statistical error will be dominated by the first and third terms in equation (9.28). On the other hand, if the decay of the transient perturbation is fast (equation (9.21) holds), the overall error is dominated by the second and third terms in equation (9.28), and such an equation becomes

$$\begin{aligned}\mathbb{V}&\left[\frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}\langle \mathrm{Q}_{n,m}\rangle_{T_0,T}\right]\\ &= \frac{1}{N^2}\sum_{n=1}^{N}2\frac{\mathbb{V}[S_{n,m}]\overline{\tau}}{M(T-T_0)} + \frac{1}{N^2}\sum_{\substack{n,k=1\\ n\neq k}}^{N}\mathrm{co}\mathbb{V}\left[\left[\sum_{m=1}^{M}\frac{\langle \mathrm{Q}_{n,m}\rangle_{T_0,T}}{M}\right],\left[\sum_{m=1}^{M}\frac{\langle \mathrm{Q}_{k,m}\rangle_{T_0,T}}{M}\right]\right].\end{aligned} \tag{9.30}$$

The first term of the right hand side of equation (9.30) decays as $\frac{c_0^2}{NM(T-T_0)}$, for some constant c_0. The covariance term of the right hand side of equation (9.30) behaves as $\frac{c^2}{N}$, for some constant c. In this fast decay scenario, the SE decays as $\sqrt{\frac{c_0^2}{NM(T-T_0)}+\frac{c^2}{N}}$, that is bounded by

$$\frac{\frac{c_0}{M^{0.5}(T-T_0)^{0.5}}+c}{N^{0.5}}. \tag{9.31}$$

Such a result suggests that we cannot exactly estimate $\overline{\mathrm{Q}}$ by only increasing M or $T-T_0$, but we need $N\to\infty$ to have null SE. This is intuitively true, since we cannot describe all possible scenarios by only increasing M or $T-T_0$ (that is related with Ω_1), but we need to explore different stochastic (and non-ergodic) scenarios by sampling on Ω_2, that is increasing N.

The dominant term of equation (9.27) depends on the decay rate of the transient perturbations $A_{n,m}$. If the transient perturbation decays fast enough, equation (9.27) simplifies to

$$\mathcal{C}_\phi \mathrm{SE} \leq \varepsilon, \tag{9.32}$$

which can be estimated as

$$\mathcal{C}_\phi \sqrt{\mathrm{V}^K \left[\frac{1}{NM} \sum_{n=1}^{N} \sum_{m=1}^{M} \langle \mathrm{Q}_{n,m} \rangle_{T_0,T} \right]} \leq \varepsilon, \tag{9.33}$$

with K sufficiently large. On the other hand, if the decay of the transient perturbation is slower than equation (9.21), the left-hand side of equation (9.27) will decay like $(T - T_0)^{-q}$ for some $q < 1$.

In the numerical experiments of section 9.2, we verify that the initialization bias decays fast enough with respect to the burn-in time T_0, and therefore that equations (9.31) and (9.32) hold.

9.1.3 Algorithm

We report in algorithm 7 the ensemble-based synchronous MC method and in algorithm 8 the ensemble-based asynchronous MC method. To simplify comparisons with the synchronous and asynchronous MC algorithms (see algorithms 1 and 3), h-statistics are used to estimate central moments and we omit to specify level $l = 0$.

As commented in sections 4.7 and 5.2, the update of the number of batches B, of the number of levels L and of the number of samples per level N can be adaptive or can be only function of the iteration counter *it* and of their previous values. We define $S_{l,p}$ as the power sum of level l and power p, where $p \in [1, P]$, P is the maximum order we need and the number of realizations N is omitted when there is no risk of misunderstanding. Concerning the asynchronous algorithm, we define $S_{b,l,p}$ as the local power sum of batch b, level l and power p, where $p \in [1, P]$, P is the maximum order we need and the number of realizations N is again omitted wherever not necessary. The global power sum of level l and order p is defined as $S_{G,l,p}$. The h-statistic of level l and power p is defined as $h_{l,p}$. We recall that $\mathrm{Q}_{n,m} = \mathrm{Q}\left(t, w_1^{(m)}, w_2^{(n)}\right)$. The left horizontal arrow denotes the update or the computation of the left value as a function of the right values.

Algorithm 7 Ensemble-based synchronous MC

```
  $N, H$ initial hierarchy
  while convergence is not True do
    if non-adaptive then
      $N \longleftarrow N, it$
    else if adaptive then
      $N \longleftarrow N, it, h_p,\ p \in [1, P]$
    end if
    for $n = 0 : N$ do
      for $m = 0 : M$ do
        $\mathrm{Q}_{n,m} \longleftarrow \text{solver}\left(w_1^{(m)}, w_2^{(n)}\right)$
      end for
      $\mathrm{Q}_n \longleftarrow \mathrm{Q}_{n,m}M,\ m \in [1, M]$
      $S_p^n \longleftarrow S_p^{n-1}, \mathrm{Q}_n, n, p,\ p \in [1, P]$
    end for
    $h_p \longleftarrow S_p, N,\ p \in [1, P]$
    convergence $\longleftarrow$ equation (4.9)
    $it = it + 1$
  end while
```

Algorithm 8 Ensemble-based asynchronous MC

```
  $B, N, H$ initial hierarchy
  while convergence is not True do
    $B \longleftarrow B, it$
    $N \longleftarrow N, it$
    for $b = 0 : B$ do
      for $n = 0 : N_b$ do
        for $m = 0 : M$ do
          $\mathrm{Q}_{n,m} \longleftarrow \text{solver}\left(w_1^{(m)}, w_2^{(n)}\right)$
        end for
        $\mathrm{Q}_n \longleftarrow \mathrm{Q}_{n,m}M,\ m \in [1, M]$
        $S_{b,p}^n \longleftarrow S_{b,p}^{n-1}, \mathrm{Q}_n, n, p,\ p \in [1, P]$
      end for
      $N = N + N_b$
      $S_{G,p} = S_{G,p} + S_{b,p},\ p \in [1, P]$
    end for
    $h_p \longleftarrow S_{G,p}, N,\ p \in [1, P]$
    convergence $\longleftarrow$ equation (4.9)
    $it = it + 1$
  end while
```

9.2 Numerical experiments

In this section, we verify the statistical framework introduced in section 9.1 by solving three problems of engineering interest, where all systems consider stochastic wind mean profiles $\overline{\boldsymbol{u}}$. First, the problem of wind flowing around a rectangle obstacle is presented in section 9.2.1. Then, wind flowing around the CAARC building is considered under two different scenarios: one with constant in time boundary conditions (section 9.2.2) and one with realistic conditions, where wind fluctuations $\boldsymbol{u}'$ are not null (section 9.2.3).

It is known that strong winds in a storm usually last one hour or ten minutes[1] [121]. Therefore, in this section, we consider a time span of ten minutes is enough to describe wind fluctuations and peaks. Such a time window must be considered for all wind profiles $\overline{\boldsymbol{u}}$.

For solving the CFD problems, we use Kratos [44, 43] as FE solver software, XMC [13] as hierarchical MC library and PyCOMPSs [16, 88, 126] as programming model for distributed computing. The integration of these software has been an important part of this thesis work, as documented in [127, 10, 11, 5, 4, 22].

The analyses are run on Salomon. This cluster presents 2 Petaflops of peak performance and is made by 1008 compute nodes. Each node is a powerful x86-64 computer equipped with 24 cores (two 2.5 GHz twelve-core Intel Xeon processors) and 128 GB RAM.

We remark that, when there is no risk of misunderstanding, we replace $\langle \mathrm{Q} \rangle_{T_0,T}$ with $\langle \mathrm{Q} \rangle$. Moreover, international units are used to measure physical quantities.

9.2.1 Rectangle obstacle problem

We remark that the rectangle obstacle problem is analyzed and solved with different configurations. In section 7.1, constant in time stochastic boundary conditions are considered, and the problem is solved with hierarchical MC methods. In section 8.2.1, constant in time deterministic boundary conditions are considered, and the problem is solved with the standard time averaging method and the ensemble averaging method. In section 9.2.1, constant in time stochastic boundary conditions are considered, and the problem is solved with the ensemble-based MC method. Therefore, in this section we use constant in time stochastic boundary conditions.

[1]For example, in [121, section 2.4.1] the authors comment that "the averaging time ... should be equal to the duration of strong winds in a storm. Typical durations being considered are 1 hour, and 10 minutes" or in [121, section 2.1] they comment "wind speeds averaged over 10 minutes ... are used in World Meteorological Organization (WMO) practice as well as in some standards and codes".

Problem formulation

We consider the two-dimensional flow around a rectangle body [28], introduced in section 7.1. The domain is reported in figure 7.1 and the system is described in section 7.1. The flow is modeled by the incompressible NS equations (see equation (2.1)) and we recall that slip boundary conditions are applied on the external boundaries and no-slip boundary conditions on the rectangle body. Standard air density and viscosity are considered and the Re is 132719. The wind inlet is uniform and steady-state on the whole inlet and is modeled as $\overline{\boldsymbol{u}} \sim \mathcal{N}(2, 0.02)$.

Even though more observables are computed, the QoI for which we assess convergence is the drag force F_d. Therefore, we set $\mathrm{Q} \equiv F_d$. The confidence $1-\phi$ and the tolerance ε of equation (4.9) are 99% and 0.04, respectively. The relative value of the tolerance with respect to sample mean, that is $\frac{\mathcal{C}_\phi \mathrm{SE}}{\mathrm{E}^{N,M}[\langle F_d \rangle]}$, is below 1.2%.

An adaptive mesh refined with respect to the time-averaged velocity field $\langle \boldsymbol{u}(t,x) \rangle_{T_0,T}$ is used. The mesh has approximately 25 thousand nodes and a minimal size, close to the rectangle body, of 0.002 m. The chosen time step is 0.02 s, which gives a CFL number of 20. The mesh and the solver are validated in section 7.1.2.

Results

In this subsection, we present the rectangle obstacle problem results. We recall that the instantaneous velocity and pressure fields at $t = 200\,\mathrm{s}$ for one realization can be observed in figure 7.5 and figure 7.6. The three key parameters we play with are the number of wind scenarios N, the ensemble size per wind realization M and the time length T of each simulation. We shall remark that the computational cost of a simulation is governed by the total simulation time T, rather than by the effective time $T - T_0$. For this reason, the overall computational cost NMT is kept constant across all different configurations. We recall that a time span of 10 minutes is considered for all wind scenarios, that is $M(T - T_0) \approx 600\,\mathrm{s}$.

First, we try to make some estimations on the initialization bias for different N, $T - T_0$ and M values. We know from equation (9.31) that the SE decays as $N^{-0.5}$ if the transient perturbation decays fast. We consider a burn-in time $T_0 = 40\,\mathrm{s}$, that is estimated in section 8.2.1. We remark that such a T_0 is the time the flow needs to go from the inlet to the body for an average velocity of $2\,\mathrm{m\,s^{-1}}$. We can observe in figures 9.1–9.3 that the SE fits well equation (9.31). Therefore, the initialization bias is negligible and equation (9.32) can be used to assess the failure probability convergence criterion.

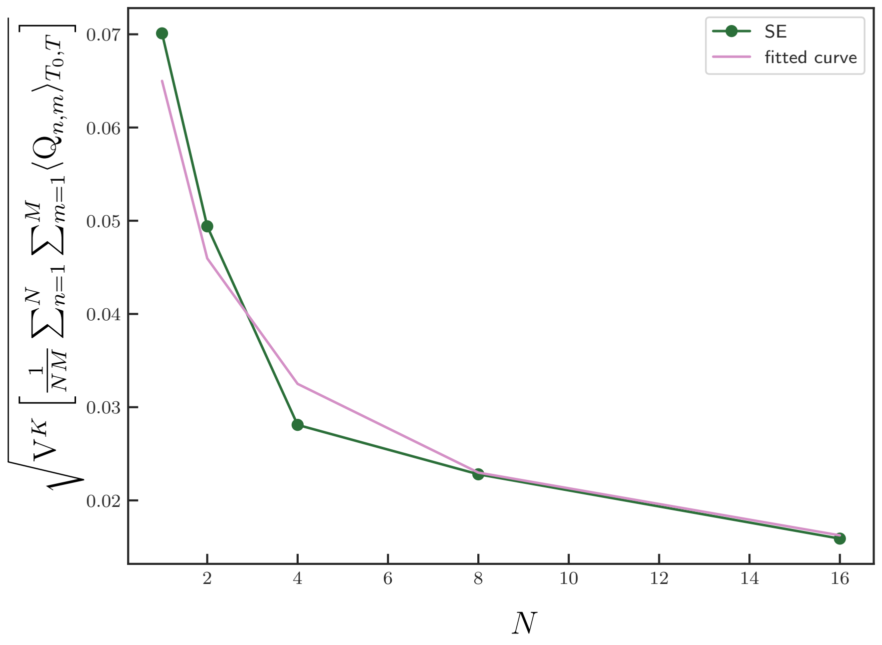


Figure 9.1: Computation of SE as function of N for the drag force $Q \equiv F_d$. $M = 10$ and $T - T_0 = 256\,\text{s}$. The fitted curve is equation (9.31).

In table 9.1 we report the solution of the stochastic problem. All the simulations present the same total time, which is given by considering NMT constant, and run a total of 2 iterations, that means the convergence criterion is checked twice and is verified only the second time. We can conclude that using $M > 1$ provides important time to solution reductions, at the price of a slightly larger SE, for the same computational cost. It is worth remarking that, for the same computational cost, the SE increases as M does, since the number of transient perturbations to discard grows and therefore the overall effective time decreases. On the other hand, one could keep the SE constant by keeping $M(T - T_0) = 600\,\text{s}$ and therefore increasing the computational cost as M grows.

For the sake of completeness, we consider as well the case in which the constraint $M(T - T_0) \approx 600\,\text{s}$ can be relaxed. We present in table 9.2 results for the case in which no constraints on $M(T - T_0)$ are considered and only the product NMT is kept constant. As expected from equation (9.31), increasing N drastically reduces SE.

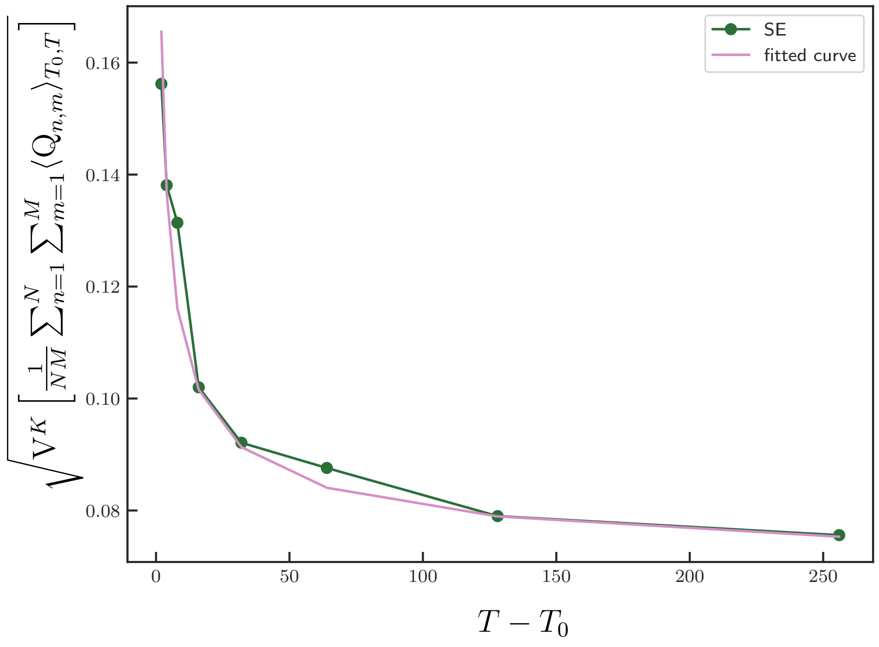


Figure 9.2: Computation of SE as function of $T - T_0$ for the drag force $Q \equiv F_d$. $K = 96$, $N = 1$ and $M = 10$. The fitted curve is equation (9.31).

$E^{N,M}[\langle F_d\rangle]$	$\frac{\mathcal{C}_\phi \text{SE}}{E^{N,M}[\langle F_d\rangle]}$	N	M	$T - T_0$	T_0	C	time to solution
3.234250	0.84%	48	1	600	40	7171	33.20
3.250853	0.95%	48	2	280	40	6844	16.77
3.243490	1.02%	48	5	88	40	6708	6.82
3.207608	1.11%	48	10	24	40	6732	3.46

Table 9.1: The table reports the drag force F_d expected value estimation and its associated SE, with a 99% confidence. N, M, $T - T_0$ and T_0 refer to the number of wind realizations, the number of ensembles per wind scenario, the effective time window and the burn-in time, respectively. C is the computational cost, expressed in CPU hours. Time to solution is the real time we need to wait for solving the problem, and it is expressed in hours. Results are sorted for decreasing time to solution.

Other observables

We select the case with $N = 48$, $M = 1$, $T - T_0 = 600\,\text{s}$ to show statistical estimators of all the observables we compute, that are the drag force, the pitching moment (computed

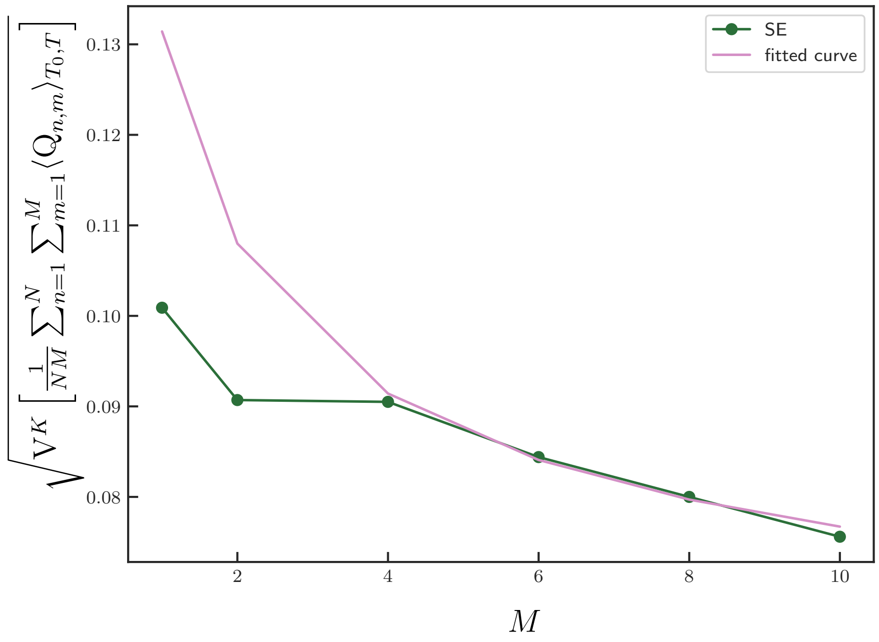


Figure 9.3: Computation of SE as function of M for the drag force $Q \equiv F_d$. $K = 96$, $N = 1$ and $T - T_0 = 256\,\mathrm{s}$. The fitted curve is equation (9.31).

$E^{N,M}[\langle F_d\rangle]$	$\frac{\mathcal{C}_\phi \mathrm{SE}}{E^{N,M}[\langle F_d\rangle]}$	N	M	$T-T_0$	T_0	C	time to solution
3.234250	0.84%	48	1	600	40	7171	33.20
3.225962	0.73%	480	1	24	40	3173	1.63

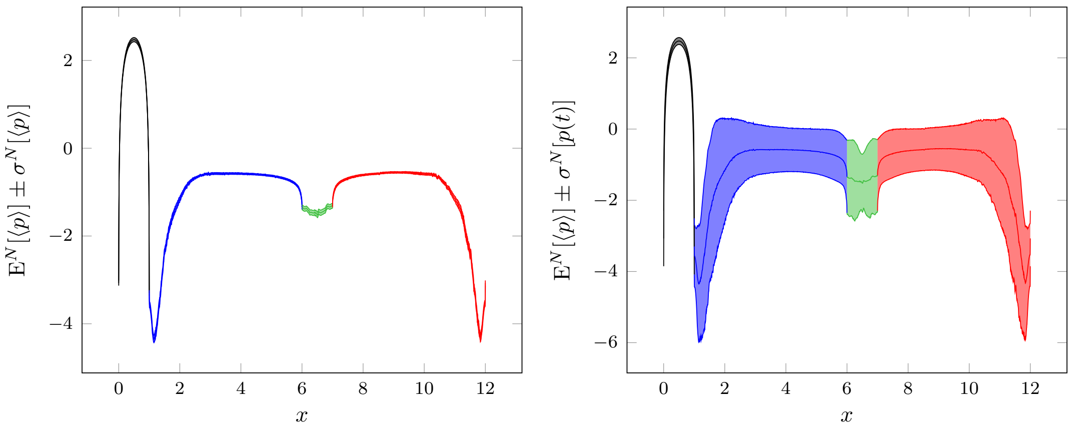
Table 9.2: The table reports the drag force F_d expected value estimation and its associated SE, with a 99% confidence. N, M, $T-T_0$ and T_0 refer to the number of wind realizations, the number of ensembles per wind scenario, the effective time window and the burn-in time, respectively. C is the computational cost, expressed in CPU hours. Time to solution is the real time we need to wait for solving the problem, and it is expressed in hours. Results are sorted for decreasing time to solution.

around the center of the rectangle body) and the pressure field around the rectangle body. Apart from the expected value, we estimate the standard deviation and its time average, which are denoted as $\sigma[Q]$ and $\sigma[\langle Q\rangle_{T_0,T}]$, respectively. $\sigma[Q]$ is an indicator of the distribution around the expected value, while $\sigma[\langle Q\rangle_{T_0,T}]$ is an error indicator of the mean estimation. We also point out that expected value $\pm$ standard deviation is a common risk measure often used in engineering decision processes (see [115] and section 3.7).

Table 9.3 shows expected value and standard deviation estimations of the drag force and the pitching moment. Figures 9.4a and 9.4b show expected value and standard deviation estimations of the pressure field.

Q	$\mathrm{E}^N[\langle \mathrm{Q}\rangle_{T_0,T}]$	$\sigma^N[\langle \mathrm{Q}\rangle_{T_0,T}]$	$\sigma^N[\mathrm{Q}]$
F_d	3.234250	0.0807	0.5780
M_p	-0.023559	0.0545	2.1309

Table 9.3: Statistical analysis of the drag force F_d and of the pitching moment M_p around the rectangle center.



(a) Mean and standard deviation of the time-averaged pressure field $\langle p(\boldsymbol{x})\rangle_{T_0,T}$. Each color represents one side of the rectangle and they are defined in figure 7.1.

(b) Mean of the time-averaged pressure field $\langle p(\boldsymbol{x})\rangle_{T_0,T}$ and standard deviation of the pressure field $p(\boldsymbol{x})$. Each color represents one side of the rectangle and they are defined in figure 7.1.

Figure 9.4

Finally, we report in table 9.4 the CVaR results for the time-averaged drag force and the drag force.

9.2.2 High-rise building problem with constant in time wind

We remark that the CAARC building problem is analyzed and solved with different configurations. In section 7.2, turbulent fluctuations around a fixed mean wind field boundary conditions are considered, and the problem is solved with hierarchical MC methods.

Q	CVaR	α
$\langle F_d \rangle_{T_0,T}$	3.36661	0.9
F_d	4.37460	0.9

Table 9.4: CVaR analysis of time-averaged drag force $\langle F_d \rangle_{T_0,T}$ and drag force F_d. Results for $N = 48$, $M = 1$, $T - T_0 = 600\,\mathrm{s}$ and $\alpha = 0.9$.

In section 8.2.2, constant in time fixed mean wind field boundary conditions are considered, and the problem is solved with the standard time averaging method and the ensemble averaging method. In section 9.2.2, constant in time stochastic mean wind field boundary conditions are considered, and the problem is solved with the ensemble-based MC method. In section 9.2.3, turbulent fluctuations around a stochastic mean wind field boundary conditions are considered, and the problem is solved with the ensemble-based MC method. Therefore, in this section we use constant in time stochastic mean wind field boundary conditions.

Problem formulation

We solve the wind flow past the CAARC building, that is introduced in section 7.2. The system is described by the NS equations (see equation (2.1)), slip boundary conditions are applied on the walls and the ceiling, no-slip boundary conditions on the building and on the floor.

We consider null fluctuations $\boldsymbol{u}' \equiv 0$ and a stochastic roughness height, modeled as $z_0 \sim \mathrm{Unif}(0.1, 0.7)$, where $\mathrm{Unif}(a, b)$ denotes a uniform distribution with lower bound a and upper bound b. This scenario is typical of sparsely built-up urban areas, suburbs and wooded areas [73]. Apart from the roughness height, other physical properties are reported in table 7.4. This gives a Re of around 119 millions, computed with a characteristic length of 45 m.

The observables we compute are the drag force F_d on the body, the base moment M_b around the center of the base on the body and the pressure field p on the body surface. The QoI for which we assess the failure probability criterion of equation (9.11) is the drag force F_d. Therefore, we set $\mathrm{Q} \equiv F_d$. The selected tolerance and confidence for solving the stochastic problem are $\varepsilon = 110000$ and $1 - \phi = 99\%$. The relative value of the tolerance, with respect to the drag force mean value, is 1.15%.

The mesh we use is adaptive with respect to a metric built on top of velocity and pressure fields and has approximately 312000 nodes. The minimal size, close to the body, is 0.2 m and the CFL is 100. We refer to section 8.2.2 for the validation.

Results

First, we verify that for $T_0 = 30\,\mathrm{s}$ the initialization bias is negligible, where $T_0 = 30\,\mathrm{s}$ is the optimal burn-in time we obtain in section 8.2.2. Since figures 9.5–9.7 show that equation (9.31) holds, equation (9.27) simplifies to equation (9.32).

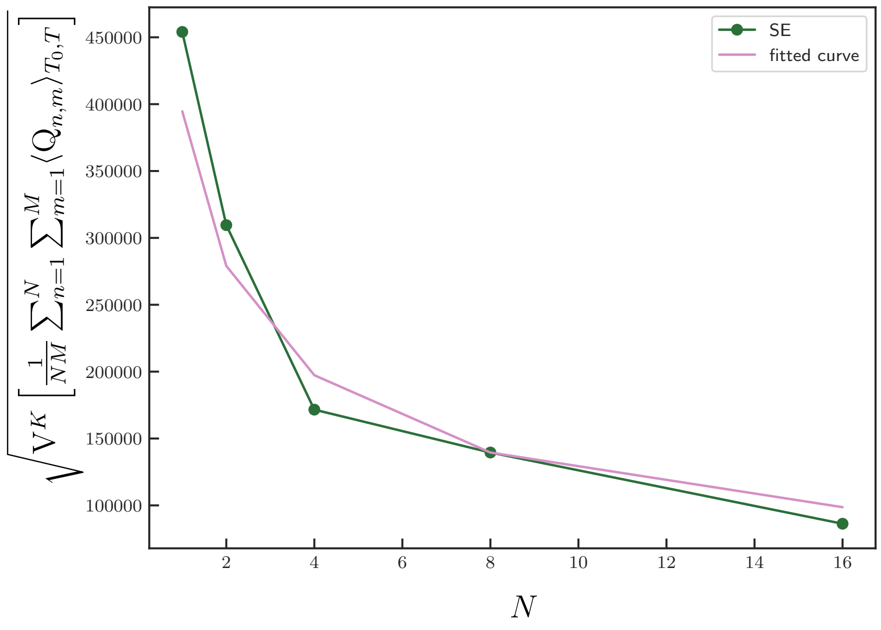


Figure 9.5: Computation of SE as function of N for the drag force $\mathrm{Q} \equiv F_d$. $M = 1$ and $T - T_0 = 48.75\,\mathrm{s}$. The fitted curve is equation (9.31).

Then, we solve the problem for a fixed total time NMT and we report results in table 9.5. We remark that the product $M(T - T_0)$ is approximately equal to 600 s for each wind scenario. We observe that exploiting multiple ensembles ($M > 1$) reduces the wall clock time, at the price of a slightly larger SE, for the same computational cost. Larger SE values appear due to the fact that the effective time is smaller as M grows.

We present as well results where only the product NMT is kept constant and the constraint $M(T - T_0) \approx 600\,\mathrm{s}$ is relaxed in table 9.6.

We report in figure 9.8 and figure 9.9 the instantaneous velocity and pressure fields at

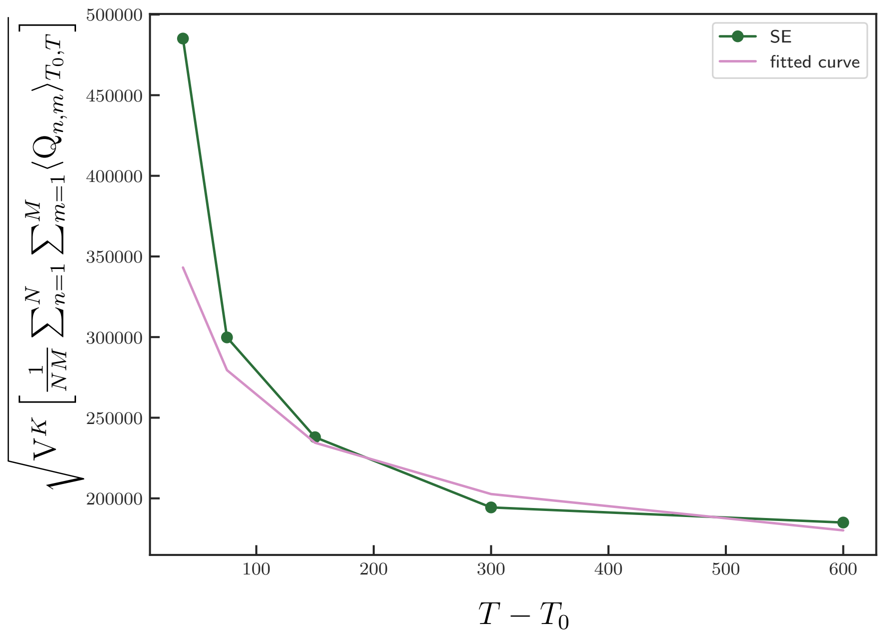


Figure 9.6: Computation of SE as function of $T - T_0$ for the drag force $\mathrm{Q} \equiv F_d$. $K = 20$, $N = 1$ and $M = 1$. The fitted curve is equation (9.31).

$\mathrm{E}^{N,M}[\langle F_d\rangle]$	$\frac{\mathcal{C}_\phi \mathrm{SE}}{\mathrm{E}^{N,M}[\langle F_d\rangle]}$	N	M	$T - T_0$	T_0	C	time to solution
9632478	0.97%	20	1	600	30	10468	20.77
9748082	1.26%	20	2	285	30	10087	10.25
9652424	1.00%	20	4	127.5	30	10263	5.27
9598793	1.12%	20	8	48.75	30	10364	2.68

Table 9.5: The table reports the drag force F_d expected value estimation and its associated SE, with a 99% confidence. N, M, $T - T_0$ and T_0 refer to the number of wind realizations, the number of ensembles per wind scenario, the effective time window and the burn-in time, respectively. C is the computational cost, expressed in CPU hours. Time to solution is the real time we need to wait for solving the problem, and it is expressed in hours. Results are sorted for decreasing time to solution.

$t = 200\,\mathrm{s}$ for one realization. We point out that the main difference with respect to figures 8.10 and 8.11 is the roughness height, that here is equal to 0.3 m.

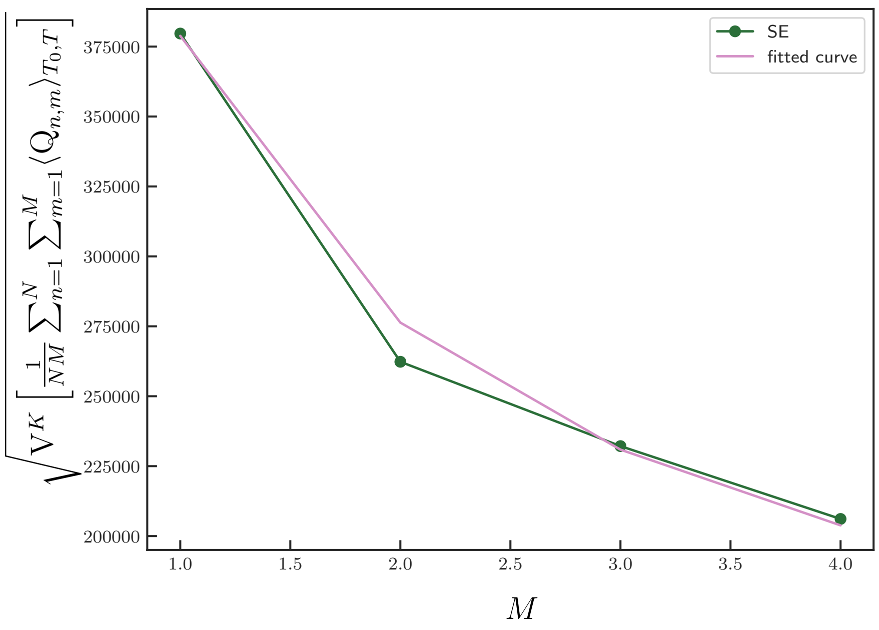


Figure 9.7: Computation of SE as function of M for the drag force $Q \equiv F_d$. $K = 40$, $N = 1$ and $T - T_0 = 75\,\text{s}$. The fitted curve is equation (9.31).

$E^{N,M}[\langle F_d\rangle]$	$\frac{\mathcal{C}_\phi \text{SE}}{E^{N,M}[\langle F_d\rangle]}$	N	M	$T - T_0$	T_0	C	time to solution
9632478	0.97%	20	1	600	30	10468	20.77
9576676	0.82%	160	1	48.75	30	10319	2.67

Table 9.6: The table reports the drag force F_d expected value estimation and its associated SE, with a 99% confidence. N, M, $T-T_0$ and T_0 refer to the number of wind realizations, the number of ensembles per wind scenario, the effective time window and the burn-in time, respectively. C is the computational cost, expressed in CPU hours. Time to solution is the real time we need to wait for solving the problem, and it is expressed in hours. Results are sorted for decreasing time to solution.

Other observables

We select the case with $N = 20$, $M = 1$, $T - T_0 = 600\,\text{s}$ to compute drag force, base moment (computed around the center of the bottom of the building) and pressure field around the building statistical estimators. Table 9.7 shows expected value and standard deviation estimations of the drag force and of the base moment. Figures 9.10 and 9.11

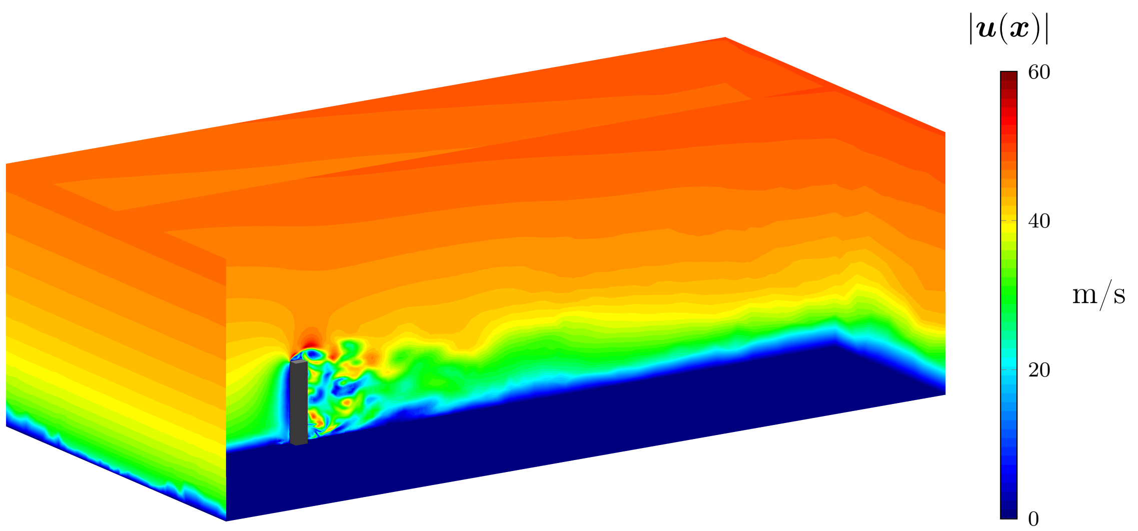


Figure 9.8: Velocity field snapshot at $t = 200\,\mathrm{s}$.

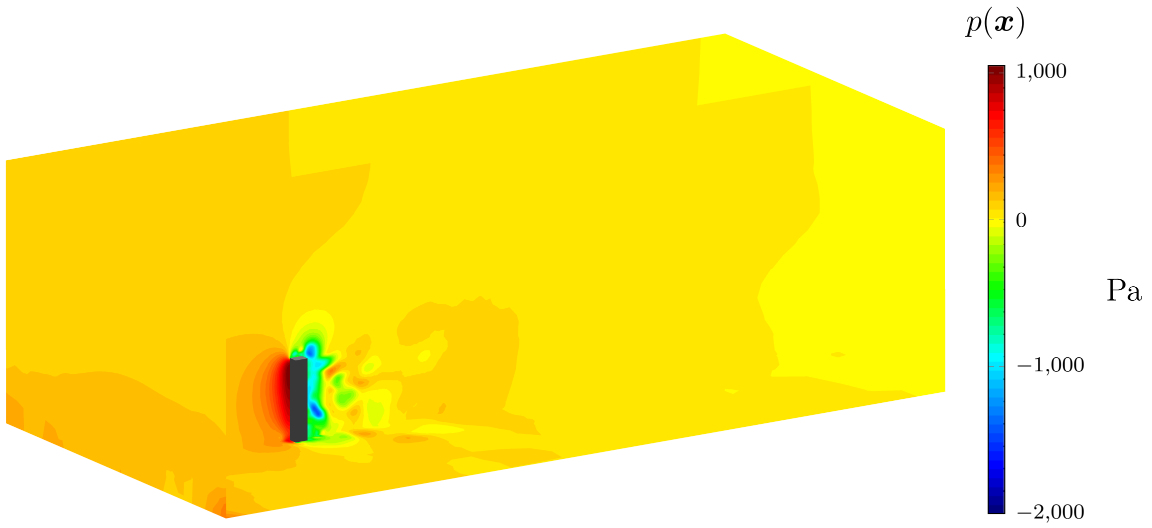


Figure 9.9: Pressure field snapshot at $t = 200\,\mathrm{s}$.

show expected value and standard deviation estimations of the pressure field.

Q	$\mathrm{E}^N[\langle\mathrm{Q}\rangle_{T_0,T}]$	$\sigma^N[\langle\mathrm{Q}\rangle_{T_0,T}]$	$\sigma^N[\mathrm{Q}]$
F_d	9632478	180333	783596
M_b	-3603	293732	7990526

Table 9.7: Statistical analysis of the drag force F_d and of the base moment M_b, computed around the center of the CAARC building base.

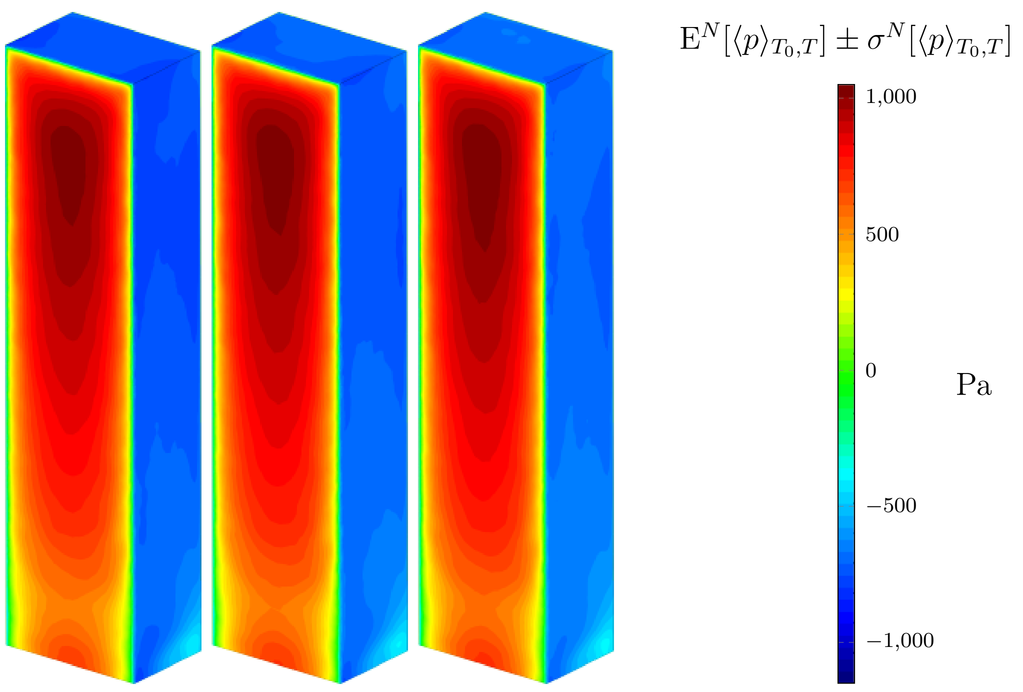

$E^N[\langle p\rangle_{T_0,T}] \pm \sigma^N[\langle p\rangle_{T_0,T}]$
1,000
500
0
−500
−1,000
Pa

Figure 9.10: Statistical analysis of the pressure field $\langle p(x)\rangle_{T_0,T}$. From left to right, $E^N[\langle p(x)\rangle_{T_0,T}] - \sigma^N[\langle p(x)\rangle_{T_0,T}]$, $E^N[\langle p(x)\rangle_{T_0,T}]$ and $E^N[\langle p(x)\rangle_{T_0,T}] + \sigma^N[\langle p(x)\rangle_{T_0,T}]$.

We report in table 9.8 the CVaR results for the time-averaged drag force and the drag force.

Q	CVaR	α
$\langle F_d\rangle_{T_0,T}$	9942443	0.9
F_d	11260707	0.9

Table 9.8: CVaR analysis of time-averaged drag force $\langle F_d\rangle_{T_0,T}$ and drag force F_d. Results for $N = 20$, $M = 1$, $T - T_0 = 600\,\mathrm{s}$ and $\alpha = 0.9$.

9.2.3 High-rise building problem with realistic wind

We remark that the CAARC building problem is analyzed and solved with different configurations. In section 7.2, turbulent fluctuations around a fixed mean wind field boundary conditions are considered, and the problem is solved with hierarchical MC methods. In section 8.2.2, constant in time fixed mean wind field boundary conditions are considered, and the problem is solved with the standard time averaging method and the

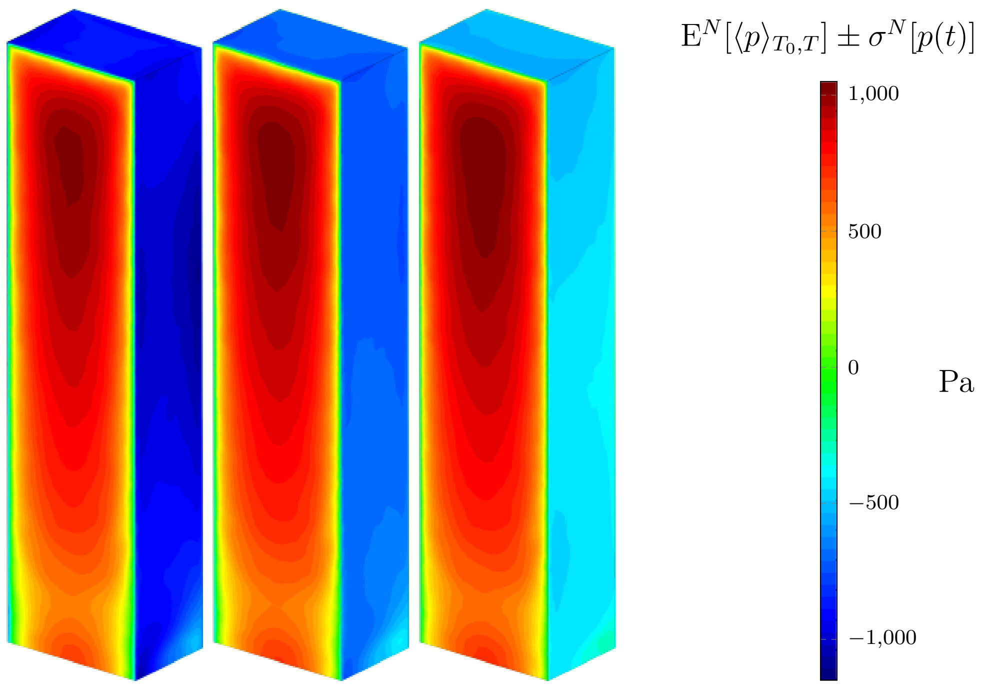


Figure 9.11: Statistical analysis of the pressure field $p(x)$. From left to right, $\mathrm{E}^N[\langle p(x)\rangle_{T_0,T}] - \sigma^N[p(x)]$, $\mathrm{E}^N[\langle p(x)\rangle_{T_0,T}]$ and $\mathrm{E}^N[\langle p(x)\rangle_{T_0,T}] + \sigma^N[p(x)]$.

ensemble averaging method. In section 9.2.2, constant in time stochastic mean wind field boundary conditions are considered, and the problem is solved with the ensemble-based MC method. In section 9.2.3, turbulent fluctuations around a stochastic mean wind field boundary conditions are considered, and the problem is solved with the ensemble-based MC method. Therefore, in this section we use turbulent fluctuations around a stochastic mean wind field boundary conditions.

Problem formulation

We solve the wind flow past the CAARC building, that is introduced in section 7.2 and whose domain can be observed in figure 7.8. The system is described by the NS equations (see equation (2.1)), slip boundary conditions are applied on the walls and the ceiling, no-slip boundary conditions on the building and on the floor.

We consider both fluctuations $\boldsymbol{u}'$, described by the Mann model (see section 2.4), and a stochastic roughness height, model as $z_0 \sim \mathrm{Unif}(0.1, 0.7)$. This scenario is typical of sparsely built-up urban areas, suburbs and wooded areas [73]. Apart from the roughness height, other physical properties are reported in table 7.4. This gives a Re of around

119 millions, computed with a characteristic length of 45 m.

The QoI for which we assess the failure probability criterion is the drag force, therefore we set $\mathrm{Q} \equiv F_d$. The tolerance and confidence of the convergence criterion are $\varepsilon = 110000$ and $1 - \phi = 99\%$, respectively. The relative value of the tolerance, with respect to the drag force mean value, is around 1.10%.

The mesh we use is adaptively refined with respect to pressure and velocity fields. It presents around 283 thousand nodes and a minimal size of 0.2 m close to the building. The chosen time step is 0.2375 s, which gives a CFL of 100. Such a configuration is validated in section 7.2.2.

Results

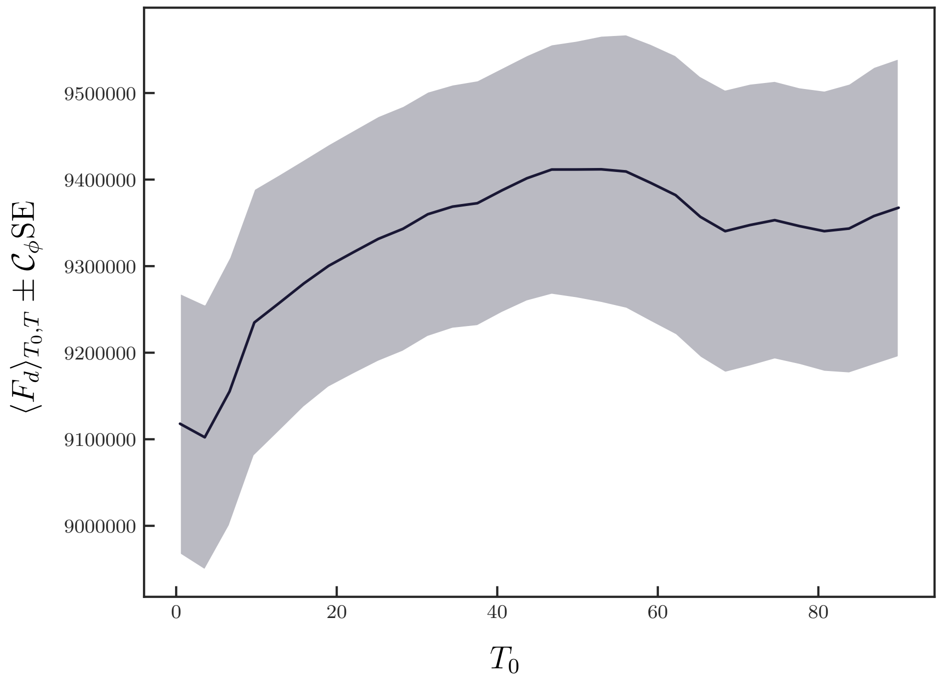


Figure 9.12: Expected value estimation and associated SE with 99% confidence plotted as function of the burn-in time, with $M = 128$. The effective time $T - T_0 = 60$ s is fixed among all realizations and all burn-in times.

We estimate the burn-in time following the approach presented in chapter 8, which estimates T_0 on top of statistical and physical constraints. Figure 9.12 shows that statistical results of the QoI are insensitive for $T_0 > 30$ s, that is larger than the time required by the wind to go from the inlet to the building.

We verify now that for $T_0 = 30\,\mathrm{s}$ equation (9.31) is valid in figures 9.13–9.15. Consequently, equation (9.11) can be estimated via equation (9.32) and the initialization bias is negligible.

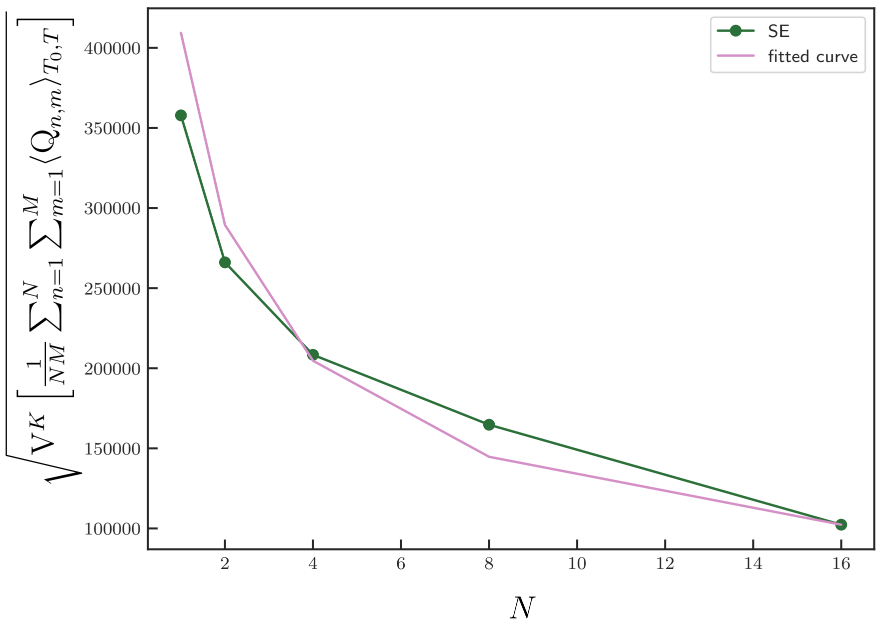


Figure 9.13: Computation of SE as function of N for the drag force $Q \equiv F_d$. $M = 1$ and $T - T_0 = 75\,\mathrm{s}$. The fitted curve is equation (9.31).

Results for a constant total time NMT and $M \approx (T - T_0) \approx 600\,\mathrm{s}$ are presented in table 9.9. We observe that the last case of the table drastically reduces the time to solution, at the price of a slightly larger SE, for the same computational cost. Therefore, $M > 1$ is the most promising configuration.

We consider also the scenario where the constraint $M(T - T_0) \approx 600\,\mathrm{s}$ is relaxed and only NMT is kept constant. Results are reported in table 9.10. We observe that both SE and time to solution are drastically reduced by increasing N and reducing M and $T - T_0$.

We report in figure 9.16 and figure 9.17 the instantaneous velocity and pressure fields at $t = 200\,\mathrm{s}$ for one realization. We point out that the main difference with respect to figures 7.9 and 7.10 is the roughness height, that here is equal to $0.3\,\mathrm{m}$.

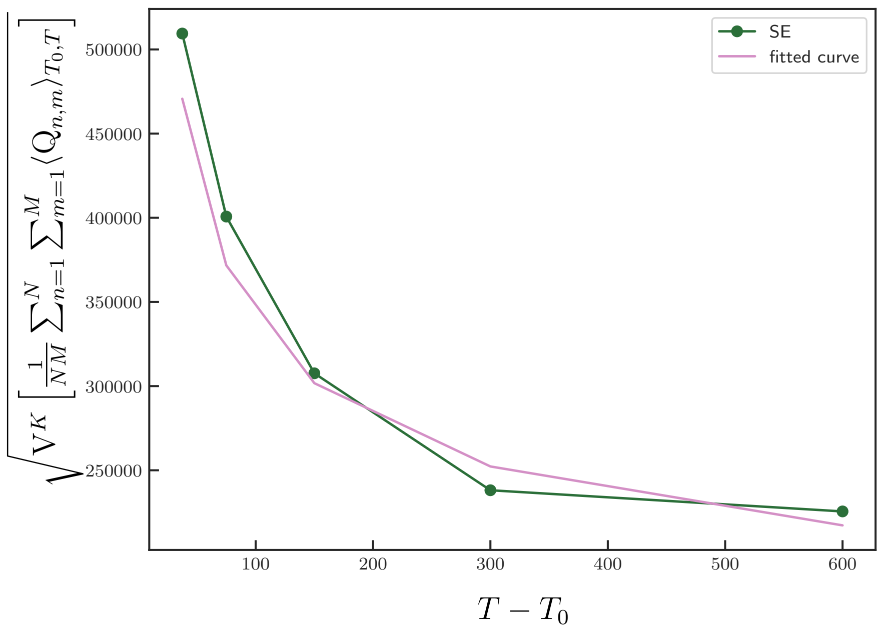


Figure 9.14: Computation of SE as function of $T - T_0$ for the drag force $\mathrm{Q} \equiv F_d$. $K = 40$, $N = 1$ and $M = 1$. The fitted curve is equation (9.31).

$\mathrm{E}^{N,M}[\langle F_d\rangle]$	$\frac{\mathcal{C}_\phi \mathrm{SE}}{\mathrm{E}^{N,M}[\langle F_d\rangle]}$	N	M	$T - T_0$	T_0	C	time to solution
10219325	0.80%	40	1	600	30	19114	37.27
9987287	0.81%	40	2	285	30	18607	18.91
10266800	0.75%	40	4	127.5	30	18625	9.58
10115719	1.07%	40	8	48.75	30	18743	4.85

Table 9.9: The table reports the drag force F_d expected value estimation and its associated SE, with a 99% confidence. N, M, $T - T_0$ and T_0 refer to the number of wind realizations, the number of ensembles per wind scenario, the effective time window and the burn-in time, respectively. C is the computational cost, expressed in CPU hours. Time to solution is the wall clock time we need to wait for solving the problem, and it is expressed in hours. Results are sorted for decreasing time to solution.

Other observables

We select the case with $N = 40$, $M = 1$, $T - T_0 = 600\,\mathrm{s}$ to show statistical results of the physical quantities we compute, that are the drag force, the base moment (computed around the base of the building) and the pressure field around the building. Table 9.11

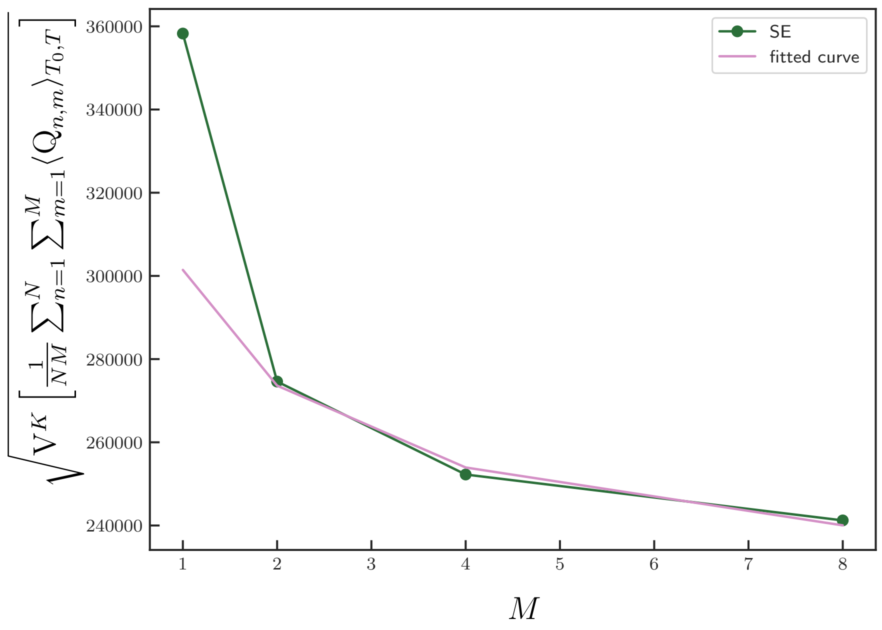


Figure 9.15: Computation of SE as function of M for the drag force $\mathrm{Q} \equiv F_d$. $K = 40$, $N = 1$ and $T - T_0 = 75\,\mathrm{s}$. The fitted curve is equation (9.31).

$\mathrm{E}^{N,M}[\langle F_d\rangle]$	$\frac{\mathcal{C}_\phi \mathrm{SE}}{\mathrm{E}^{N,M}[\langle F_d\rangle]}$	N	M	$T - T_0$	T_0	C	time to solution
10219325	0.80%	40	1	600	30	19114	37.27
10131701	0.45%	320	1	48.75	30	19133	4.95

Table 9.10: The table reports the drag force F_d expected value estimation and its associated SE, with a 99% confidence. N, M, $T - T_0$ and T_0 refer to the number of wind realizations, the number of ensembles per wind scenario, the effective time window and the burn-in time, respectively. C is the computational cost, expressed in CPU hours. Time to solution is the wall clock time we need to wait for solving the problem, and it is expressed in hours. Results are sorted for decreasing time to solution.

shows expected value and standard deviation estimations of the drag force and the base moment. Figures 9.18 and 9.19 show expected value and standard deviation estimations of the pressure field around the building.

We report in table 9.12 the CVaR results for the time-averaged drag force and the drag force.

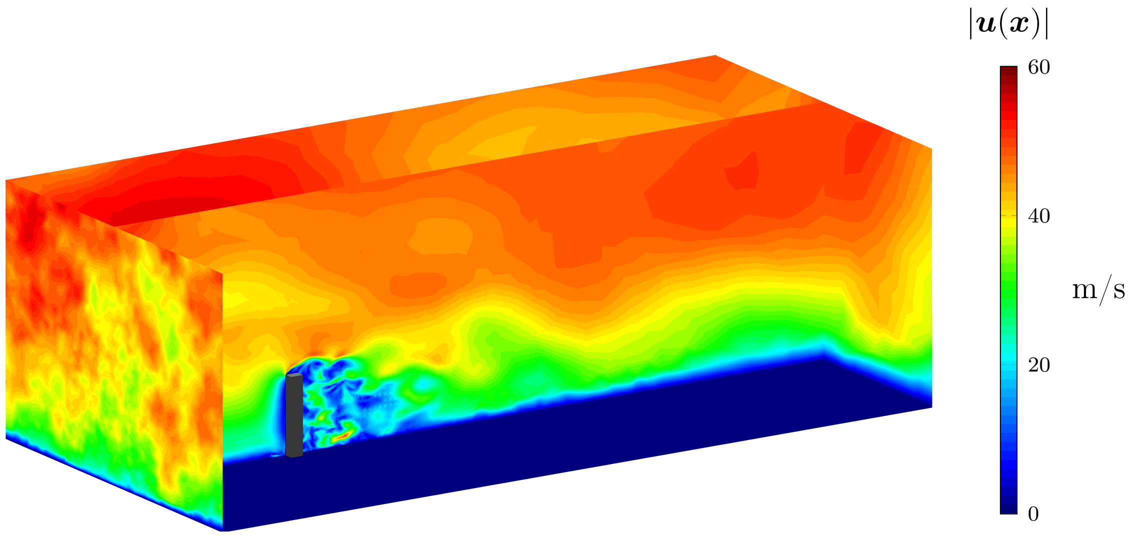


Figure 9.16: Velocity field snapshot at $t = 200\,\mathrm{s}$.

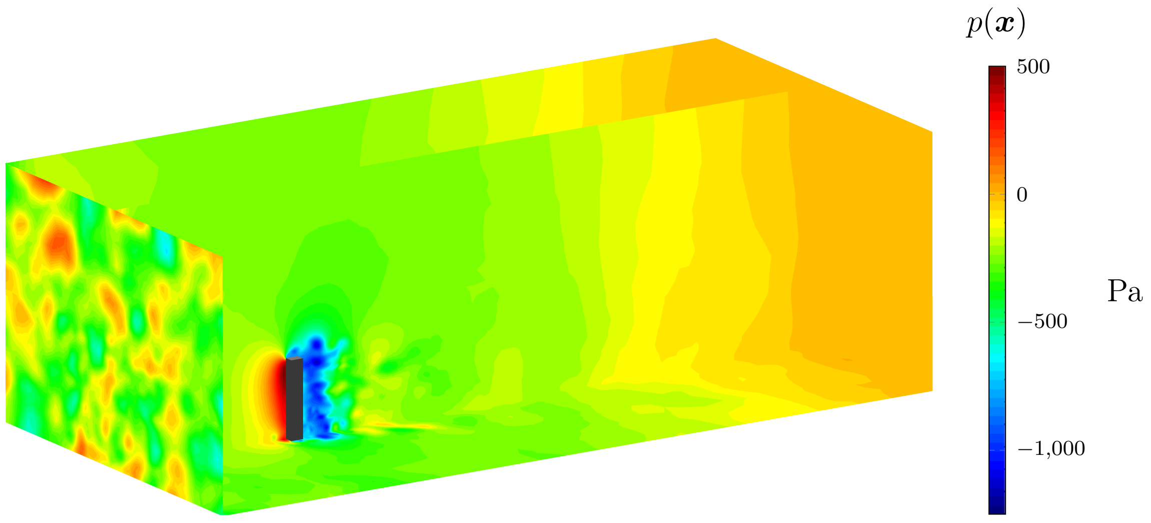


Figure 9.17: Pressure field snapshot at $t = 200\,\mathrm{s}$.

Q	$\mathrm{E}^N[\langle \mathrm{Q}\rangle_{T_0,T}]$	$\sigma^N[\langle \mathrm{Q}\rangle_{T_0,T}]$	$\sigma^N[\mathrm{Q}]$
F_d	10219325	222721	1742534
M_b	41210	321223	9397070

Table 9.11: Statistical analysis of the drag force F_d and of the base moment M_b.

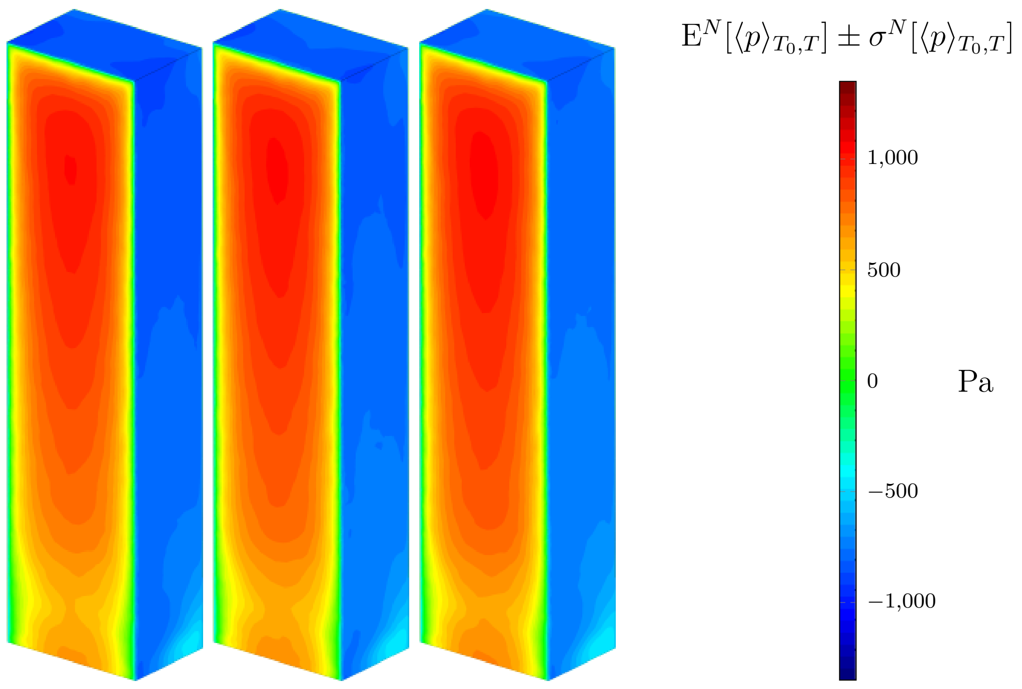

$E^N[\langle p\rangle_{T_0,T}] \pm \sigma^N[\langle p\rangle_{T_0,T}]$
1,000
500
0
−500
−1,000
Pa

Figure 9.18: Statistical analysis of the pressure field $\langle p(x)\rangle_{T_0,T}$. From left to right, $E^N[\langle p(x)\rangle_{T_0,T}] - \sigma^N[\langle p(x)\rangle_{T_0,T}]$, $E^N[\langle p(x)\rangle_{T_0,T}]$ and $E^N[\langle p(x)\rangle_{T_0,T}] + \sigma^N[\langle p(x)\rangle_{T_0,T}]$.

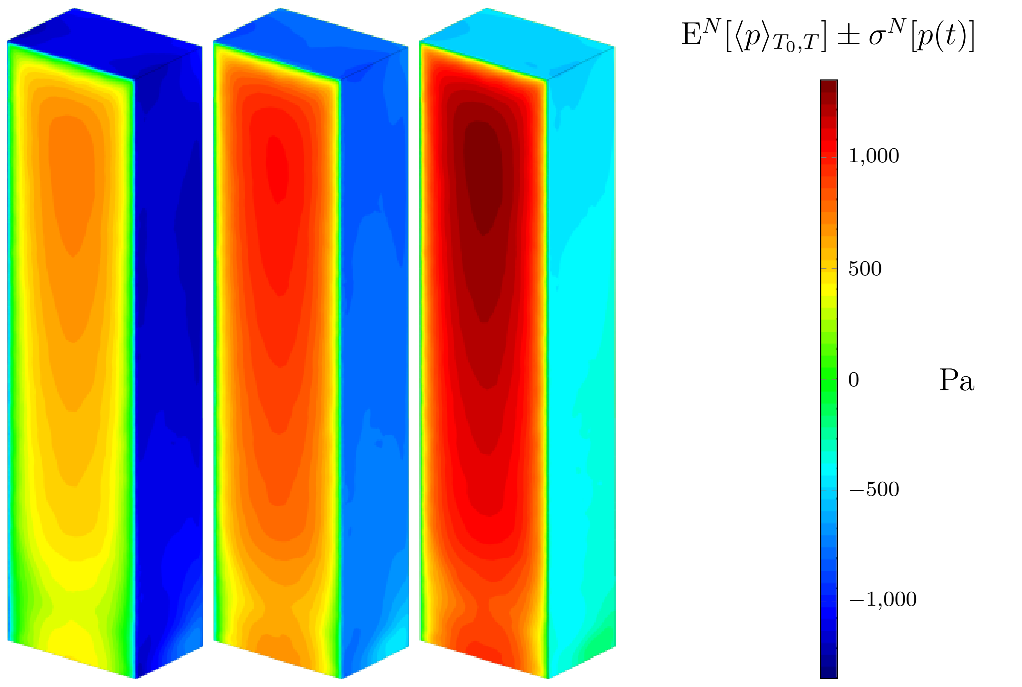


Figure 9.19: Statistical analysis of the pressure field $p(x)$. From left to right, $E^N[\langle p(x)\rangle_{T_0,T}] - \sigma^N[p(x)]$, $E^N[\langle p(x)\rangle_{T_0,T}]$ and $E^N[\langle p(x)\rangle_{T_0,T}] + \sigma^N[p(x)]$.

Q	CVaR	α
$\langle F_d \rangle_{T_0,T}$	10652026	0.9
F_d	13535051	0.9

Table 9.12: CVaR analysis of time-averaged drag force $\langle F_d \rangle_{T_0,T}$ and drag force F_d. Results for $N = 40$, $M = 1$, $T - T_0 = 600\,\mathrm{s}$ and $\alpha = 0.9$.

9.3 Summary

This chapter introduces a framework that integrates the Monte Carlo method and the ensemble averaging technique for solving chaotic problems characterized by both ergodic and non ergodic random variables. Our objective is to find the optimal configuration of the framework that minimizes the computational cost and/or the time to solution of the simulation, while satisfying a failure probability convergence criterion. An error analysis, inspired by our work in chapter 8, is conducted, and two sources of error are identified: an initialization bias and a statistical error. The convergence rate of the SE in the optimal case of fast decay of the transient perturbation is derived.

The framework is validated by solving chaotic flow systems with stochastic boundary conditions and with time horizon constraints typical of wind engineering problems. For all numerical examples, decay rates are estimated to assess if the initialization bias is negligible. We achieve speedups up to a factor 10 by using the ensemble-based MC method, for the same computational cost.

Statistics of multiple observables (drag force, base and pitching moment and pressure field) are efficiently and accurately computed by applying the proposed ensemble-based MC framework. Therefore, decisions based on top of such statistics can be taken faster.

Chapter 10

Conclusion

In this chapter we present the achievements, the new contributions and future research lines of this thesis work. First, we report the thesis objectives set in chapter 1 and we comment if they have been fulfilled or not.

- Extend hierarchical MC methods to efficiently run on HPC environments and a new class of asynchronous hierarchical MC algorithms is introduced. This objective is completed, as shown in chapters 4 and 5. The framework is implemented within the Kratos open source software [44, 43] and the XMC open source software [13] and is tested on different supercomputers. The implemented methods are applied in chapters 7–9 to solve the problems of interest on HPC systems.

- Integrate hierarchical MC methods with AMR strategies for performing UQ. This objective is completed. The software development is concluded, and a preliminary study about the different strategies we propose is conducted in chapter 6. However, due to the difficulties of validating standard MLMC hypotheses for our target systems, we do not apply such strategies for solving the problems of interest, but we limit to using single-level methods and AMR.

- Introduce different strategies for minimizing the overall computational cost and the time to solution of ensemble-based methods. This objective is completed, and results are reported in chapter 8. Exhaustive comparisons are conducted and show the consistency and the efficacy of our proposals.

- Integrate ensemble-based and hierarchical MC methods for solving time-dependent stochastic problems. This objective is completed, as shown in chapter 9. Exhaustive comparisons between hierarchical MC methods and the joint of ensemble averaging and hierarchical MC are conducted, and the advantage of using the ensemble-based MC method for the class of problems we are interested in is demonstrated.

The achievements discussed in previous chapters are summarized in section 10.1. Generic conclusions of this manuscript follow in section 10.2. Finally, future research lines conclude this work in section 10.3.

10.1 Achievements

The first remarkable scientific contribution of this thesis is the development of the asynchronous hierarchical MC methods, presented in chapter 5. The main difference with respect to state of the art hierarchical MC methods is the new level of parallelism, between batches, that permits to maximize the CPU usage during the simulation runtime. We show the superiority of the asynchronous methods over their synchronous counterparts when running on HPC systems, since they require a smaller wall clock time to solve the problem, for the same accuracy. Moreover, a strong scalability test shows the optimal performance of our implementation.

After assessing the correct implementation and the efficiency of our framework, we enhance it by proposing an integration of AMR strategies and hierarchical MC methods in chapter 6. To this end, we first discuss different error estimator strategies for performing AMR and then present the integration of AMR strategies and hierarchical MC methods, which makes use of persistent storage to increase parallelism between solver tasks. Two proposals are discussed, and we show that the stochastic adaptive refinement MLMC method is more promising for complex problems.

In order for our work to be relevant for industrial applications, in chapters 7–9 we focus on complex problems of engineering interest. Specifically, we solve chaotic flows around obstacles and buildings. Different combinations of settings are considered, and therefore these problems are solved with different methods.

With the aim of reducing the overall wall clock time of the simulation, we introduce in chapter 8 the ensemble averaging method, which makes use of concurrency capabilities of HPC systems for solving chaotic and ergodic problems. We first verify that the ensemble averaging method works for high Re flows. Then, we present different strategies for reducing the time duration and the computational cost of the initial burn-in phase.

We verify that the ensemble averaging method outperforms standard time averaging, drastically reducing the overall time to solution.

Finally, we combine ensemble averaging and the MC method in chapter 9, in order to solve wind engineering problems with stochastic boundary conditions. We observe that important time to solution reduction can be achieved by using the ensemble-based MC method.

10.2 Closure

We can say that our developments go in the direction of making accessible to the industry solving problems of science and engineering interest with state of the art techniques in computational mechanics, uncertainty quantification, and high performance computing. In fact, we present strategies that are non-intrusive and can easily be integrated with existing software.

It is known that the energy consumption of a modern supercomputer is very demanding[1]. Within this thesis work, we mention two points that are crucial and that try to address this problem: make efficient usage of modern supercomputers and minimize the wall clock time the user has to wait for solving the simulation. We demonstrate that our proposals achieve optimal strong scalability and maximize the CPU usage of supercomputers. Moreover, we observe that important time to solution reductions can be achieved by applying ensemble-based methods, thus making it possible to take decisions faster.

In addition, we never forget about accuracy and reliability, which are the pillars of modern computational science. The asynchronous hierarchical MC methods present the same accuracy as state of the art methods. Moreover, we demonstrate that the statistical ensemble averaging framework and the statistical ensemble-based MC framework allow for accurate and reliable statistical estimator computations.

In this manuscript, the complex engineering problems we solve are mainly chaotic fluid flows around obstacles and buildings. As commented in chapter 1, the choice of such systems is motivated by the ExaQUte project. However, as remarked in previous chapters, all of our developments can be easily applied to other systems, as for example is done in this master's degree thesis [99], where the author applies our framework to solve stochastic structural stability analysis problems.

[1]The energy consumption of the MareNostrum 4 supercomputer is 1.3 MWatt/year [19].

Recalling the objectives presented in chapter 1, we can say that all of them are accomplished. The hierarchical MC methods, the ensemble averaging method and the ensemble-based MC method are developed and available within the Kratos open source software [95] and the XMC open source software [13]. All of the examples presented in this thesis work are open source and available at [96].

Last but not least, we would like to comment that our framework has been successfully used by an engineering company. The company applied the framework developed within this thesis work to solve the problem of wind flowing past a high-rise building with a parametrically triangulated facade [22, sections 4 and 5].

10.3 Future research lines

We try to give an overview of the remaining improvements and future research lines that may follow from this thesis work.

As commented in chapter 5, the update on the fly of the hierarchy of asynchronous methods is non-adaptive. This implies that no screening phase is required, and therefore all of the computational resources are exploited from the very beginning. A natural extension is considering adaptive hierarchy updates. This implies that an initial screening phase is run, where, probably, not all of the computational resources are exploited. Then, once the hierarchy is adaptively estimated, the required number of simulations are launched and the machine is fully utilized, as for the asynchronous non-adaptive MC methods. We mention that a possible challenge we may face when implementing the asynchronous adaptive methods is the modification on the fly of the hierarchy of batches which have already been launched, since this is managed by the programming model for distributed computing.

As mentioned in chapter 1, the framework we develop is compatible with both OpenMP and MPI parallelisms. In section 5.4 we demonstrate that the single-level asynchronous MC method achieves optimal scalability and efficiency when integrated with the MPI parallel solver Kratos. We comment that it would be interesting to conduct a scalability test exploiting MPI parallelism and multi-level methods, similar to what we do for the OpenMP parallel case of section 5.4. In [12] we solve the stochastic wind flow around a modern high-rise building system (see figure 10.1) using multifidelity Monte Carlo methods and the MPI parallel solver Kratos. The multifidelity Monte Carlo method uses multiple levels, and an idea can be to conduct the scalability test on top of this numerical experiment.

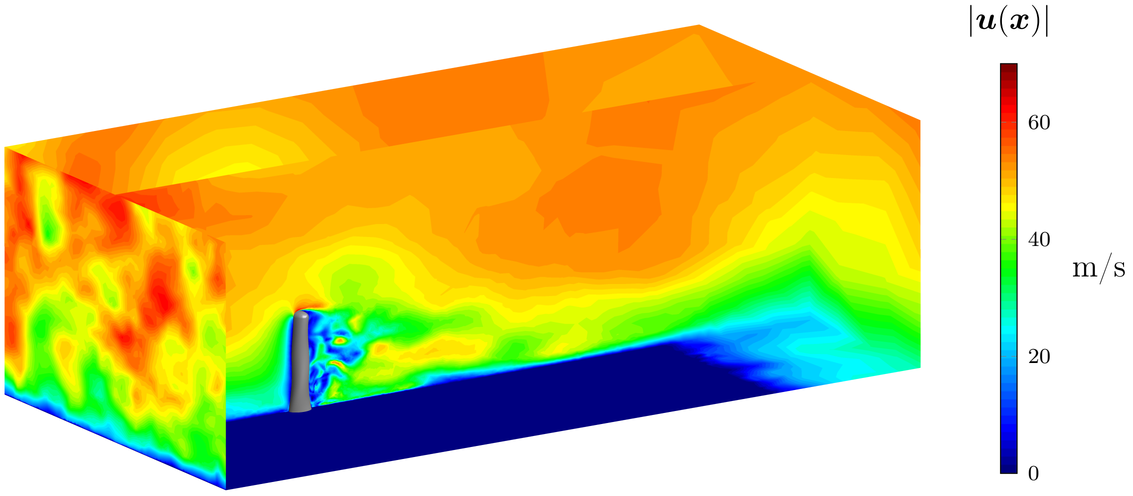


Figure 10.1: Velocity field snapshot at $t = 400\,\mathrm{s}$ of wind flowing past a modern high-rise building.

Concerning the application of the MLMC method to fluid dynamics problems, it would be important to explore up to which Reynolds number we can benefit from the use of multi-level methods to accelerate the solution of stochastic fluid flow problems. Moreover, it would be interesting to apply the deterministic adaptive refinement MLMC method and the stochastic adaptive refinement MLMC method to solve complex systems satisfying the MLMC hypotheses.

Bibliography

[1] M. Ainsworth and J. T. Oden. A posteriori error estimation in finite element analysis, 1997.

[2] F. Alauzet. Metric-based anisotropic mesh adaptation, 2007.

[3] F. Alauzet and P. Frey. Estimateur d'erreur géométrique et métriques anisotropes pourl'adaptation de maillage. Partie I: aspects théoriques. Technical report, INRIA, RR-4759, 2003.

[4] R. Amela, Q. Ayoul-Guilmard, S. Ganesh, R. Tosi, R. M. Badia, F. Nobile, R. Rossi, and C. Soriano. D5.2 Release of ExaQUte MLMC Python engine. Technical report, Open Access Repository of the ExaQUte project: Deliverables, 2019.

[5] R. Amela, R. M. Badia, S. Böhm, R. Tosi, C. Soriano, and R. Rossi. D4.2 Profiling report of the partner's tools, complete with performance suggestions. Technical report, Open Access Repository of the ExaQUte project: Deliverables, 2019.

[6] M. S. Andre. *Aeroelastic Modeling and Simulation for the Assessment of Wind Effects on a Parabolic Trough Solar Collector*. PhD thesis, Technische Universität München, 2018.

[7] J. Ang, K. Evans, A. Geist, M. Heroux, P. Hovland, O. Marques, L. C. McInnes, E. Ng, and S. Wild. Report on the workshop on extreme-scale solvers: Transitions to future architectures. Technical report, U.S. Department of Energy, 2012.

[8] E. Angelino, E. Kohler, A. Waterland, M. Seltzer, and R. P. Adams. Accelerating MCMC via parallel predictive prefetching. In *Uncertainty in Artificial Intelligence - Proceedings of the 30th Conference, UAI 2014*, 2014.

[9] Aurelien Guichard. 30 St Mary Axe from Leadenhall Street, 2010. [Online; accessed October 14, 2021; `https://creativecommons.org/licenses/by-sa/2.0/legalcode`].

[10] Q. Ayoul-Guilmard, R. M. Badia, J. Ejarque, M. Núñez, S. Ganesh, F. Nobile, C. Soriano, C. Roig, R. Rossi, and R. Tosi. D1.3 First public Release of the solver. Technical report, Open Access Repository of the ExaQUte project: Deliverables, 2020.

[11] Q. Ayoul-Guilmard, S. Ganesh, F. Nobile, R. M. Badia, J. Ejarque, B. Keith, A. Kodakkal, M. Núñez, C. Roig, R. Rossi, R. Tosi, and C. Soriano. D1.4 Final public Release of the solver. Technical report, Open Access Repository of the ExaQUte project: Deliverables, 2020.

[12] Q. Ayoul-Guilmard, S. Ganesh, F. Nobile, M. Núñez, and R. Tosi. D5.5 Report on the application of MLMC to wind engineering applications. Technical report, Open Access Repository of the ExaQUte project: Deliverables, 2021.

[13] Q. Ayoul-Guilmard, S. Ganesh, F. Nobile, R. Rossi, R. Tosi, R. M. Badia, and R. Amela. XMC, 2020.

[14] Q. Ayoul-Guilmard, S. Ganesh, M. Núñez, R. Tosi, F. Nobile, R. Rossi, and C. Soriano. D5.3 Report on theoretical work to allow the use of MLMC with adaptive mesh refinement. Technical report, Open Access Repository of the ExaQUte project: Deliverables, 2020.

[15] Q. Ayoul-Guilmard, S. Ganesh, M. Núñez, R. Tosi, F. Nobile, R. Rossi, and C. Soriano. D5.4 Report on MLMC for time dependent problems. Technical report, Open Access Repository of the ExaQUte project: Deliverables, 2020.

[16] R. M. Badia, J. Conejero, C. Diaz, J. Ejarque, D. Lezzi, F. Lordan, C. Ramon-Cortes, and R. Sirvent. COMP Superscalar, an interoperable programming framework. *SoftwareX*, 3–4, 2015.

[17] S. Badia and J. V. Gutiérrez-Santacreu. Convergence towards weak solutions of the Navier-Stokes equations for a finite element approximation with numerical subgrid scale modeling. *IMA Journal of Numerical Analysis*, In press, 2013.

[18] J. Banks, J. Carson, B. L. Nelson, and D. Nicol. *Discrete-Event System Simulation (4th Edition).* Prentice Hall, 4 edition, 2004.

[19] Barcelona Supercomputer Center. Marenostrum. `https://www.bsc.es/marenostrum/marenostrum`, 2021. [Online; accessed 29-September-2021].

[20] J. Bennett, R. Grout, P. Pébay, D. Roe, and D. Thompson. Numerically stable, single-pass, parallel statistics algorithms. In *Proceedings - IEEE International Conference on Cluster Computing, ICCC*, 2009.

[21] P. Beyhaghi, S. Alimohammadi, and T. Bewley. Uncertainty Quantification of the time averaging of a Statistics Computed from Numerical Simulation of Turbulent Flow. feb 2018.

[22] S. Bidier, U. Khristenko, R. Tosi, R. Wuchner, A. Michalski, R. Rossi, and C. Soriano. D7.3 Report on UQ results and overall user experience. Technical report, Open Access Repository of the ExaQUte project: Deliverables, 2021.

[23] P. Blondeel, P. Robbe, C. Van hoorickx, G. Lombaert, and S. Vandewalle. Multilevel Monte Carlo for uncertainty quantification in structural engineering. 2018.

[24] A. L. Braun and A. M. Awruch. Aerodynamic and aeroelastic analyses on the CAARC standard tall building model using numerical simulation. *Computers and Structures*, 87(9-10):564–581, may 2009.

[25] F. Brezzi and M. Fortin. *Mixed and Hybrid Finite Element Methods.* Springer Series in Computational Mathematics. Springer New York, 1991.

[26] J. A. S. Brian M. Adams and Michael S. Eldred and Gianluca Geraci and Russell W. Hooper and And, John D. Jakeman and Kathryn A. Maupin and Jason A. Monschke and Ahmad A. Rushdi and Wildey, L. P. Swiler, and T. M. Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.10 User's Manual. Technical report, SAND2014-4633 Unlimited Release, July 2014. Updated May 15, 2019, 2014.

[27] A. E. Brockwell. Parallel Markov Chain Monte Carlo simulation by pre-fetching. *Journal of Computational and Graphical Statistics*, 2006.

[28] L. Bruno, M. V. Salvetti, and F. Ricciardelli. Benchmark on the aerodynamics of a rectangular 5:1 cylinder: An overview after the first four years of activity. *Journal of Wind Engineering and Industrial Aerodynamics*, 126:87–106, mar 2014.

[29] T. F. Chan, G. H. Golub, and R. J. LeVeque. Updating formulae and a pairwise algorithm for computing sample variances. In H. Caussinus, P. Ettinger, and R. Tomassone, editors, *COMPSTAT 1982 5th Symposium held at Toulouse 1982*, pages 30–41, Heidelberg, 1982. Physica-Verlag HD.

[30] T. F. Chan, G. H. Golub, and R. J. Leveque. Statistical computing: Algorithms for computing the sample variance: Analysis and recommendations. *American Statistician*, 37(3):242–247, 1983.

[31] A. J. Chorin. A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics*, 2(1):12–26, 1967.

[32] A. J. Chorin. Numerical solution of the Navier-Stokes equations. *Mathematics of Computation*, 22(104):745–745, 1968.

[33] V. Cima, S. Böhm, J. Martinovič, J. Dvorsky̌, K. Janurová, T. V. Aa, T. J. Ashby, and V. Chupakhin. HyperLoom: A Platform for Defining and Executing Scientific Pipelines in Distributed Environments. In *Proceedings of the 9th Workshop and 7th Workshop on Parallel Programming and RunTime Management Techniques for Manycore Architectures and Design Tools and Architectures for Multicore Embedded Computing Platforms*, pages 1–6. ACM, 2018.

[34] K. A. Cliffe, M. B. Giles, R. Scheichl, and A. L. Teckentrup. Multilevel Monte Carlo methods and applications to elliptic PDEs with random coefficients. *Computing and Visualization in Science*, 14(1):3–15, 2011.

[35] R. Codina. Stabilized finite element approximation of transient incompressible flows using orthogonal subscales. *Computer Methods in Applied Mechanics and Engineering*, 191(39-40):4295–4321, 2002.

[36] R. Codina, S. Badia, J. Baiges, and J. Principe. Variational Multiscale Methods in Computational Fluid Dynamics. In *Encyclopedia of Computational Mechanics Second Edition*, pages 1–28. John Wiley & Sons, Ltd, Chichester, UK, 2017.

[37] R. Codina, J. Principe, O. Guasch, and S. Badia. Time dependent subscales in the stabilized finite element approximation of incompressible flow problems. *Computer Methods in Applied Mechanics and Engineering*, 196(21-24):2413–2430, 2007.

[38] N. Collier, A. L. Haji-Ali, F. Nobile, E. von Schwerin, and R. Tempone. A continuation multilevel Monte Carlo algorithm. *BIT Numerical Mathematics*, 55(2):399–432, 2015.

[39] O. Colomés, S. Badia, R. Codina, and J. Principe. Assessment of variational multiscale models for the large eddy simulation of turbulent incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 285:32–63, mar 2015.

[40] O. Colomés, S. Badia, and J. Principe. Mixed finite element methods with convection stabilization for the large eddy simulation of incompressible turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, 304(1):294–318, jun 2016.

[41] A. Cornejo, V. Mataix, F. Zárate, and E. Oñate. Combination of an adaptive remeshing technique with a coupled FEM–DEM approach for analysis of crack propagation problems. *Computational Particle Mechanics*, 7(4):735–752, 2020.

[42] J. Cotela, E. Onate, and R. Rossi. *Applications of turbulence modeling in civil engineering*. PhD thesis, Universitat Politècnica de Catalunya, 2016.

[43] P. Dadvand, R. Rossi, M. Gil, X. Martorell, J. Cotela, E. Juanpere, S. R. Idelsohn, and E. Oñate. Migration of a generic multi-physics framework to HPC environments. *Computers and Fluids*, 80(1):301–309, 2013.

[44] P. Dadvand, R. Rossi, and E. Oñate. An object-oriented environment for developing finite element codes for multi-disciplinary applications. *Archives of Computational Methods in Engineering*, 17(3):253–297, 2010.

[45] C. Dapogny, C. Dobrzynski, and P. Frey. Three-dimensional adaptive domain remeshing, implicit domain meshing, and applications to free and moving boundary problems. *Journal of Computational Physics*, 262:358–378, apr 2014.

[46] M. Davari, R. Rossi, P. Dadvand, I. López, and R. Wüchner. A cut finite element method for the solution of the full-potential equation with an embedded wake. *Computational Mechanics*, 63(5):821–833, 2019.

[47] M. Drela. *Flight Vehicle Aerodynamics.* FLIGHT VEHICLE AERODYNAMICS. MIT Press, 2014.

[48] D. Drzisga, B. Gmeiner, U. Rüde, R. Scheichl, and B. Wohlmuth. Scheduling Massively Parallel Multigrid for Multilevel Monte Carlo Methods. *SIAM Journal on Scientific Computing*, 39(5):S873–S897, 2017.

[49] R. Durrett. *Probability: Theory and Examples.* Cambridge University Press, 2019.

[50] P. S. Dwyer. Moments of Any Rational Integral Isobaric Sample Moment Function. *The Annals of Mathematical Statistics*, 8(1):21–65, 1937.

[51] M. Eigel, C. Merdon, and J. Neumann. An adaptive multilevel monte carlo method with stochastic bounds for quantities of interest with uncertain data. *SIAM-ASA Journal on Uncertainty Quantification*, 4(1):1219–1245, 2016.

[52] J. Ejarque, R. Tosi, M. Núñez, S. Böhm, and R. M. Badia. D4.5 Framework development and release. Technical report, Open Access Repository of the ExaQUte project: Deliverables, 2021.

[53] H. C. Elman, D. J. Silvester, and A. J. Wathen. *Finite Elements and Fast Iterative Solvers : with Applications in Incompressible Fluid Dynamics: with Applications in Incompressible Fluid Dynamics.* Oxford University Press, 2005.

[54] W. Fang and M. B. Giles. Multilevel Monte Carlo method for ergodic SDEs without contractivity. *Journal of Mathematical Analysis and Applications*, 2019.

[55] P. J. Frey and F. Alauzet. Anisotropic mesh adaptation for CFD computations. *Computer Methods in Applied Mechanics and Engineering*, 194(48-49):5068–5082, 2005.

[56] M. J. Gander. 50 years of time parallel time integration. In T. Carraro, M. Geiger, S. Körkel, and R. Rannacher, editors, *Multiple Shooting and Time Domain Decomposition Methods*, pages 69–113, Cham, 2015. Springer International Publishing.

[57] R. G. Ghanem and P. D. Spanos. *Stochastic Finite Elements: A Spectral Approach.* Springer New York, 2011.

[58] M. B. Giles. Multilevel Monte Carlo Path Simulation. *Operations Research*, 56(3):607–617, 2008.

[59] M. B. Giles. Multilevel Monte Carlo methods. *Acta Numerica*, 24(May):259–328, 2015.

[60] P. R. Halmos. The Theory of Unbiased Estimation. *The Annals of Mathematical Statistics*, 17(1):34–43, 1946.

[61] 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.

[62] S. Heinrich. Monte Carlo complexity of global solution of integral equations. *Journal of Complexity*, 14(2):151–175, 1998.

[63] S. Heinrich. Multilevel monte carlo methods. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 2179, pages 58–67. Springer Verlag, 2001.

[64] S. Heinrich and E. Sindambiwe. Monte Carlo complexity of parametric integration. *Journal of Complexity*, 15(3):317–341, 1999.

[65] H. Hoel, E. Von Schwerin, A. Szepessy, and R. Tempone. Implementation and analysis of an adaptive multilevel Monte Carlo algorithm. *Monte Carlo Methods and Applications*, 2014.

[66] J. D. Holmes and T. K. Tse. International high-frequency base balance benchmark study. *Wind and Structures, An International Journal*, 18(4):457–471, 2014.

[67] S. Huang, Q. S. Li, and S. Xu. Numerical evaluation of wind effects on a tall steel building by CFD. *Journal of Constructional Steel Research*, 2007.

[68] D. W. Hubbard. *How to Measure Anything: Finding the Value of "Intangibles" in Business*. John Wiley & Sons, 2 edition, 2010.

[69] T. J. R. Hughes. Multiscale phenomena: Green's functions, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods. *Computer Methods in Applied Mechanics and Engineering*, 127(1–4):387–401, nov 1995.

[70] T. J. R. Hughes, G. R. Feijóo, L. Mazzei, and J.-B. Quincy. The variational multiscale method—a paradigm for computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 166(1–2):3–24, nov 1998.

[71] J. C. Hunt. A theory of turbulent flow round two-dimensional bluff bodies. *Journal of Fluid Mechanics*, 61(4):625–706, 1973.

[72] J. C. Hunt. Turbulence structure in thermal convection and shear-free boundary layers. *Journal of Fluid Mechanics*, 138:161–184, 1984.

[73] J. JCSS. Probabilistic model code. *Joint Committee on Structural Safety*, 2001.

[74] E. W. Jenkins, V. John, A. Linke, and L. G. Rebholz. On the parameter choice in grad-div stabilization for the Stokes equations. *Advances in Computational Mathematics*, 40(2):491–516, sep 2013.

[75] N. Jiang. A Higher Order Ensemble Simulation Algorithm for Fluid Flows. *Journal of Scientific Computing*, 64(1):264–288, 2015.

[76] N. Jiang and W. Layton. An algorithm for fast calculation of flow ensembles. *International Journal for Uncertainty Quantification*, 4(4):273–301, 2014.

[77] J. C. Kaimal and J. J. Finnigan. *Atmospheric boundary layer flows: their structure and measurement.* Oxford University Press, 1994.

[78] B. Keith, A. Apostolatos, A. Kodakkal, R. Rossi, R. Tosi, B. Wohlmuth, and C. Soriano. D2.3. Adjoint-based error estimation routines. Technical report, Open Access Repository of the ExaQUte project: Deliverables, 2019.

[79] B. Keith, U. Khristenko, and B. Wohlmuth. A fractional PDE model for turbulent velocity fields near solid walls. *Journal of Fluid Mechanics*, 916, 2021.

[80] B. Keith, U. Khristenko, and B. Wohlmuth. Learning the structure of wind: A data-driven nonlocal turbulence model for the atmospheric boundary layer. *Physics of Fluids*, 33(9):095110, sep 2021.

[81] B. Keith, A. Kodakkal, M. Núñez, R. Rossi, R. Tosi, B. Wohlmuth, and R. Wüchner. Towards risk averse structural design optimization with uncertain wind loading: Two-dimensional benchmarks. Workshop on Frontiers of Uncertainty Quantification in Fluid Dynamics, 2019.

[82] D. E. Keyes. Exaflop/s Pourquoi et comment. *Comptes Rendus - Mecanique*, 339(2-3):70–77, 2011.

[83] B. Krasnopolsky, N. Nikitin, and A. Lukyanov. Optimizing generation of multiple turbulent flow states. *Journal of Physics: Conference Series*, 1129(1):12020, 2018.

[84] B. I. Krasnopolsky. An approach for accelerating incompressible turbulent flow simulations based on simultaneous modelling of multiple ensembles. *Computer Physics Communications*, 229:8–19, 2018.

[85] B. I. Krasnopolsky. Generation of multiple turbulent flow states for the simulations with ensemble averaging. *Supercomputing Frontiers and Innovations*, 5(2):55–62, 2018.

[86] S. Krumscheid, F. Nobile, and M. Pisaroni. Quantifying uncertain system outputs via the multilevel Monte Carlo method — Part I: Central moment estimation. *Journal of Computational Physics*, 2020.

[87] G. J. Lord, C. E. Powell, and T. Shardlow. *An Introduction to Computational Stochastic PDEs.* Cambridge Texts in Applied Mathematics. Cambridge University Press, 2014.

[88] F. Lordan, E. Tejedor, J. Ejarque, R. Rafanell, J. Álvarez, F. Marozzo, D. Lezzi, R. Sirvent, D. Talia, and R. M. Badia. ServiceSs: An Interoperable Programming Framework for the Cloud. *Journal of Grid Computing*, 12(1):67–91, 2014.

[89] Y. Luo and Z. Wang. An ensemble algorithm for numerical solutions to deterministic and random parabolic pdes. *SIAM Journal on Numerical Analysis*, 56(2):859–876, jan 2018.

[90] V. Makarashvili, E. Merzari, A. Obabko, A. Siegel, and P. Fischer. A performance analysis of ensemble averaging for high fidelity turbulence simulations at the strong scaling limit. *Computer Physics Communications*, 219:236–245, 2017.

[91] J. Mann. The Spatial Structure of Neutral Atmospheric Surface-Layer Turbulence. *Journal of Fluid Mechanics*, 273, 1994.

[92] J. Mann. Wind field simulation. *Probabilistic Engineering Mechanics*, 13(4):269–282, 1998.

[93] Martidaniel. 2017 BSC Superordenador MareNostrum-4 Barcelona-Supercomputing-Center, 2017. [Online; accessed October 14, 2021; `https://creativecommons.org/licenses/by-sa/4.0/legalcode`].

[94] V. Mataix Ferrándiz. *Innovative mathematical and numerical models for studying the deformation of shells during industrial forming processes with the Finite Element Method.* PhD thesis, Universitat Politècnica de Catalunya, 2020.

[95] V. Mataix Ferrándiz, P. Bucher, R. Rossi, R. Zorrilla, J. Cotela Dalmau, J. Maria, M. A. Celigueta, and G. Casas. KratosMultiphysics/Kratos: KratosMultiphysics 8.1, 2020.

[96] V. Mataix Ferrándiz, R. Tosi, I. de Pouplana, R. Zorrilla, L. Gracia, S. Warnakulasuriya, M. Núñez, B. Chandra, A. Ghantasala, P. Bucher, J. Cotela, A. Franci, F. Arrufat, K. B. Sautter, S. Latorre, J. González-Usúa, A. Geiser, P. Dadvand, M. Maso, D. Baumgaertner, M. A. Celigueta, R. Kikkeri Nagaraja, S. Wenczowski, B. Saridar, C. Roig, M. Zidan, and G. Casas. KratosMultiphysics/Examples: Kratos Examples 8.1, 2020.

[97] Mattes. Climbing B747-400 - Airfoil positions, 2009. [Online; accessed October 14, 2021; `https://creativecommons.org/licenses/by-sa/1.0/legalcode`].

[98] A. Melendo, A. Coll, M. Pasenau, E. Escolano, and A. Monros. www.gidhome.com, 2021.

[99] M. Meßmer. Uncertainty Quantification in Structural Stability Analysis using Monte Carlo Method. Masterarbeit, Technische Universität München, 2020.

[100] S. P. Meyn and R. L. Tweedie. Stability of Markovian processes III: Foster–Lyapunov criteria for continuous-time processes. *Advances in Applied Probability*, 25(3):518–548, 1993.

[101] S. Mishra and C. Schwab. Sparse tensor multi-level Monte Carlo finite volume methods for hyperbolic conservation laws with random initial data. *Mathematics of Computation*, 81(280):1979–2018, 2012.

[102] S. Mishra, C. Schwab, and J. Šukys. Multi-level Monte Carlo finite volume methods for nonlinear systems of conservation laws in multi-dimensions. *Journal of Computational Physics*, 231(8):3365–3388, 2012.

[103] S. Mishra, C. Schwab, and J. Šukys. Multilevel monte carlo finite volume methods for shallow water equations with uncertain topography in multi-dimensions. *SIAM Journal on Scientific Computing*, 34(6), 2012.

[104] G. Nastac, J. W. Labahn, L. Magri, and M. Ihme. Lyapunov exponent as a metric for assessing the dynamic content and predictability of large-eddy simulations. *Physical Review Fluids*, 2(9), 2017.

[105] F. T. Nieuwstadt, J. Westerweel, and B. J. Boersma. *Turbulence: Introduction to theory and applications of turbulent flows*. Springer, 2016.

[106] N. V. Nikitin. Disturbance growth rate in turbulent wall flows. *Fluid Dynamics*, 44(5):652–657, 2009.

[107] E. D. Obasaju. Measurement of forces and base overturning moments on the CAARC tall building model in a simulated atmospheric boundary layer. *Journal of Wind Engineering and Industrial Aerodynamics*, 1992.

[108] P. Pébay, T. B. Terriberry, H. Kolla, and J. Bennett. Numerically stable, scalable formulas for parallel and online computation of higher-order multivariate central moments with arbitrary weights. *Computational Statistics*, 31(4):1305–1325, 2016.

[109] B. Peherstorfer, K. Willcox, and M. Gunzburger. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *SIAM Review*, 60(3):550–591, 2018.

[110] M. Pisaroni. *Multi Level Monte Carlo Methods for Uncertainty Quantification and Robust Design Optimization in Aerodynamics*. PhD thesis, Lausanne, EPFL, 2017.

[111] M. Pisaroni, F. Nobile, and P. Leyland. A Continuation Multi Level Monte Carlo (C-MLMC) method for uncertainty quantification in compressible inviscid aerodynamics. *Computer Methods in Applied Mechanics and Engineering*, 326:20–50, 2017.

[112] S. B. Pope. *Turbulent Flows*. Cambridge University Press, aug 2000.

[113] J. Principe, R. Codina, and F. Henke. The dissipative structure of variational multiscale methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 199(13–16):791–801, feb 2010.

[114] R. T. Rockafellar and J. O. Royset. On buffered failure probability in design and optimization of structures. *Reliability Engineering and System Safety*, 95(5):499–510, may 2010.

[115] R. T. Rockafellar and J. O. Royset. Engineering Decisions under Risk Averseness. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering*, 1(2):04015003, mar 2015.

[116] A. Rohatgi. Webplotdigitizer: Version 4.5, 2021.

[117] S. Ross. *A first course in probability.* Prentice Hall, Upper Saddle River, N.J., 8th edition, 2010.

[118] R. Rossi, M. Lazzari, and R. Vitaliani. Wind field simulation for structural engineering purposes. *International Journal for Numerical Methods in Engineering*, 61(5):738–763, 2004.

[119] R. Rossi, R. Tosi, M. Núñez, B. Keith, B. Wohlmuth, J. Pons-Prats, and J. Principe. A Time-Parallel Monte Carlo Approach for Unsteady Systems. Platform for Advanced Scientific Computing (PASC) Conference, 2021.

[120] P. Sagaut. *Large Eddy Simulations for Incompressible Flows*, volume 3. Springer Berlin, 2000.

[121] E. Simiu and DongHun Yeo. *The Atmospheric Boundary Layer*, chapter 2, pages 17–54. John Wiley & Sons, Ltd, 2019.

[122] L. Szpruch, S. Vollmer, K. Zygalakis, and M. B. Giles. Multi Level Monte Carlo methods for a class of ergodic stochastic differential equations. *ArXiv preprint*, page 30, 2016.

[123] G. R. Tabor and M. H. Baba-Ahmadi. Inlet conditions for large eddy simulation: A review, 2010.

[124] T. Tamura. Towards practical use of LES in wind engineering. *Journal of Wind Engineering and Industrial Aerodynamics*, 96(10-11), 2008.

[125] Y. Tamura and A. Kareem. *Advanced structural wind engineering.* Springer Japan, 2013.

[126] E. Tejedor, Y. Becerra, G. Alomar, A. Queralt, R. M. Badia, J. Torres, T. Cortes, and J. Labarta. PyCOMPSs: Parallel computational workflows in Python. *International Journal of High Performance Computing Applications*, 31(1):66–82, 2017.

[127] R. Tosi, R. Amela, R. Badia, M. Núñez, R. Rossi, and C. Soriano. D1.2 First release of the softwares. Technical report, Open Access Repository of the ExaQUte project: Deliverables, 2019.

[128] R. Tosi, R. Amela, R. Badia, and R. Rossi. A parallel dynamic asynchronous framework for Uncertainty Quantification by hierarchical Monte Carlo algorithms. *Journal of Scientific Computing*, 89(28):25, 2021.

[129] R. Tosi, R. Amela, J. Pons-Prats, R. Badia, and R. Rossi. Scalable distributed asynchronous Monte Carlo algorithms workflow design. Workshop on Frontiers of Uncertainty Quantification in Fluid Dynamics, 2019.

[130] R. Tosi, M. Núñez, B. Keith, J. Pons-Prats, B. Wohlmuth, and R. Rossi. Scalable dynamic asynchronous Monte Carlo framework applied to wind engineering problems. International Conference on Uncertainty Quantification & Optimisation, 2020.

[131] R. Tosi, M. Núñez, B. Keith, J. Pons-Prats, B. Wohlmuth, and R. Rossi. Scalable Dynamic Asynchronous Monte Carlo Framework Applied to Wind Engineering Problems. In M. Vasile and D. Quagliarella, editors, *Advances in Uncertainty Quantification and Optimization Under Uncertainty with Aerospace Applications. Proceedings of the 2020 UQOP International Conference*. Springer International Publishing, 2022.

[132] R. Tosi, M. Núñez, J. Pons-Prats, J. Principe, and R. Rossi. D3.3 Report of ensemble based parallelism for turbulent flows and release of solvers. Technical report, Open Access Repository of the ExaQUte project: Deliverables, 2021.

[133] R. Tosi, M. Núñez, J. Pons-Prats, J. Principe, and R. Rossi. D3.4 Report on the calibration of parallel methods for transient problems in wind engineering. Technical report, Open Access Repository of the ExaQUte project: Deliverables, 2021.

[134] R. Tosi, J. Principe, J. Pons-Prats, and R. Rossi. A parallel dynamic asynchronous framework for Uncertainty Quantification and Ensemble Average. VI ECCOMAS YOUNG INVESTIGATORS CONFERENCE, 2021.

[135] T. Von Kármán. Progress in the statistical theory of turbulence. *Proceedings of the National Academy of Sciences of the United States of America*, 34(11):530, 1948.

[136] Q. Wang, S. A. Gomez, P. J. Blonigan, A. L. Gregory, and E. Y. Qian. Towards scalable parallel-in-time turbulent flow simulations. *Physics of Fluids*, 25(11), 2013.

[137] B. P. Welford. Note on a Method for Calculating Corrected Sums of Squares and Products. *Technometrics*, 4(3):419–420, 1962.

[138] D. Xiu and G. E. Karniadakis. The Wiener–Askey Polynomial Chaos for Stochastic Differential Equations. *SIAM Journal on Scientific Computing*, 24(2):619–644, 2003.